Autonomous navigation system for a rover with robotic arm using convolutional neural networks

Aziz El Mrabet[1], Hicham Hihi[1], Mohammed Khalil Laghraib[1], Mbarek Chahboun[1], Aymane Amalaoui[2]
[1]Laboratory of Engineering, Systems and Applications, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco
[2]Laboratory of Innovative Technologies, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Article Info

Article history:

Received Oct 22, 2024
Revised May 21, 2025
Accepted Jun 8, 2025

Keywords:

Convolutional neural networks
MATLAB
Python
Raspberry Pi 3
Robot arm
V-shaped

ABSTRACT

The aim of this project is to design and develop an autonomous rover equipped with a KUKA robotic arm. This mobile vehicle will be able to move autonomously thanks to the use of machine learning techniques. It will also be able to detect and retrieve objects using the KUKA arm. The rover will feature Mecanum wheels for improved maneuverability and will be controlled by a Raspberry Pi 3 board, with machine learning algorithms implemented using TensorFlow and Python. The development process will follow the V-methodology. The use of such an autonomous rover and its manipulative capabilities opens the way to many practical applications, including sampling in dangerous or difficult-to-access environments, search and rescue operations in the event of natural disasters or industrial accidents, and inspection and maintenance of industrial or construction sites. The rover could also be used for educational purposes, enabling students to explore the concepts of robotics and artificial intelligence.

Corresponding Author:

Aziz El Mrabet
Laboratory of Engineering, Systems and Applications, National School of Applied Sciences
Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: aziz.elmrabet@usmba.ac.ma

1. INTRODUCTION

The rapid advancement of robotics has led to the development of autonomous systems capable of performing intricate tasks with minimal human intervention. A significant breakthrough in this field is the integration of robotic arms with mobile platforms, which enhances their versatility and enables operation in diverse environments. Such systems have found applications in search and rescue, industrial automation, and agricultural robotics, among others. However, despite notable progress in the development of mobile robots and robotic arms, most existing solutions tend to focus on either mobility or manipulation, limiting their ability to perform a broad range of tasks autonomously. For instance, systems designed for autonomous space rendezvous often prioritize navigation using active sensors like light detection and ranging (LiDAR), relegating visual sensors such as cameras to secondary roles [1], [2]. These systems are typically highly specialized and tailored to specific environments, which restricts their broader applicability.

Recent advancements in machine learning and sensor technologies have enabled significant improvements in autonomous navigation and object manipulation [3]. Nevertheless, many of these systems rely on specialized hardware or lack the flexibility required for diverse real-world applications. While studies have demonstrated the effectiveness of machine learning algorithms in robotic systems for tasks such as sorting and object classification [4], the integration of these algorithms with mobile platforms and robotic arms remains underexplored. Furthermore, the development of autonomous rovers equipped with Mecanum wheels enabling omnidirectional movement has provided superior maneuverability compared to traditional wheel configurations [5]. This capability is critical for navigating complex environments, traversing obstacles, and reaching target locations with precision.

The integration of a KUKA robotic arm further enhances the rover's object detection and manipulation capabilities, enabling it to retrieve objects, perform inspections, and execute highly precise tasks. The combination of robotic arms with mobile platforms has been explored in numerous studies, demonstrating their potential across industries such as agriculture and manufacturing [6]–[9]. Advanced machine learning techniques, including deep learning and multitask convolutional neural networks (MCNN), have further improved the adaptability and accuracy of these systems. For example, the YOLO-MCNN model has proven effective in completing multiple tasks, such as target detection, pose estimation, and obstacle segmentation, which are essential for autonomous operations [7], [10]. These advancements reduce the need for manual intervention and enhance operational efficiency.

This paper presents a structured and validated approach to the development of an autonomous rover system, employing the V-methodology a proven process for mission-critical projects requiring high reliability and performance. The rover's design incorporates a Raspberry Pi 3 platform, which processes real-time data from sensors and cameras to guide the vehicle through its environment. Additionally, the study explores the integration of machine learning algorithms, particularly convolutional neural networks (CNNs), which are central to the rover's capabilities. The modular design of the system, developed using Catia V5, is discussed alongside the application of advanced algorithms and the integration of a KUKA robotic arm with the mobile platform. The paper also highlights the importance of secure communication protocols, such as open platform communications unified architecture (OPC UA), and advanced data management techniques in ensuring the reliability and safety of the autonomous system. By providing a comprehensive overview of the development process and design considerations, this paper aims to contribute to the broader field of autonomous robotics. The following sections delve into the system's design, the methodology employed, and the evaluation of its performance, while also discussing implications for future research.

2. METHOD

The development of this robot followed a structured and systematic approach based on the V-methodology. The V-shaped development model is an iterative, incremental approach for software projects that have well-defined requirements but need flexibility. This development cycle follows a V-shape with sequential and parallel steps, including requirements analysis, design, implementation, testing, and validation. This model is particularly well-suited to mission-critical projects; each phase was meticulously executed to ensure the system's reliability, functionality, and reproducibility.

2.1. Requirements analysis

A comprehensive analysis of both functional and non-functional requirements was conducted to define the scope and objectives of the project. The functional requirements focus on the rover's core capabilities, which include autonomous navigation using machine learning techniques, omnidirectional movement enabled by Mecanum wheels, obstacle detection and avoidance using ultrasonic sensors, and object detection and retrieval facilitated by a KUKA arm and a camera-based vision system. These capabilities ensure the rover can operate effectively in dynamic environments, locate target objects, and interact with its surroundings.

In addition to functional requirements, the system was designed to meet several non-functional requirements, such as performance, reliability, and usability. The rover must operate without failure for extended periods, recover from errors autonomously, and provide clear status updates to non-technical users. Hardware specifications include the use of a Raspberry Pi 3 as the central processing unit, Mecanum wheels for enhanced maneuverability, a KUKA arm for object retrieval, and a camera for object detection and recognition. On the software side, the system leverages TensorFlow and Python for machine learning and control algorithms. Finally, the design adheres to specific budget, size, and weight constraints to ensure practicality and feasibility.

2.2. System architecture

The system architecture is modular, with each module performing a specific function and communicating asynchronously via dedicated topics. This design ensures flexibility, scalability, and ease of maintenance. The camera module captures images using the rover's onboard camera and publishes them to the "image_topic" for use by other modules. The object detection module subscribes to this topic and analyzes the pictures with a CNN based on the residual networks (ResNets)-50 architecture. This module detects and identifies objects within the images, publishing the results (e.g., object coordinates and type) to the "object_detection_topic".

The KUKA arm control module subscribes to the "object_detection_topic" to receive object coordinates and calculates the arm movements required to grasp the detected objects using inverse kinematics. It then publishes arm movement commands to the "arm_control_topic". The wheel control module subscribes to this topic and adjusts the rover's Mecanum wheels to position the rover for object retrieval, publishing wheel movement commands to the "wheels_control_topic". Finally, the ultrasonic sensor module measures distances to surrounding objects and publishes this data to the system, enabling obstacle avoidance and navigation.

2.3. Implementation details

The implementation phase involved integrating hardware components and developing software algorithms to bring the system to life. The Raspberry Pi 3 was set up to connect with the Mecanum wheels, KUKA arm, camera, and ultrasonic sensors, functioning as the central hub for data processing and control. The control algorithms were executed in Python, utilizing TensorFlow for CNN-based object detection. The ResNet-50 model was chosen for its balance of accuracy and computational efficiency, and it was fine-tuned for our specific object detection task using a dataset of labeled images.

A publish-subscribe model facilitated communication between modules. This model allows asynchronous and independent operation while ensuring seamless data exchange. This approach enables each module to function autonomously while remaining synchronized with the overall system.

2.4. Verification and validation

Each module was rigorously tested in isolation and as part of the integrated system to ensure it met the specified requirements. The object detection algorithm was validated using a dataset of labeled images. The rover's navigation and object retrieval capabilities were tested in a controlled environment, confirming that all functional and non-functional requirements were met. These tests demonstrated the system's ability to autonomously navigate, detect and avoid obstacles, locate target objects, and retrieve them using KUKA arm.

3. OBJECT DETECTION ALGORITHM

This study focuses on the implementation of machine learning and deep learning methodologies to design an object detection framework. As a branch of artificial intelligence, machine learning allows computational systems to autonomously improve performance through data-driven learning autonomously, bypassing the need for direct programming. Such techniques are applied extensively in domains ranging from voice and facial recognition to target identification systems. Machine learning strategies are typically categorized into three primary types: supervised learning for labeled data analysis, unsupervised learning for pattern discovery in unlabeled datasets, and reinforcement learning for decision-making optimization through iterative feedback.

Deep learning, on the other hand, is a branch of machine learning that exploits artificial neural networks to solve complex problems. As illustrated in Figure 1, deep neural networks, such as CNNs, are particularly well-suited to computer vision and image classification tasks. In our object detection system, we have used a pre-entrained CNN.

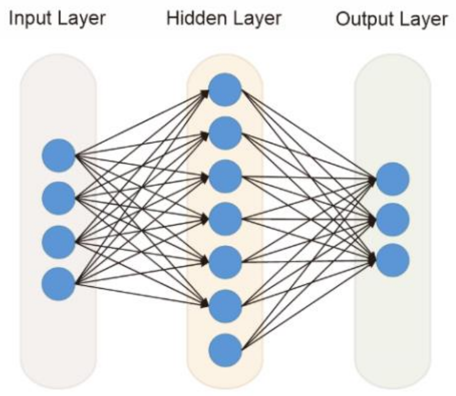


Figure 1. Deep neural network architecture

3.1. Presentation of the neural network convolution algorithm and the ResNet-50 model

CNNs represent a class of deep learning models commonly used for image classification. As shown in Figure 2, they exploit convolution operations to extract relevant features from images, thereby reducing the complexity of the model in terms of parameters to be learned. These convolutional networks are generally composed of convolution, pooling, and fully connected layers, enabling classification to be performed. The network is trained by iteratively adjusting the weights of the various layers, based on the training data. CNN has proven its worth in many computer vision applications [11], [12].

Image
Specific filter
Edges

Figure 2. Edge detection process

The ResNet architecture was selected for this project. ResNet is a deep convolutional network that uses residual connections, making it easy to train very deep models. The network is built from stacked residual convolution blocks. In addition, transfer learning is often employed with ResNet, enabling pre-trained convolution layers to be exploited for image feature extraction.

Early approaches to image processing were based on the use of filters to extract features such as object contours. From a mathematical point of view, this involves the application of convolution operations, which consist of sums of elementary products over image blocks. Convolution can be defined on a 2D matrix and can be extended to volumes [13]. The image is then represented as a tensor, with dimensions for height, width, and number of channels [14], [15].

3.1.1. Padding and stride in convolutions

Padding and stride are key operations for dealing with the loss of information at the edges of the image when applying convolutions. Padding consists of adding zeros around the image to take into account pixels located on the edges, while stride controls the size of the output by adjusting the distance traveled by the filter during convolution. The following section explores these concepts in detail [16], [17].

Padding: when a convolution is applied with a vertical edge filter, pixels in the image's corners are used less than those in the center, leading to a loss of edge information. To solve this problem, it is common to add a frame around the image. As illustrated in Figure 3, this padding usually involves adding zeros around the original image so that pixels at the edges can be taken into account during convolution. The dpi parameter corresponds to the number of elements added to each side of the image.

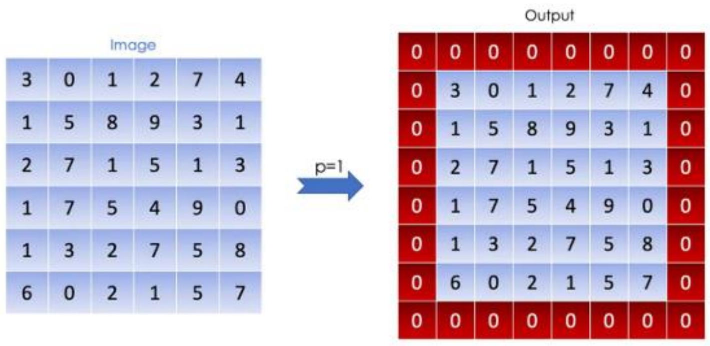


Image					
3	0	1	2	7	4
1	5	8	9	3	1
2	7	1	5	1	3
1	7	5	4	9	0
1	3	2	7	5	8
6	0	2	1	5	7

Output							
0	0	0	0	0	0	0	0
0	3	0	1	2	7	4	0
0	1	5	8	9	3	1	0
0	2	7	1	5	1	3	0
0	1	7	5	4	9	0	0
0	1	3	2	7	5	8	0
0	6	0	2	1	5	7	0
0	0	0	0	0	0	0	0

Figure 3. Padding in CNNs

Stride: the stride corresponds to the speed at which the filter moves over the image during convolution. The size of the output decreases with a larger stride, while a smaller stride keeps it larger. This distance is represented by the s parameter. For example, a stride of 1 indicates that the filter moves one pixel at a time, while a stride of 2 indicates that it moves two pixels at each step. As shown in Figure 4, these concepts are illustrated in the following images, showing an example of padding with $p = 1$ and a convolutional product with $s = 1$.

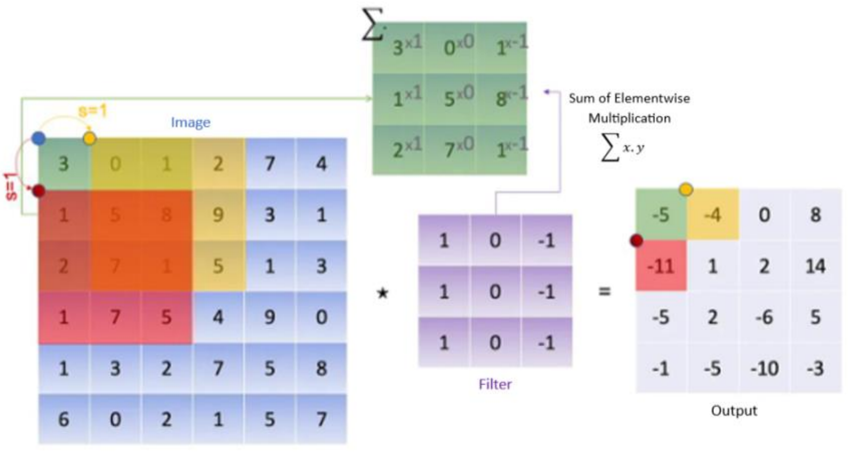


Figure 4. Convolution operation

A rigorous definition of the convolution operation requires clarity on two critical components: padding and stride. Padding, parameterized by p, preserves spatial edge information by appending zeros around the input matrix. Meanwhile, stride (s) governs the displacement interval of the filter during convolution, directly influencing output dimensionality. The convolutional output is computed as a 2D matrix, where each entry corresponds to the summation of element-wise products between the filter's 3D tensor and an overlapping sub-cube of the input tensor. For an image with dimensions $[n_h,\ n_w,\ n_c]$ (n_h: the size of the height, n_w: the size of the widh, and n_c: the number of channels). In the case of an RGB image, for example, $n_C = 3$ we have red, green, and blue. By convention, we consider that the K filter is gridded and has an odd dimension noted f, which allows each pixel to be centered in the filter and therefore to take into account all the elements surrounding it, so that we apply a filter of dimension $[f, f, n_C]$. The convolutional product between the image and the filter is a 2D matrix, each element of which is the sum of the multiplication per element of the cube (filter) and the sub-cube of the given image, as illustrated in Figure 5.

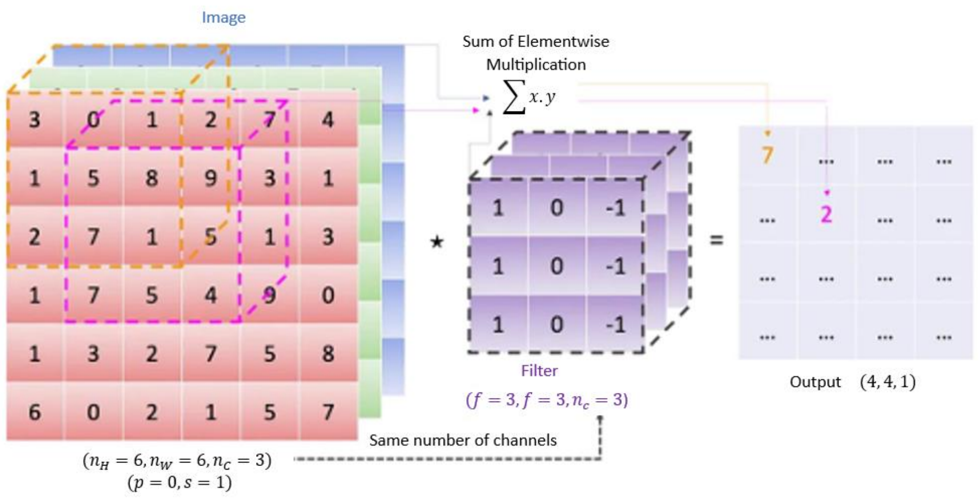


Image
Sum of Elementwise Multiplication
$\sum x.y$
Filter
$(f = 3, f = 3, n_c = 3)$
Output (4, 4, 1)
Same number of channels
$(n_H = 6, n_W = 6, n_C = 3)$
$(p = 0, s = 1)$

Figure 5. The cube (filter) and the sub-cube of the given image

Mathematically, the dimensionality of the convolution operation between an input image I and filter K is defined as in (1).

$$\dim\big(conv(I, K)\big) = \begin{cases} \left(\left[\frac{n_H + 2p - f}{s} + 1\right], \left[\frac{n_w + 2p - f}{s} + 1\right]\right); s > 0 \\ (n_h + 2p - f, n_w + 2p - f)\ ; s = 0 \end{cases} \tag{1}$$

Where $\lfloor x \rfloor$ is the floor function of x. Common convolution variants include:
– Valid convolution: $p = 0$
– Same convolution: output size=input size→ $p = \frac{f-1}{2}$
– 1×1 convolution: Employs a unit-sized filter ($f = 1$), often used to reduce channel depth (n_C) while retaining spatial resolution (n_H, n_W).

In the illustrative example, Figure 5, filter values are manually initialized for clarity. However, in practical CNNs, the $f \times f \times n_C$ filter parameters are optimized automatically via backpropagation during training.

3.1.2. Pooling

Pooling layers downsample spatial dimensions (n_H, n_W) while preserving channel depth (n_C). This operation applies a fixed-aggregation function (non-trainable) to localized regions of the input tensor, traversed by a filter of size $f \times f$ with stride s. The output dimensions are governed by (2).

$$\dim(pooling(image)) = \begin{cases} \left(\left\lfloor \frac{n_H+2p-f}{s} + 1 \right\rfloor, \left\lfloor \frac{n_w+2p-f}{s} + 1 \right\rfloor, n_C\right); s > 0 \\ (n_h + 2p - f, n_w + 2p - f, n_C)\ ; s = 0 \end{cases} \tag{2}$$

Standard practice employs square filters ($f \times f$), typically with $f = 2$ and $s = 2$ to halve the spatial resolution while avoiding overlap. Common pooling functions include:
– Average pooling: computes the mean of values within the filter's receptive field.
– Max pooling: extracts the maximum value from the filter's window.

Unlike convolutional layers, pooling utilizes predefined operations (no learnable parameters), prioritizing computational efficiency and translational invariance in deep networks.

3.1.3. Building a convolutional neural network layer by layer

A CNN is constructed by stacking layers, each performing specific operations like convolution, activation, pooling, and fully connected layers. For example, in the 3rd layer:

Input: $a^{t-1}$with size $\left(n_H^{(l-1)}, n_W^{[l-1]}, n_C^{[l-1]}\right)$, $a^{[0]}$ being the image in the input
Padding: $p^{[l]}$
Stride: $s^{[l]}$
Number of filters: $n_C^{[l]}$where each K^n has the dimension: $\left(f^{[l]}, f^{[l]}, n_C^{[l-1]}\right)$
Bias of the n^{th}convolution $b_n^{[l]}$
Activation function: $\psi^{[l]}$]
Output: $a^{[l]}$ with size $(n_H^{[l]}, n_W^{[l]}, n_C^{[l]})$

and:

$$\begin{gathered} \forall n \epsilon \left[1,2, \dots, n_C^{[l]}\right] conv\left(a^{[l-1]}, K^n\right)_{x,y} = \psi^{[l]}\left(\sum_{i=1}^{n_H^{[l-1]}} \sum_{k=1}^{n_W^{[l-1]}} \sum_{k=1}^{n_C^{[l-1]}} K_{i,j,k}^{(n)} a_{x+i-1,y+j-1,k} + b_n^{[l]}\right) \\ \dim\left(conv\left(a^{[l-1]}, K^{(1)}\right)\right) = \left(n_H^{[l]}, n_W^{[l]}\right)]\ ] \end{gathered} \tag{3}$$

thus:

$$\begin{gathered} a^{[l]} = \left[\psi^{[l]}\left(conv\left(a^{[l-1]}, K^{(1)}\right)\right), \psi^{[l]}\left(conv\left(a^{[l-1]}, K^{(1)}\right)\right), \dots, \psi^{[l]}\left(conv\left(a^{[l-1]}, K^{\left(n_C^{[l]}\right)}\right)\right)\right] \\ \dim\left(a^{[l]}\right) = \left(n_H^{[l]}, n_W^{[l]}, n_C^{[l]}\right) \end{gathered} \tag{4}$$

with:

$$n_{\frac{H}{W}}^{[l]} = \begin{cases} \left\lfloor \frac{n_{\frac{H}{W}}^{[l-1]} + p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor ; s > 0 \\ n_{\frac{H}{W}}^{[l-1]} + 2p^{[l]} - f^{[l]}\ ; s = 0 \end{cases} \tag{5}$$

where $n_C^{[l]} = number\ of\ filters$. The learned parameters at the l^{th} layer are:
- Filters with $\left(f^{[l]} \times f^{[l]} \times n_C^{[l-1]}\right) \times n_C^{[l]}$ parameters.
- bias with $(1 \times 1 \times 1) \times n_C^{[l]}$ parameters (broadcasting).

Pooling layers, as shown in Figure 6, hierarchically abstract spatial features by subsampling inputs while maintaining channel consistency $(n_H^{[l]}, n_W^{[l-1]}, n_C^{[l-1]})$. The layer's operation is formalized as follows:

Input: $a^{[l]}$ (activation from layer $l-1$) with dimensions $\left(n_H^{[l]}, n_W^{[l-1]}, n_C^{[l-1]}\right)$, and $a^{[0]}$ denotes the raw input image.
Parameters: Filter size $f^{[l]}$, stride $s^{[l]}$, and padding $p^{[l]}$ (rarely applied in pooling).
Pooling function: $\phi^{[l]}$ (aggregation operation, e.g., max or average).
Output: $a^{[l]}$ with dimensions $\left(n_H^{[l]}, n_W^{[l]}, n_C^{[l]} = n_C^{[l-1]}\right)$

The output activation at position (x, y, z) in layer ll is computed as:

$$a_{x,y,z}^{[l]} = pool\left(a^{[l-1]}\right)_{x,y,z} = \phi^{[l]}\left(\left(a_{x+i-1,y+j-1,z}\right)_{(i,j)\in[1,2,\dots,f^{[l]}]}\right) \qquad (6)$$
$$\dim\left(a^{[l]}\right) = \left(n_H^{[l]}, n_W^{[l]}, n_C^{[l]}\right)$$

with:

$$n_{\frac{H}{W}}^{[l]} = \begin{cases} \left\lfloor \dfrac{n_{\frac{H}{W}}^{[l-1]} + p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor ; s > 0 \\ n_{\frac{H}{W}}^{[l-1]} + 2p^{[l]} - f^{[l]} ; s = 0 \end{cases} \qquad (7)$$
$$n_C^{[l]} = n_C^{[l-1]}$$

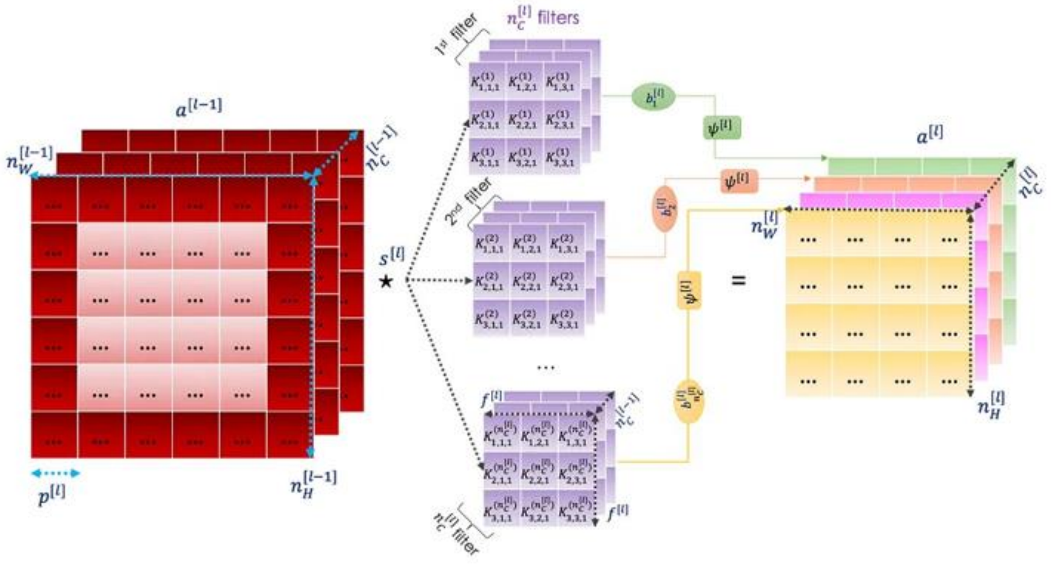


Figure 6. The pooling layers

CNNs are hierarchically structured architectures in Figure 7 that iteratively apply convolutional layers, nonlinear activation functions, and pooling operations. This sequence convolution →activation→pooling is repeated across successive layers to progressively extract higher-order spatial and semantic features from input images. The extracted feature maps are then flattened and fed into fully connected (dense) layers, augmented with additional activations, to perform classification or regression tasks.

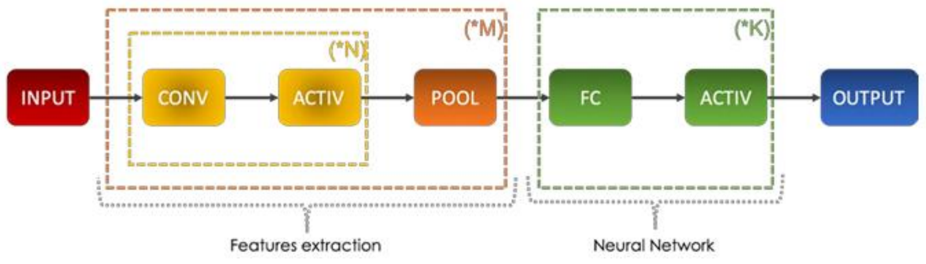


Figure 7. The sequential steps involved in a CNN

A hallmark of CNNs is their ability to systematically reduce spatial resolution while increasing channel depth (CNN) as the network deepens, balancing computational efficiency with representational capacity. This three-dimensional transformation from raw pixel data to abstract feature hierarchies is visualized in Figure 8. This figure illustrates the architectural evolution of tensors across layers.

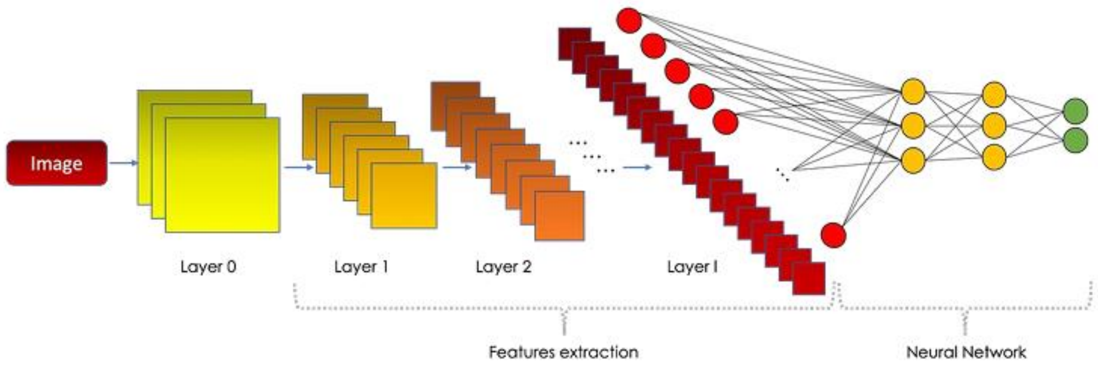


Figure 8. Structure of a CNN

3.1.4. ResNet

ResNets enhance traditional CNN architectures by integrating skip connections that bypass intermediate layers, as shown in Figure 9. These connections mitigate performance degradation in very deep networks by enabling the propagation of unaltered gradients and features. Without skip connections, the behavior of a residual block is governed by standard linear and nonlinear transformations as in (8) and (9).

$$z_j^{[i]} = \sum_{l=1}^{n_{i-1}} w_{j,l}^{[i]} a_l^{[i-1]} + b_j^{[i]} \tag{8}$$

$$\longrightarrow a_j^{[i]} = \psi^{[i]}\left(z_j^{[i]}\right) \tag{9}$$

By initializing weights $w^{[i]}$ and biases $b^{[i]}$ to zero and selecting an identity activation $\psi^{[i]}$, the residual block simplifies to $a^{[i]} = a^{[i-2]}$, preserving network performance even if added layers contribute minimally. To ensure dimensional compatibility between $a^{[i-2]}$, and $a^{[i]}$, same convolutions (matching spatial dimensions) are typically applied. When mismatches occur, a learnable projection tensor W_s adjusts the skip connection:

$$\longrightarrow a_j^{[i]} = \psi^{[i]}\left(z_j^{[i]} + a_j^{[i-2]}\right) \tag{10}$$

where W_s might be a fixed tensor or a learned one, and $\dim(W_s) = \left[n^{[i]}, n^{[i-2]}\right]$.

Inception networks introduce a paradigm shift by deploying modules that perform multiple operations in parallel within a single layer, as shown in Figure 10. Specifically, each Inception module applies convolutional, pooling, and fully connected operations simultaneously, allowing the network to capture features at various spatial scales. This architecture eliminates the need for manual selection of layer types and enhances model efficiency and representational richness.

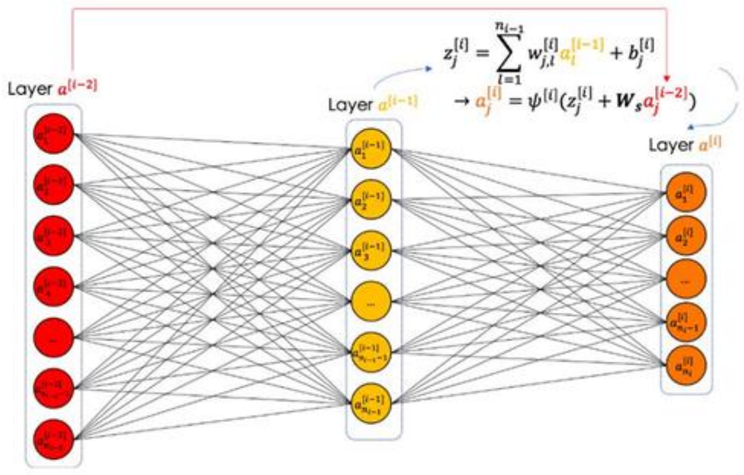

Layer $a^{[l-2]}$
Layer $a^{[l-1]}$
Layer $a^{[l]}$
$z_j^{[l]} = \sum_{l=1}^{n_{l-1}} w_{j,l}^{[l]} a_l^{[l-1]} + b_j^{[l]}$
$\rightarrow a_j^{[l]} = \psi^{[l]}(z_j^{[l]} + W_s a_j^{[l-2]})$

Figure 9. Feedforward neural network architecture

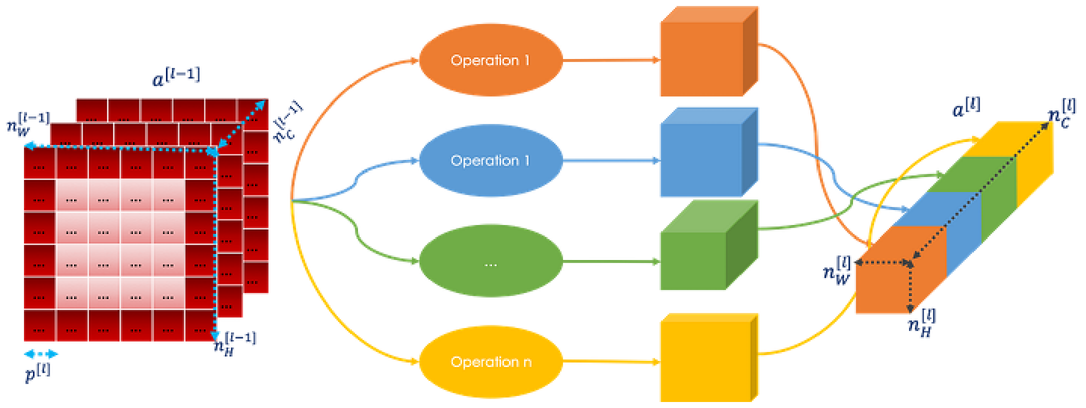


Figure 10. The result of all the operations

3.2. Robot module

3.2.1. Robot Mecanum

Wheeled mobile robots resembling machines are becoming increasingly prevalent and are widely utilized in industrial settings for automated transportation or logistical purposes, such as the conveyance of goods, parts, and even people. When dealing with expensive and sensitive loads, the mobile robot must be reliable and safe while providing efficient movement. A machine equipped with wheels can maneuver around, leading to more efficient usage. A mobile robot capable of serving multiple stations within a production line can enhance product capacity and quality [18], [19].

In this research, an omnidirectional robot is employed, which has the ability to move in any direction. It is a holonomic robot with four specialized wheels, namely the Mecanum wheel system, each driven by a separate stepper motor. For such a robot, the number of controlled degrees of freedom is equal to the total number of degrees of freedom of the robot [20], [21]. It can move in any direction on a planar surface due to its freely rotating rollers placed on the wheel surface at a 45-degree angle. Figure 11 depicts a mobile robot model with a Mecanum wheel, with a coordinate system attached at the center of the wheel hub, where the unit axis is denoted. The robot's position and orientation are represented as. The robot's linear velocity is and its angular velocity is, while is the angular velocity of the i-th wheel and represents the linear wheel velocity. The angle between the free-sliding roller axis and the wheel hub axis can be either positive or negative, depending on whether the wheel is left or right [22]–[24].

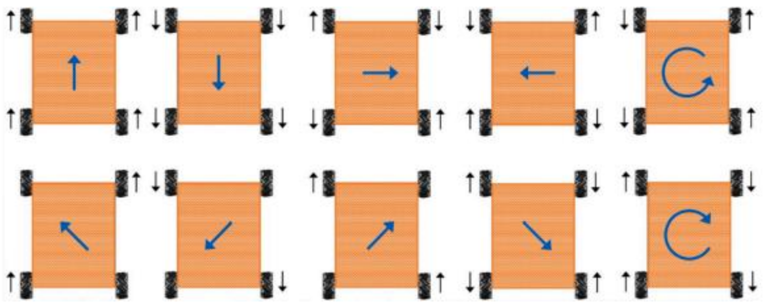


Figure 11. The direction of mobile robot motion is based on the Mecanum's various velocities of the wheels

The direction of the mobile robot's motion based on the Mecanum wheel motion is shown in Figure 11. The following forward kinematic equation describes the relationship between the mobile robot's velocity and the velocity of each wheel as in (12).

$$\begin{bmatrix} v_{xr} \\ v_{yr} \\ w_r \end{bmatrix} = \frac{1}{4r_m}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -\frac{1}{a+b} & \frac{1}{a+b} & \frac{1}{a+b} & -\frac{1}{a+b} \end{bmatrix}\begin{bmatrix} w_{m1} \\ w_{m2} \\ w_{m3} \\ w_{m4} \end{bmatrix} \tag{12}$$

Where r_m is the radius of the Mecanum wheel, and a and b describe the length and width of the mobile robot.

3.2.2. KUKA arm control

Denavit-Hartenberg (DH) parameters are a commonly used method for describing the kinematics of manipulator arms in robots [25], [26]. They provide a systematic way of representing the relationships between the various links of an articulated manipulator arm. The DH parameter method relies on four main parameters for each joint of a manipulator arm: the angle of rotation about the common z-axis (θ), the link length (d), the distance along the x-axis between the common z-axes (a), and the angle of inclination to the common x-axis (α). The homogeneous transformation matrix DH parameters for the KUKA arm are obtained according to this approach by the following relationship and Table 1. Figure 12 shows the KUKA arm robot.

$$A_i = Rot_{z_i\theta_i}Trans_{z_id_i}Trans_{x_ia_i}Rot_{x_i\alpha_i} \tag{13}$$

$$A_i = \begin{bmatrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} & a_iC_{\theta_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} & a_iC_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

Table 1. DH parameters for the KUKA arm

Joint i	$\theta_i(deg)$	$d_i(mm)$	a_i	α_i(deg)
1	θ_1	889	0	90
2	θ_2	0	2340	0
3	θ_3	0	0	90
4	θ_4	1440	0	-90
5	θ_5	40	0	-90

The robot model was imported from SolidWorks using an XML file and then integrated into MATLAB/Simulink. Specifically, we used the “smimport” command, which is part of the Simscape Multibody toolbox, to convert the XML file into a Simulink model containing the 3D mechanical components. After import, inputs were configured to control each joint angle, and outputs were set to read the measured values. To achieve precise motion control, we implemented a feedback loop using a proportional-integral-derivative (PID) controller to minimize deviations between the commanded and measured joint angles. Each joint in the model is therefore associated with both a control input and a corresponding measurement output.

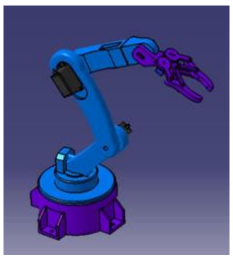


Figure 12. KUKA arm

As shown in Figure 13, we transform this representation into a single block (sub-system). The input from the robot's control blocks is connected to a Simulink block that performs the robot's inverse kinematics. This inverse kinematics block uses a position vector (x, y, and z) as input to determine the joint angles required to achieve the desired position and orientation of the end effector. Using this approach, we can precisely control the robot's motion by specifying the desired position coordinates in three-dimensional space. The PID corrector allows me to adjust the joint angles in real time to reduce deviations between commanded and measured values, ensuring more precise and reliable robot motion as shown in Figure 14.

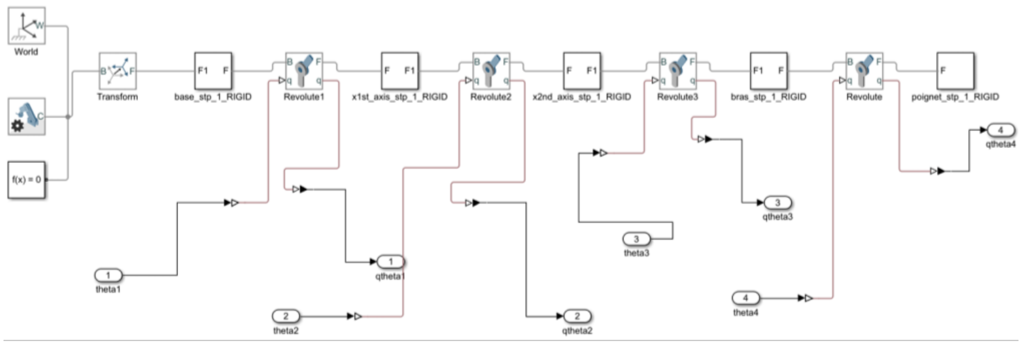


Figure 13. The sub-system module of the arm in Simulink

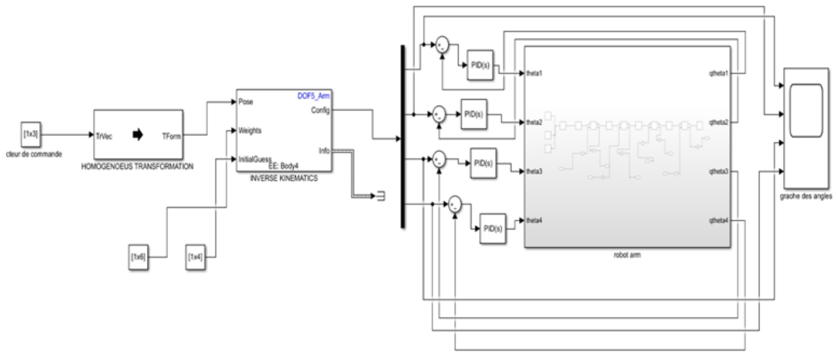


Figure 14. The complete arm model is in Simulink

This control method, based on inverse kinematics and the PID corrector, offers great flexibility in achieving precise robot movements in response to specific end-effector positions and orientations. As shown in Figure 15, the simulation results. For example, x, y, and z values in the control vector are selected and entered. Simulink plays a crucial role in the simulation of intelligent robots by providing an advanced modeling and simulation environment, enabling multi-domain modeling, realistic simulation, control design and motion planning, integration with other tools, and model validation. It facilitates the development, optimization, and performance verification of robots before their actual implementation. This helps to speed up the design process and improve the reliability and performance of intelligent robots.

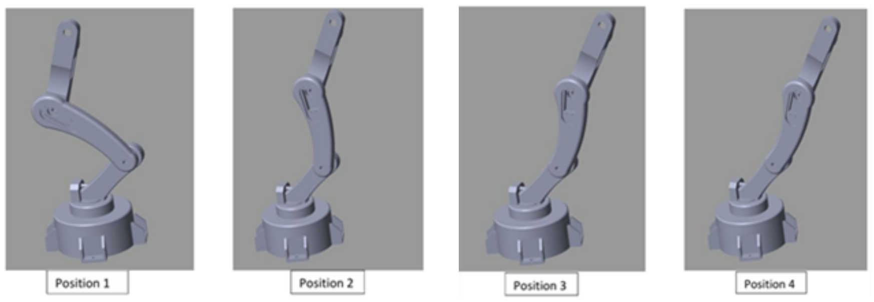
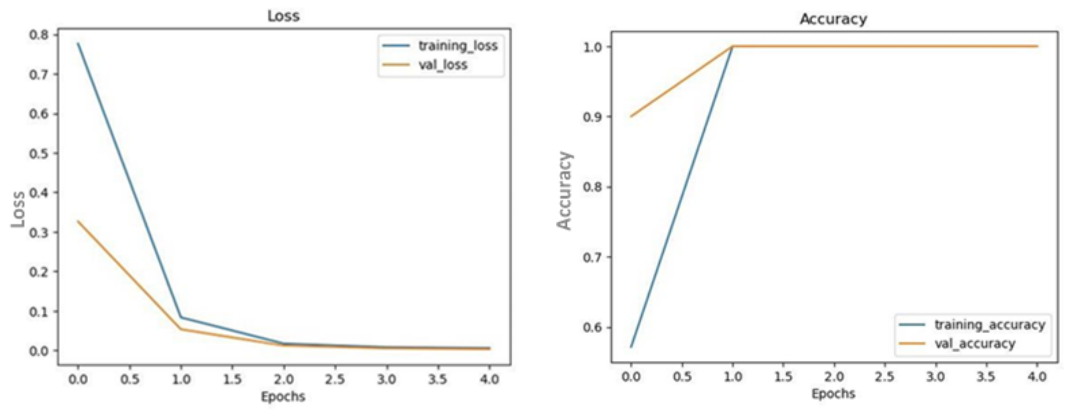


Figure 15. Simulation of the arm in Simulink

4. RESULTS AND DISCUSSION

To evaluate the performance of our machine learning model, we visualized in Figure 16 the results by plotting loss in Figure 16(a) and accuracy in Figure 16(b) curves using a custom function, plot_loss_curves. This function takes as input the history of the trained model, specifically the loss and accuracy values across all epochs, and generates separate plots for the training and validation datasets. In these two graphs, we can see that our machine learning model demonstrated excellent performance, achieving perfect accuracy (1.0) on both the training and validation datasets. This indicates that the model successfully generalized to new data without overfitting, a critical consideration in machine learning. Additionally, our findings align with previous studies that emphasize the importance of monitoring loss and accuracy curves to detect overfitting [27]. However, unlike some studies that report a trade-off between accuracy and computational complexity [28], [29], our model achieved high accuracy without requiring excessive computational resources. This may be attributed to the efficient architecture of the CNN and the high-quality training data used in this study.



(a) (b)

Figure 16. Result of the custom function plot_loss_curves of (a) loss curve and (b) accuracy curve

The performance of our CNN-based object detection system is consistent with recent advancements in deep learning for robotics [30], [31]. For instance, the use of TensorFlow for implementing the CNN aligns with best practices in the field, as highlighted by [32]. However, our approach differs in its integration of real-time ultrasonic sensors for environmental awareness, which enhances the robot's ability to navigate dynamic environments a feature not extensively explored in prior work. Future work should validate these findings on larger and more diverse datasets to ensure generalizability. Second, the 3D-printed Mecanum wheels, while effective, may have durability issues in harsh environments. Testing the wheels under more extreme conditions would provide valuable insights into their long-term performance.

The assembly of the robot involved integrating several critical components, listed in Table 2. The stepper motors (Model: PSM57HS2A54-2P) and servo motors (Models: MG996R and 9G) were essential for driving the Mecanum wheels and operating the KUKA robotic arm, respectively. The camera module (Camera V2 for Raspberry Pi, 8 MP) plays a pivotal role in the robot's vision system, capturing images for object detection.

Table 2. Equipment of the robot

Item	Model number	Quantity
Step motor	PSM57HS2A54-2P	4
Servo motor	MG996R	4
Servo motor micro	9g	3
Camera	Camera V2 for Raspberry Pi 8 MP	1
Obstacle detector	HC-SR04	3
Raspberry	Raspberry Pi 3 board	1
Battery	External battery 20000 mAh PD 3.0/QC 4.0	1

Figure 17 illustrates the 3D-printed Mecanum wheels, which were crucial for achieving the desired mobility. The wheels' ability to move in any direction without changing orientation allows the robot to navigate tight spaces and adjust its position with precision. This capability is key for tasks like object retrieval in cluttered environments. Figure 18 shows the completed robot.

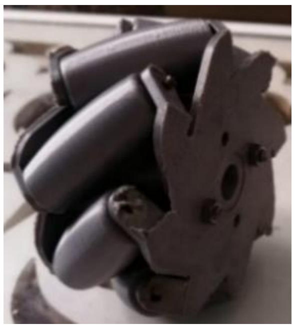


Figure 17. Mecanum wheel printed with a 3D printer

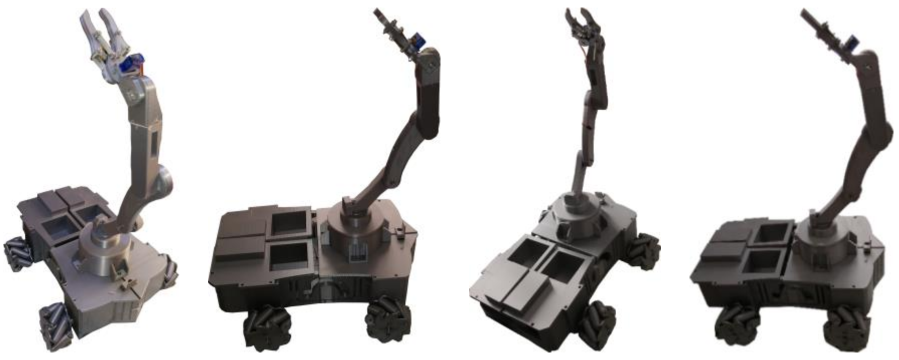


Figure 18. The completed robot

The robot's object detection system is powered by a CNN implemented using TensorFlow. The camera captures images and publishes them on the "image_topic," where the CNN processes the images to detect objects. Upon detecting an object, the system publishes the detection information, including the coordinates of the detected objects, on the "object_detection_topic." The KUKA robotic arm, equipped with six servo motors, uses this detection data to execute the necessary movements to grasp the object. The precision of the CNN in detecting objects and the accuracy of the servo motors in positioning the arm are critical for successful object retrieval. The integration of these systems ensures that the robot can autonomously detect and manipulate objects in real-time.

To enhance the robot's environmental awareness, three ultrasonic sensors (HC-SR04) were mounted on the robot one at the front and two on the sides. These sensors continuously measure the distance between the rover and surrounding objects, publishing this data to the "ultrasonic_sensor_module." The real-time distance measurements allow the robot to avoid collisions and navigate safely in dynamic environments. The integration of these sensors with the robot's control system enables the rover to autonomously make decisions about its path, ensuring safe operation even in complex and unpredictable environments.

5. CONCLUSION

In this study, we developed and validated an autonomous rover system equipped with a KUKA robotic arm, using extensive simulations to assess its performance in navigation and object manipulation tasks. These simulations proved essential for testing the system in a risk-free virtual environment, allowing us to refine the architecture, control strategies, and perception algorithms before transitioning to physical deployment. The results confirmed the feasibility and reliability of our approach, demonstrating the rover's ability to navigate complex terrains autonomously and perform precise object detection, localization, and grasping operations using the robotic arm. This work lays a strong foundation for real-world implementation, where the system can be adapted to various application domains such as search and rescue missions, industrial automation, and planetary exploration. In future work, we aim to integrate advanced learning methods such as reinforcement learning to improve decision-making in dynamic environments, and to conduct real-world experiments to evaluate the system's robustness under diverse operational conditions. Overall, our findings highlight the potential of simulation-driven development in robotics and pave the way toward the deployment of versatile, intelligent, and autonomous robotic platforms.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Aziz El Mrabet	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		
Hicham Hihi		✓		✓		✓		✓		✓		✓	✓	✓
Mohammed Khalil Laghraib			✓		✓	✓				✓				✓
Mbarek Chahboun						✓				✓		✓		✓
Aymane Amalaoui					✓	✓		✓		✓		✓		✓

C : **C**onceptualization	I : **I**nvestigation	Vi : **Vi**sualization
M : **M**ethodology	R : **R**esources	Su : **Su**pervision
So : **So**ftware	D : **D**ata Curation	P : **P**roject administration
Va : **Va**lidation	O : Writing - **O**riginal Draft	Fu : **Fu**nding acquisition
Fo : **Fo**rmal analysis	E : Writing - Review & **E**diting	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available on request from the corresponding author, [AEM]. The data, which contain information that could compromise the privacy of research participants, are not publicly available due to certain restrictions.

REFERENCES

[1] D. Rondao, L. He, and N. Aouf, "AI-based monocular pose estimation for autonomous space refuelling," *Acta Astronautica*, vol. 220, pp. 126–140, 2024, doi: 10.1016/j.actaastro.2024.04.003.
[2] Z. Li *et al.*, "A YOLO-GGCNN based grasping framework for mobile robots in unknown environments," *Expert Systems with Applications*, vol. 225, 2023, doi: 10.1016/j.eswa.2023.119993.
[3] J. Liang, W. Luo, and Y. Qin, "Path planning of multi-axis robotic arm based on improved RRT*," *Computers, Materials and Continua*, vol. 81, no. 1, pp. 1009–1027, 2024, doi: 10.32604/cmc.2024.055883.
[4] D. K. Nkemelu, D. Omeiza, and N. Lubalo, "Deep convolutional neural network for plant seedlings classification," *arXiv-Computer Science*, pp. 1–5, 2018.
[5] M. Hasan, N. Vasker, and M. S. H. Khan, "Real-time sorting of broiler chicken meat with robotic arm: XAI-enhanced deep learning and LIME framework for freshness detection," *Journal of Agriculture and Food Research*, vol. 18, 2024, doi: 10.1016/j.jafr.2024.101372.
[6] M. Hasan, N. Vasker, M. M. Hossain, M. I. Bhuiyan, J. Biswas, and M. R. Ahmmad Rashid, "Framework for fish freshness detection and rotten fish removal in Bangladesh using mask R–CNN method with robotic arm and fisheye analysis," *Journal of Agriculture and Food Research*, vol. 16, 2024, doi: 10.1016/j.jafr.2024.101139.

[7] X. Du *et al.*, "Comprehensive visual information acquisition for tomato picking robot based on multitask convolutional neural network," *Biosystems Engineering*, vol. 238, pp. 51–61, 2024, doi: 10.1016/j.biosystemseng.2023.12.017.

[8] D. Leite, A. Brito, and G. Faccioli, "Advancements and outlooks in utilizing convolutional neural networks for plant disease severity assessment: a comprehensive review," *Smart Agricultural Technology*, vol. 9, 2024, doi: 10.1016/j.atech.2024.100573.

[9] B. Gülmez, "Advancements in maize disease detection: a comprehensive review of convolutional neural networks," *Computers in Biology and Medicine*, vol. 183, 2024, doi: 10.1016/j.compbiomed.2024.109222.

[10] C. Chen, N. A. Mat Isa, and X. Liu, "A review of convolutional neural network based methods for medical image classification," *Computers in Biology and Medicine*, vol. 185, 2025, doi: 10.1016/j.compbiomed.2024.109507.

[11] R. Chundi *et al.*, "Exploring diabetes through the lens of AI and computer vision: methods and future prospects," *Computers in Biology and Medicine*, vol. 185, 2025, doi: 10.1016/j.compbiomed.2024.109537.

[12] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision*. Upper Saddle River, New Jersey: Prentice Hall, 1998.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge, United Kingdom: Cambridge University Press, 2004.

[14] C. Wang, N. Komodakis, H. Ishikawa, O. Veksler, and E. Boros, "Inference and learning of graphical models: theory and applications in computer vision and image analysis," *Computer Vision and Image Understanding*, vol. 143, pp. 52–53, 2016, doi: 10.1016/j.cviu.2016.01.001.

[15] Y. Haruna, S. Qin, A. H. Adama Chukkol, A. A. Yusuf, I. Bello, and A. Lawan, "Exploring the synergies of hybrid convolutional neural network and vision transformer architectures for computer vision: a survey," *Engineering Applications of Artificial Intelligence*, vol. 144, 2025, doi: 10.1016/j.engappai.2025.110057.

[16] Z. Hou, H. Wang, Y. Yue, M. Xiong, and C. Che, "A novel method for cause portrait of aviation unsafe events based on hierarchical multi-task convolutional neural network," *Expert Systems with Applications*, vol. 270, 2025, doi: 10.1016/j.eswa.2025.126466.

[17] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, 2021, doi: 10.1186/s40537-021-00444-8.

[18] R. R. Murphy, *Introduction to AI robotics*. Cambridge, United Kingdom: MIT Press, 2019.

[19] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Berlin, Heidelberg: Springer, 2008.

[20] Y. Uchida, T. Saito, and T. Hatakeyama, "Development of a multi-purpose module system using Mecanum wheel module," *International Journal of Applied Electromagnetics and Mechanics*, vol. 59, no. 3, pp. 967–975, 2019, doi: 10.3233/JAE-171096.

[21] J. B. Song and K. S. Byun, "Design and control of a four-wheeled omnidirectional mobile robot with steerable omnidirectional wheels," *Journal of Robotic Systems*, vol. 21, no. 4, pp. 193–208, 2004, doi: 10.1002/rob.20009.

[22] M. U. Shafiq *et al.*, "Real-time navigation of mecanum wheel-based mobile robot in a dynamic environment," *Heliyon*, vol. 10, no. 5, 2024, doi: 10.1016/j.heliyon.2024.e26829.

[23] R. Carbonell, Á. Cuenca, J. Salt, E. Aranda-Escolástico, and V. Casanova, "Remote path-following control for a holonomic Mecanum-wheeled robot in a resource-efficient networked control system," *ISA Transactions*, vol. 151, pp. 377–390, 2024, doi: 10.1016/j.isatra.2024.05.041.

[24] P. N. Dao and M. H. Phung, "Nonlinear robust integral based actor–critic reinforcement learning control for a perturbed three-wheeled mobile robot with Mecanum wheels," *Computers and Electrical Engineering*, vol. 121, 2025, doi: 10.1016/j.compeleceng.2024.109870.

[25] J. J. Craig, *Introduction to robotics: mechanics and control*. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2005.

[26] S. Zenhari, H.-C. Möhring, and A. V. Torbati, "Comprehensive analysis of kinematic models based on the DH method and screw theory for a five-axis machine tool," *Procedia CIRP*, vol. 130, no. 27, pp. 1745–1751, 2024, doi: 10.1016/j.procir.2024.10.310.

[27] J. Zhao, J. Yin, Y. Shi, L. Qiao, and G. Ma, "User entertainment experience analysis of artificial intelligence entertainment robots based on convolutional neural networks in park plant landscape design," *Entertainment Computing*, vol. 52, 2025, doi: 10.1016/j.entcom.2024.100817.

[28] J. Singh *et al.*, "Real-time convolutional neural networks for emotion and gender classification," *Procedia Computer Science*, vol. 235, pp. 1429–1435, 2024, doi: 10.1016/j.procs.2024.04.134.

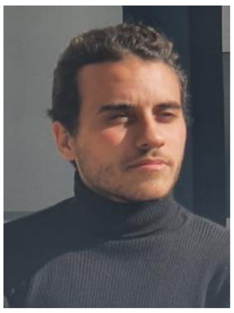
[29] A. A. K. Farizhandi and M. Mamivand, "Processing time, temperature, and initial chemical composition prediction from materials microstructure by deep network for multiple inputs and fused data," *Materials and Design*, vol. 219, 2022, doi: 10.1016/j.matdes.2022.110799.

[30] J. V. Kumar and V. K. Elumalai, "A proximal policy optimization based deep reinforcement learning framework for tracking control of a flexible robotic manipulator," *Results in Engineering*, vol. 25, 2025, doi: 10.1016/j.rineng.2025.104178.

[31] W. Xia, Y. Lu, W. Xu, and X. Xu, "Deep reinforcement learning based proactive dynamic obstacle avoidance for safe human-robot collaboration," *Manufacturing Letters*, vol. 41, pp. 1246–1256, 2024, doi: 10.1016/j.mfglet.2024.09.151.

[32] F. Fan *et al.*, "Spatiotemporal path tracking via deep reinforcement learning of robot for manufacturing internal logistics," *Journal of Manufacturing Systems*, vol. 69, pp. 150–169, 2023, doi: 10.1016/j.jmsy.2023.06.011.

BIOGRAPHIES OF AUTHORS

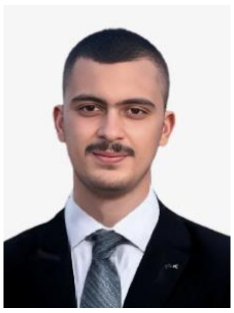


Aziz El Mrabet received the engineering degree in Mechanical Engineering and Automated Systems from Université Sidi Mohamed Ben Abdellah, National School of Applied Science, Fez, in 2022. He is actively pursuing a Ph.D. in the Laboratory of Engineering, Systems, and Applications at the National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco. His research focuses on intelligent system control, robotics, and the use of artificial intelligence. He can be contacted at email: aziz.elmrabet@usmba.ac.ma.

Hicham Hihi is a full Professor at the National School of Applied Sciences and at the Laboratory of Engineering, Systems and Applications in Sidi Mohamed ben Abdellah University of Fez, Morocco. He received the Ph.D. degree in 2008 from the Ecole Centrale Lille (France) in Control Engineering, and the HDR degree in 2016 from the Cadi Ayyad University of Marrakech, Morocco. From 2015 to 2018, he was Director of the Electrical Engineering Department at ENSA in Marrakech at Cadi Ayyad University. Also, he has been the President of the International Conference on Monitoring Industrial Systems since 2011. From 2015 to 2019, he was President of the Association of Research and Industrial Innovation (Rinnovaindus) and vice-president since 2019. His research interests include: i) modeling and simulation of physical systems by using bond graphs, discrete-event and hybrid systems, with mechatronic applications, and ii) the advanced management of energy of electrical systems and vehicles. He is the author and co-author of more than 100 scientific publications (journals, newspapers, and international and national conferences). Since 2020, he has been responsible for the research team: "Renewable energy and control systems", and since 2023, he has been the deputy director of the Laboratory of Engineering, Systems and Applications at the ENSA of Fez. He is also a responsible member of several projects. He can be contacted at email: hicham.hihi@usmba.ac.ma.

Mohammed Khalil Laghraib received the engineering degree in Mechanical Engineering and Automated Systems from Université Sidi Mohamed Ben Abdellah, National School of Applied Science, Fez, Morocco, in 2022, and master 2 EEEA in information processing and instrumentation for engineers from University Jean Monnet Saint-Etienne, France, in 2023. He is actively pursuing a Ph.D. in the Laboratory of Engineering, Systems, and Applications at the National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco. His research focuses on planning and supervision of an autonomous vehicle operating in an uncertain environment. He can be contacted at email: Mohammedkhalil.laghraib@gmail.com.

Mbarek Chahboun obtained his master’s degree in Electronic and Embedded Systems from Université Moulay Ismail, Morocco, in 2020. Currently, he is pursuing his Ph.D. in Electrical and Power Engineering at the Systems and Applications Engineering Laboratory, National School of Applied Sciences at Sidi Mohamed Ben Abdellah University, Fez, Morocco. His research interests include adaptive control, nonlinear control, with applications to power conversion and renewable energy systems, while sharing his expertise through his articles. He can be contacted at email: mbarek.chahboun@usmba.ac.ma.



Aymane Amalaoui received his engineering degree in Mechanical Engineering and Automated Systems from the National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco, in 2023. He is currently pursuing a Ph.D. at the Innovative Technologies Laboratory within the same institution. His research focuses on smart city systems and the application of artificial intelligence in promoting sustainable development in Morocco. He can be contacted at email: aymane.amalaoui@usmba.ac.ma.