

Development of a Digital Twin for Robotic Inspection Using Computer Vision and Machine Learning

P.V.Pramila^{1*}, *R.Jothilakshmi*², *K R Senthil Kumar*³, *B. Revathi*⁴, *A. Kistan*⁵, and *Bhavya Lingampalli*⁶

¹Department of Computer Science Engineering, SIMATS Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, Tamil Nadu 602105, India

²Department of Information Technology, R.M.D Engineering College, Kavaraipettai, Tamil Nadu 601206, India

³Department of Mechanical Engineering, R.M.K. Engineering College, Kavaraipettai, Tamil Nadu 602105, India

⁴Department of Mathematics, Saveetha school of Engineering, Saveetha institute of Medical and Technical Sciences, Saveetha University, Chennai, Tamil Nadu 600077, India

⁵Department of Chemistry, Panimalar Engineering College, Chennai, Tamil Nadu 600123, India

⁶Department of Artificial Intelligence and Robotics, School of Engineering, Dayananda Sagar University, Bengaluru, Karnataka 562112. India

Abstract. This article reveals the creation and execution of a digital twin (DT) for the robotic inspection system of mechanical parts, applying the machine vision method. The examination is carried out on the KUKA iiwa robot that has a camera for the purpose of identifying and sorting the parts that are positioned on the work area. A vision system based on a YOLO convolutional neural network (CNN) was utilized for the tasks of object detection and classification. The dataset was automatically created via the use of 3D CAD models along with domain randomization to mitigate the risk of overfitting. The simulation environment was set up in IsaacSim, and the vision system was interfaced through ROS Noetic with hand-eye calibration also included. The performance of the YOLO network evidenced an overall precision of 94.8% and recall of 87.7% in the test dataset indicating the parts' effective detection and classification. The research underlines the convergence of simulation, deep learning, robotics, and the potential for automated inspections in manufacturing sectors.

1 Introduction

The rapid advancement of manufacturing technologies has increasingly demanded automated solutions for quality control and inspection tasks. Machine vision systems, combined with robotic manipulators, offer precise and efficient methods for detecting and classifying components in industrial settings. However, developing and validating such systems in real environments can be costly and time-consuming. Digital twins (DTs) provide a virtual

* Corresponding author: pramilapv.sse@saveetha.com

representation of physical systems, enabling simulation, testing, and optimization before deployment in real applications [1-6].

Digital twins can be defined as a virtual replica, whether at the asset, process, or system level, which mirrors the real-life object, entity, or process in a dynamic and continuous synchronistic manner. In addition, digital twins are different from simulations in the way they utilize real-time or near real-time data, physics, and data-oriented approaches like machine learning for monitoring, analysis, prediction, and optimization to study systems and entities in varied conditions [3, 6-12].

For example, digital twins are utilized in the manufacturing and robotic fields as cyber-physical systems, which aid in design validation, performance, and decision-making processes that incorporate simulation, sensing, control, and vision feedback [4,5,8,9,13]. Furthermore, the potential and contributions of computer vision, which underpin digital twins, were emphasized in the relevant literature, with respect to environment perception, synchronization, and intelligent inspection in complex scenarios, such as industry [10–14].

This study focuses on creating a DT of a robotic inspection system, wherein a KUKA iiwa robot with an attached camera performs part detection and classification. The vision system relies on a YOLO-based convolutional neural network trained with synthetic images generated from 3D CAD models, enhanced through domain randomization techniques to improve generalization. The simulation environment was developed using IsaacSim, providing realistic kinematics and visual representation of the workspace. Communication and control integration between the robot and vision system were implemented using ROS Noetic. This framework allows for testing inspection strategies, evaluating detection performance, and exploring potential improvements in robotic automation without interfering with actual manufacturing operations.

From this perspective, machine learning should function as the perception and intelligence layer of the digital twin, facilitating automated perception of visual information and enabling interaction between the digital and the robotic inspection.

2 Literature Review

Digital twin (DT) technology has shifted its focus from static virtual twins to dynamic and data-driven virtual twins that facilitate the synchronization, monitoring, and decision-making processes with the aid of real-time technology. Recent studies have shed much light on the significance and importance of DT technology with regards to the implementation of intelligent manufacturing systems, human-centric Industry 4.0/5.0 concepts, and the coordination within multirobot environments [6,7,15].

Object detection, classification, and manufacturing process monitoring have been successfully implemented in manufacturing systems. Rodrigues and Lasthaus discussed the computer vision approaches available for the manufacturing industry's components. Yu et al. proposed an extensive survey on the role and uses of computer vision for assisted additive manufacturing. Computer vision was also considered for sustainability-oriented manufacturing systems and for the development of industrial safety systems known as Safety 4.0.

In recent times, several studies specifically explored the concept of integrating ML models with DT. A DT framework with the support of RTA for flexible manufacturing systems was proposed by Ullah et al. [3]. Kovari proposed a DT framework with the support of vision transformers for the Industry 5.0 concept. In this work, the potential of ML models for DT was explored. O'Donovan et al. [11] highlighted how computer vision can improve the sustainability of steel manufacturing by enabling real-time monitoring and quality assessment, reducing waste, and optimizing resource usage. Similarly, Yousif et al. [12] explored the role of vision-based systems in the context of Safety 4.0, demonstrating how

advanced industrial protection can be achieved through automated detection of hazards and compliance monitoring. In the realm of robotic assembly, Touhid et al. [13] evaluated the synchronization of digital twins using YOLOv8 for real-time object detection, emphasizing the importance of accurate vision-driven feedback for maintaining DT fidelity in dynamic manufacturing environments.

Advanced manufacturing planning and process optimization is another area where digital twins have been applied. Wang et al. [7] reviewed DT-driven multi-robot collaborative manufacturing with challenges in real-time data fusion and scalability, while Li et al. [8] proposed feature level DT for intelligent machining using model-based definition data and Vinci-Carlavan et al. [9] developed DT for operations management in engineering-to-order environment. Doroszuk et al. [14] integrated computer vision with Discrete Element Method and Smoothed Particle Hydrodynamics (DEM-SPH) simulations to calibrate a laboratory-scale digital twin for copper ore milling, demonstrating the applicability of vision-assisted DTs in process-specific optimization. Kozin [15] further discussed the operational management benefits of digital twin technology in car maintenance and repair, illustrating how DTs, when combined with visual inspection and monitoring, can streamline production and service planning.

In general, the literature shows that machine learning, computer vision, and digital twin technologies are increasingly converging. However, most of the existing research either focuses on perception models or virtual system modeling in isolation or is mostly dependent on offline analysis. In this context, the contribution of the current work is an integrated ML-driven digital twin framework, which offers real-time perception, continuous data exchange, and synchronized virtual modeling for intelligent industrial automation.

3 Materials and Methods

This section is dedicated to presenting the architecture of the proposed system and the methods used to implement it. It begins with an overview of the parts inspection problem and its specifications. Following this, the solution architecture is described and its subsystems are detailed, with the main materials and methods employed to develop them. Finally, the integration of the system parts is performed.

3.1 Inspection Problem

A typical application of machine vision systems in manufacturing is the inspection of parts through the detection and classification of these objects in images. The case study in this work is the simulation of a visual inspection routine, which is performed by a fixed robot with a camera attached to its flange. The real system used as a reference for the construction of the DT is shown in Fig. 1. The inspection is performed on a set of mechanical parts that are arranged on a work area. The vision system identifies which parts are on that area and, upon locating a specific part, the robot approaches and acquires a new image of the “target” part.



Fig. 1. Real reference system, KUKA iiwa robot with camera attached to the end effector and parts arranged on a work area.

To characterize the inspection task, five real mechanical parts were selected. This set, shown in Fig. 2a, consists of the following aluminum parts: a U-shaped support parts were also used, which are shown in Fig. 2b and make up the simulation environment.

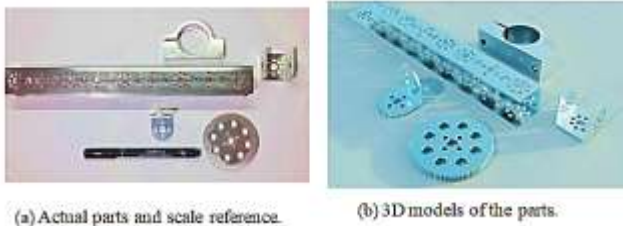


Fig. 2. Set of mechanical parts.

The inspection routine begins when the system control receives a message with the name of the target part. Subsequently, the robot moves to a zero position, where the entire work area is visible to the flange camera. An image is captured and processed by the vision system, obtaining information about which parts are on the work surface, as well as their position in the image plane. With this information, the robot executes approach movements on the target parts and captures new images; this operation is repeated according to the number of "target" parts detected. For example, if the target part is a gear and three gears are detected on the work area, three approaches are performed.

3.2 System Architecture

The architecture of this work is divided into subsystems, so that its development can be carried out in modules. Fig. 3 presents a diagram with this division. All components of the diagram are software applications and have the following functions:

- Simulation: Simulate the robot's kinematics, the environment or "world" in which the robot is inserted, and the camera;
- Vision System: Acquire images of the environment, process them with a machine learning algorithm, and return the positions of the parts. In parallel with the inspection routine, it must also perform the training and validation of the machine learning algorithms.

- **Control:** Defines the sequence of operations executed by the vision system, reads the state of the robot's joints, and publishes new joint states according to the result of the part positions.

Unlike classification-focused, feature extraction, object localization, and classification CNN models, which are typical with classification-focused CNNs like AlexNet or ResNet, the network integrates feature extraction, object localization, or classification into a single network, as with YOLOv5. CSPDarknet53 efficiently supports gradient flow, resulting in low computational complexity, which makes it ideal for robotic inspection.

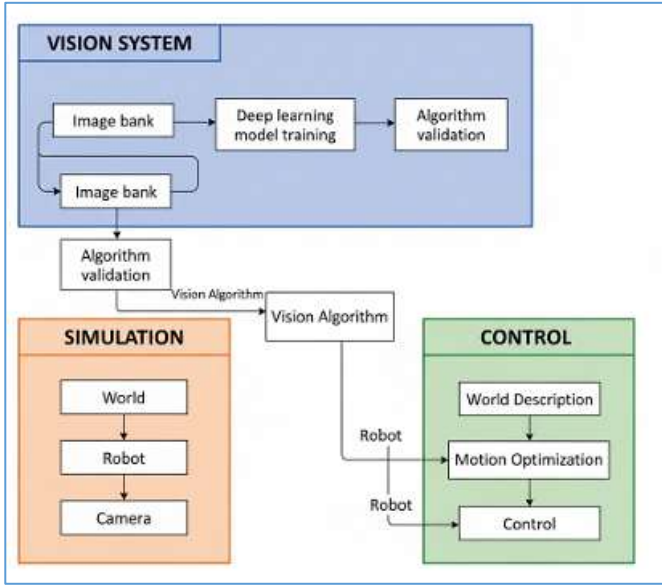


Fig. 3. Diagram of the proposed simulation, with information flows between modules.

3.3 Vision System

The vision system aims to process the images acquired by the camera to obtain descriptions of the environment. These descriptions are used to guide the robot's actions. To perform the image processing, supervised ML techniques with a CNN structure were selected, which have good performance in object classification tasks. The structure of this system is presented in the diagram in Figure 4 and can be summarized in three stages: acquisition and preparation of a database (in this case, images), training of the selected CNN, and validation of the CNN based on performance metrics. The training process is iterative, and according to the results obtained in the validation stage, the input data can be changed (for example, by increasing the number of samples) or the training parameters of the algorithms. At the end of this process, a file containing the internal parameters of the CNN is saved [16-20].

In the context of the present paper's suggested inspection system, the part dedicated to machine learning relies on a convolutional network known as You Only Look Once (YOLO), which deals with object detection and classification simultaneously. Specifically, the YOLOv5s network has been utilized and relies on a convolutional neural network known as CSPDarknet53 that functions with a Path Aggregation Network.

The network was built using the popular deep learning library PyTorch, as it was recognized to be not only flexible but also to have sizable support from developers worldwide, hence best suited for research-oriented development. The training of YOLOv5, as conducted, was directly from the official YOLOv5 repository.

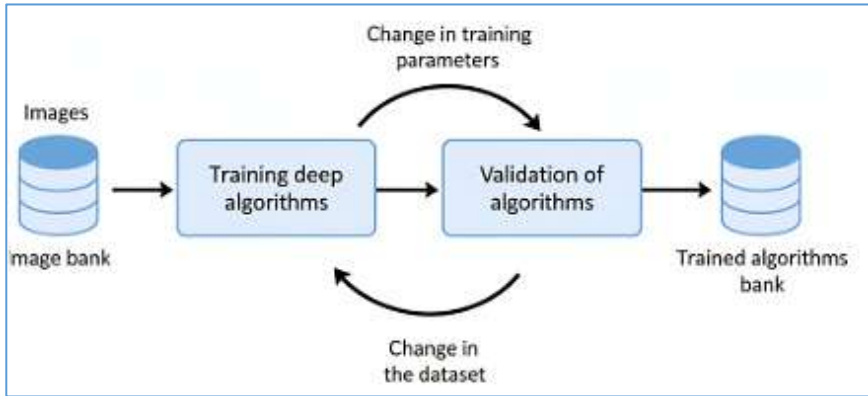


Fig. 4. Vision system diagram.

3.3.1 Image Database

Building a database, in this case an image database, is the first step in applying ML techniques after defining the objective of the work. The images in the dataset are used in the training, validation, and testing stages of the algorithms, and their quality is fundamental to obtaining good results. Since the methods used are supervised ML, in addition to the images, the target information that the algorithm must be able to infer is necessary. In the case of object classification, each image file has an associated label file (usually encoded in text), which contains position information in the form of bounding boxes and classes of the objects present in the image. The bounding boxes are encoded in the center, width, and height of the object in pixels.

The creation of datasets traditionally requires a great deal of manual effort to label each object in the image. In addition, the volume of data required for training DL networks is quite high, easily reaching the order of millions of images with the ImageNet dataset consisting of 1.2 million images. This combination of volume and manual effort is a limitation in the use of DL algorithms. One way to overcome this problem is to build synthetic images in an automated way. Rodrigues (2020) shows the automatic creation of a dataset with images of mechanical parts from the 3D CAD models of the parts [18-23]. In addition, the author performs the training of ML algorithms, obtaining good results in the task of classifying objects in the images.

The image dataset was built automatically from the 3D models of the parts. This is done within the IsaacSim simulation tool, which allows texturing of the parts and application of the domain randomization (DR) method. Tobin et al. (2017) presents this method as a way to create variability in the simulated data. The hypothesis tested by the author is that if the variability in simulation is significant, the ML models trained in simulation will be able to generalize in applications with real data. In the case of the problem of classifying objects in images, this variability consists of the random alteration of texture, position, quantity, and illumination of the simulated objects. Two samples of the dataset generated with this method are presented in Fig. 5, highlighting variations in texture, illumination, and quantity of pieces in the workspace. Approximately 2500 samples were generated to compose the dataset, a quantity considered sufficient based on previous work.



Fig. 5. Samples of the dataset applying DR.

The label information, which encodes the position and class of the objects, is also generated automatically. Fig. 6 exemplifies this information by plotting it on a synthetic image of the workspace. The numbering observed on the boundary of the bounding boxes is the class coding of the objects.

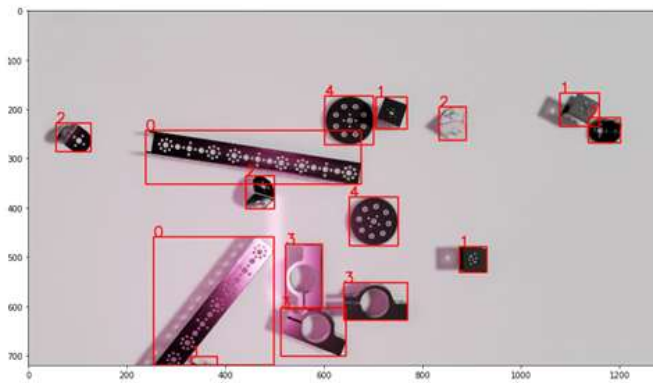


Fig. 6. Example of a sample of the dataset with bounding box labels and class numbering.

3.3.2 Artificial Neural Network Architecture

Since the objective of this work is to develop the DT of a robotic device with a vision system, which acts as an inspection system in manufacturing, processing time was a major factor in the choice.

In short, the selected YOLO algorithm performs two tasks on images: the detection and classification of detected objects in a single network architecture. The strategy for performing these tasks consists of dividing the input image with a grid and, for each part of the division, inferring the object's boundaries in the image and its class. This process is exemplified in Figure 7, where the input image generates a map of bounding boxes that indicate the algorithm's confidence in the existence of an object in the bounded region and a second probability map of the region representing a class of the dataset. Finally, the bounding boxes with the lowest confidence are eliminated, and the predominant classes are applied to the remaining bounding boxes, generating the final detection.

```
import torch
import torch.nn as nn

class MyNeuralNet():
    def init():
        conv1 = nn.Conv2d(1, 6, 5)
        conv2 = nn.Conv2d(1, 6, 5)
        fc1 = nn.Linear(16 * 5 * 5, 120)
        fc2 = nn.Linear(120, 84)
        fc3 = nn.Linear(84, 10)
        #...
```

Fig. 7. Detection and classification strategy of the YOLO network.

In this work, YOLOv5s was chosen considering its appropriate trade-off between inference speed and detection precision. The former is an essential factor for implementing this model, considering it will be used for inspection purposes. The network includes CSPDarknet53 as a backbone for feature extraction, a Path Aggregation Network for feature fusion, and a YOLO network for object detection at different spatial resolutions.

The optimization of object localization is done using Complete Intersection over Union (CIoU) loss, whereas for objectness confidence and classification, Binary Cross Entropy (BCE) loss is being used. This loss function formulation is able to simultaneously optimize object presence, bounding boxes, and classification.

The YOLOv5 network was trained with the PyTorch 1.13 framework. The framework was accelerated using CUDA 11.7. The network was trained using an SGD optimizer with a momentum of 0.937 and weight decay of 0.0005. The batch size was set to 16 images. The images are resized to 640×640 pixels.

In our case, the training was performed for 50 epochs with an initial learning rate of 0.01, which decayed with a cosine function. In addition, all the inherent principles of data augmentation, which are incorporated in YOLOV5, such as random scaling, flip, HSV, and mosaic, were enabled.

3.3.3 Deep Learning Library

DL libraries are abstractions of mathematical functions in a programming language, usually Python and C++. Their goal is to make the construction of deep learning models more flexible in research and commercial applications. Currently, two libraries stand out: PyTorch and TensorFlow, maintained by Facebook and Google, respectively. In this work, PyTorch is used for application development. In terms of performance, they are equivalent, and the choice was based on PyTorch's syntax.

One of the advantages explored in the use of these libraries is the rapid construction of artificial neural network structures through the chaining of small functions. The example in Figure 8 illustrates this process, where a CNN structure, is implemented by the MyNeuralNetwork class in the Python programming language. The basic functions are defined in `init()` by two convolutions (`conv1` and `conv2`) and three fully connected layers (`fc1`, `fc2`, and `fc3`).

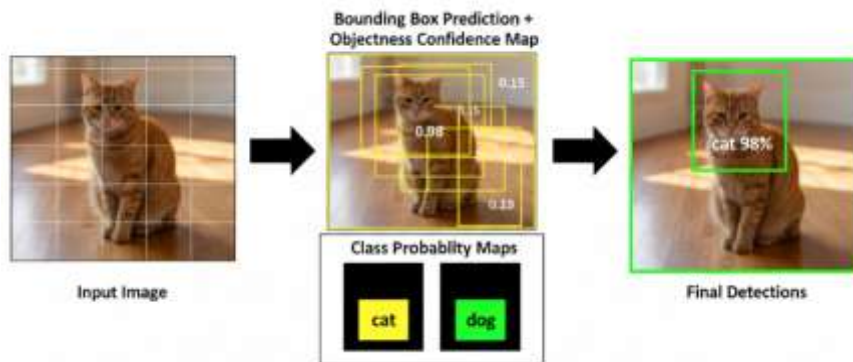


Fig. 8. Example of using the PyTorch library in the Python programming language.

3.3.4 Network Training

The YOLO network training phase begins with dividing the dataset into two parts: a training part and a validation part. A ratio of 80% - 20% is common in the literature, with the majority of images used for training. The validation part is used during training to evaluate the network's performance on accuracy metrics and adjust its internal parameters. Additionally, a test dataset is used after training to quantify the network's performance on new images.

Besides dividing the dataset, it is necessary to choose the training hyperparameters. These parameters are not the network's internal parameters, such as the w weights. The main hyperparameters are learning rate and epochs, which define, respectively, the size of the minimization step and the number of iterations. Each epoch represents a complete pass of all training samples through the network. Learning rates of 0.01 and 50 epochs were used for training. The main hyperparameters and data amounts used in training are summarized below:

- Epochs: 50;
- Learning rate: 0.01;
- Training dataset: 1812 images;
- Validation dataset: 773 images;
- Test dataset: 500 images.

3.3.5 Training Validation

The performance of the trained YOLO network is evaluated on a test dataset with 500 images not used during training. This evaluation is divided into the network's two tasks: object detection and classification. For detection, a confusion matrix is generated to quantify precision and recall. For classification, a confusion matrix with size $n \times n$ is constructed, where n is the number of classes, and different parts are used in training.

The precision and recall metrics represent, in this context of part detection, respectively, the number of correct detections out of all those performed and the number of correct detections out of all possible ones. This problem generates the binary matrix in Figure 9, where the algorithm can infer whether or not a part exists in a region of the image. Therefore, the result of the detection can be one of the four options in the matrix: VP True Positive, FP False Positive, FN False Negative, and VN True Negative. It is important to note that in this part detection problem, there is no labeling for regions without parts, so the VN result will not be obtained, remaining null in the matrix.

		Labeling	
		Part	without part
Inference	Part	TP	FP
	without part	FN	TN

Fig. 9. Confusion matrix for part detection.

For the classification task, a 5×5 confusion matrix is constructed, with the five types of parts described as before. The model of this matrix is illustrated in Figure 10. The main diagonal represents the quantity of True Positives, the columns quantify the False Negatives, and the rows the False Positives.

		Labeling				
		Profile	U Support	L Support	Bearing	Gear
Classification	Profile	TP				
	U Support					
	L Support			...		
	Bearing					
	Gear					TP

Fig. 10. Confusion matrix for classification.

3.4 Simulation

The simulation module aims to build a design test (DT) of the robot and its work environment, and also to enable integration with the vision system. The simulation was developed using IsaacSim software, whose purpose is to serve as a platform for developing robotics applications. Furthermore, this software allows for physics simulation, such as collisions, and photorealistic visualization of the 3D models that make up the simulation, an important factor in this context of image processing.

3.4.1 3D Models

One of the main components of a DT is digital representations of real devices through CAD models. Within the context of this work, the models used are 3D and have the function of generating a graphical representation of the problem under study through the composition of a scene, illustrated in Fig. 11. The elements that make up this scene are: robot, workspace, and parts for detection. There are also components without a 3D structure in the scene, such as lighting, camera, and textures. All simulation components are organized in USD (Universal Scene Description) format.

The 3D models of the parts are obtained from free online repositories, the source files for the KUKA iiwa robot model are from the ROS (Robot Operating System) Industrial repository and also allow free use and modification.

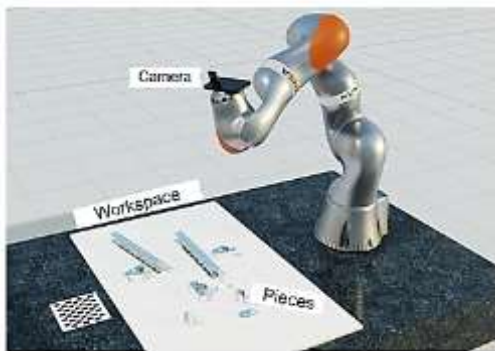


Fig. 11. Composite scene with 3D models of the robot, parts, camera, and workspace.

3.4.2 Robot movement in simulation

The kinematics of the KUKA iiwa study robot were described using the DH parameters. The geometry results in relatively simple parameters indicated in Table 1 below. Since all joint axes can be aligned simultaneously with the Z^{\wedge} axis of the base, the a_i parameter column is null. The parameters were assigned with the robot fully extended, and additionally, the joint rotation limits were added to the table.

Table 1. DH parameters for the KUKA iiwa study robot, plus joint rotation limits.

i	α_i	d_i	a_i	θ_i	Limits (degrees)
1	$-\pi/2$	d02	0	θ_1	$\pm 170^\circ$
2	$\pi/2$	0	0	θ_2	$\pm 120^\circ$
3	$\pi/2$	d24	0	θ_3	$\pm 170^\circ$
4	$-\pi/2$	0	0	θ_4	$\pm 120^\circ$
5	$-\pi/2$	d46	0	θ_5	$\pm 170^\circ$
6	$\pi/2$	0	0	θ_6	$\pm 120^\circ$
7	0	d67	0	θ_7	$\pm 175^\circ$

The distances between the coordinate systems d02, d24, d46, and d67 are, respectively, 340 mm, 400 mm, 400 mm, and 126 mm, according to the manufacturer's website.

Within the simulation environment, the robot is described by a URDF (Unified Robot Description Format) file, which contains kinematic, dynamic, 3D model, and collision model information. The URDF description of the KUKA iiwa was implemented based on the ROS Industrial repository (HOORN, 2021), and modifications were made to ensure the description matches the one performed using the DH convention.

Compared to the description made via DH, there were coordinate systems rotated in Z' that were corrected. In this work, only the position of the simulated robot's end effector is analyzed, based on the workspace description provided by the vision system. Manipulator velocities and accelerations are not analyzed. The movement to the pose indicated by the vision system result is performed by transforming the 2D coordinate system of the image into the 3D system of the robot's base. Since there is no depth information with a monocular camera, a working height z was fixed for the camera. Knowing the camera's aperture angle, this transformation between the image plane and the workspace plane was also performed.

3.5 Vision-Robot Control and Integration

The integration between the vision system and the simulated robot is done with the ROS software. ROS is a set of free software tools and libraries dedicated to the development of robotics applications. In this work, it is used for data communication between the simulation and the vision system. This task is facilitated by ROS through a standardization of messages and a publisher & subscriber communication structure between nodes and topics. A node, in the context of ROS, is an executable program responsible for one or more tasks. A topic can be interpreted as a memory space where nodes can write or read messages with standardized formats.

A simplified computation diagram with the system's nodes and topics is presented in Fig. 12, where nodes are represented by ellipses and topics by rectangles. The simulation tool used, IsaacSim, allows communication with ROS through a specific node named /OmniIsaacRosBridge. An /inspection_routine node, external to the simulation, was built to control the inspection task. Communication between these two nodes is done by exchanging messages in the topics. There are two main messages in the communication: the robot's joint status and the RGB camera image.

The inspection node is triggered by a third message with the target part type, that is, which type of part should be approached and a detailed image acquired. Subsequently, this node reads the messages from the /rgb topic, which contains the simulation camera image, and performs an inference with the previously trained YOLO network. The result of this inference returns an estimate of the positions (bounding boxes) of the parts and their classes.

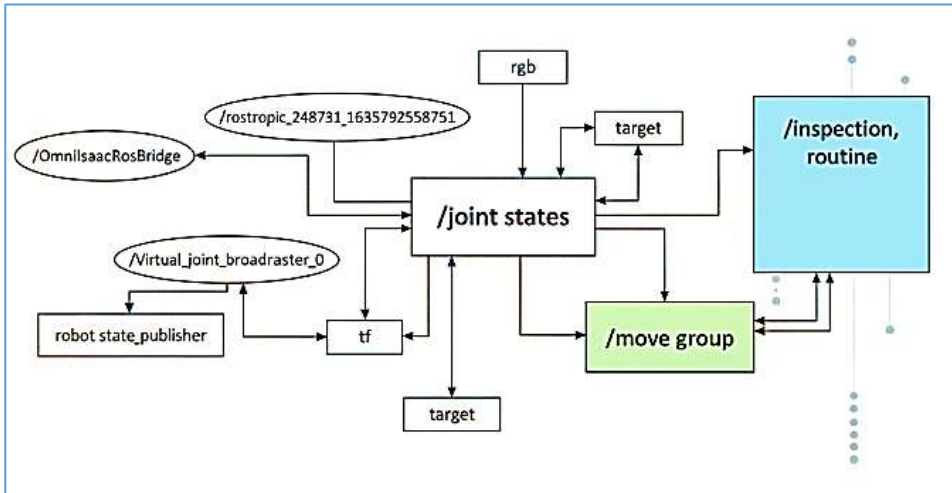


Fig. 12. Simplified computational graph with main nodes and topics of the system.

It is worth noting that the ROS version used in this work is Noetic. The entire simulation environment and the YOLO network are run on a Linux machine with Ubuntu 20.04, Intel 10700 processor, 32 GB of RAM and NVIDIA RTX 3060 graphics card.

In this work, iterative methods of the MoveIt! framework are used to solve the inverse kinematics of the robot and generate its trajectory. MoveIt! is an open-source software tool with motion planning, object manipulation, inverse kinematics, and collision checking functionalities.

3.5.1 Handeye Calibration

The Hand-Eye calibration process was performed in simulation by positioning a frame of fixed ArUco markers, as illustrated in Figure 13. The markers are detected in the simulated camera image using computer vision algorithms, and the pose of the frame, or target, is estimated. The robot is moved to 15 different poses, and the poses of the target, camera, and end effector are recorded, forming the $AX = BX$ problem. The solution to this problem is computed numerically using a standard calibration method and the MoveIt! tool in the ROS software.



Fig. 13. Scene used for Handeye calibration with ArUco target.

3.6 Integration of Digital Twin and Machine Learning

In the above suggested framework, the concept of machine learning (ML) represents a fundamental function of the digital twin. In this respect, the DT may essentially comprise the following components of the simulation of the Robotic System for Visual Inspection: the simulation of the robotic system, the simulation of the environment, the simulation of the workspace, the ML function. In the simulation of the Robotic System for Visual Inspection, there are components of the robot's kinematics and the camera. In addition, there are the components of the environment and the mechanical parts. On the other hand, the ML model, which utilizes a convolutional neural network based on the YOLO approach, acts as a perception layer for the digital twin, converting the uninterpreted visual images into more structured knowledge that allows the digital twin to update its virtual world view and inform the actions of the robots within the simulation environment.

However, the data flow within the proposed framework has been set up in a bidirectional fashion. On one end, the generated visual information as per the simulated camera and, in the end, the real camera will be sent to the ML module for making appropriate inferences. However, the inferred information will be used again to update the digital twin and make control decisions, thus keeping the digital twin up to date about the prevailing state within the inspection environment.

This, however, does not apply to the current research, as it primarily deals with the implementation of an appropriate task for the operation of the digital twin. Nevertheless, it ought to be mentioned that such an architectural design inherently supports the implementation of feedback and predictive characteristics. The dynamic nature of gathered results from the implemented inspection task can eventually permit the recognition of trends in defective components, non-standard distributions of parts, as well as non-standard system operation.

4 Results and Discussion

This section presents the results of the YOLO network in the detection and classification tasks and its integration into the simulation environment. A first analysis is performed on the test dataset and then on different simulation scenes. Finally, a brief analysis with real images is presented and discussed.

4.1 Object Detection and Classification

During the training phase, the progress of minimizing the cost function, or loss, was also analyzed. The values of this function throughout the epoch iterations are the error sums for each set of training and validation images. The reduction of this value, and therefore of the error, indicates an improvement in network performance. Fig. 14 shows the training and validation loss during 50 epochs. It is clear that there is a lower error reduction rate after the twentieth epoch, which is also evident in the graph in Figure 15 with the precision and recall metrics. This behavior implies that training could be completed in at least half the epochs and in half the time. The total training time for 50 epochs was 35 minutes with an NVIDIA RTX 3060 graphics card.

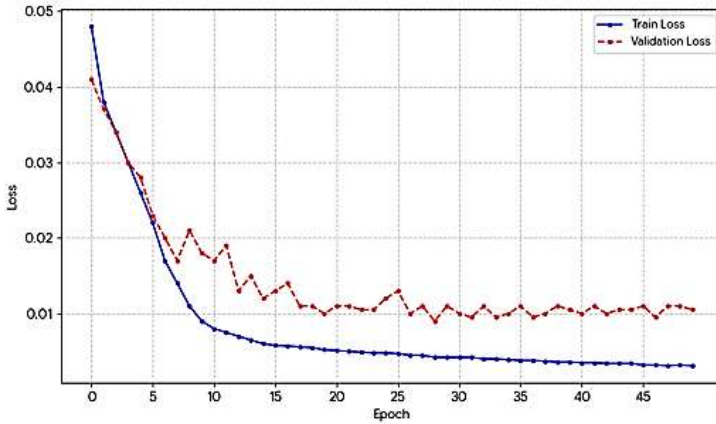


Fig. 14. Minimization of training and validation cost functions.

The precision and recall metrics throughout the training process are presented in Fig. 15. As observed in the loss graph, performance stabilizes after 20 epochs, precision reaches values above 90%, while recall remains between 80% and 85%.

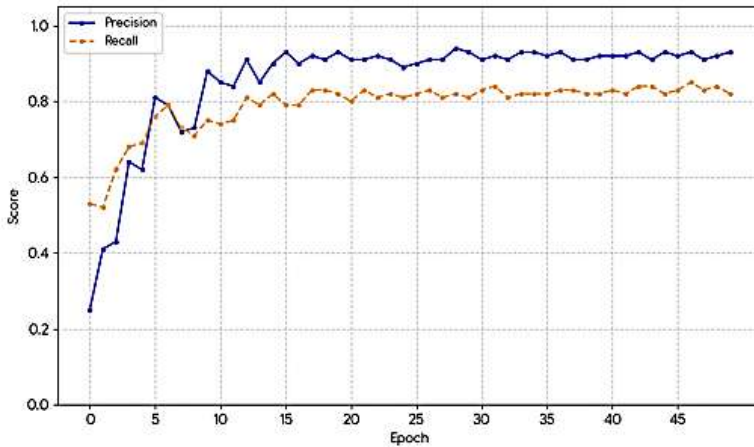


Fig. 15. Evolution of Precision and Recall metrics during training.

The two detection and classification tasks were evaluated on a test dataset containing 500 synthetic images, and the confusion matrix in Figure 16 was generated with the five classes present in the dataset. It is worth noting that the classes are not balanced in the dataset, that is, there is a different number of samples for each piece. To facilitate the visualization of the results in the confusion matrix, rates were used instead of absolute values. An extra dimension was also added to the matrix to represent False Positive and False Negative detections. Table 2 presents the metrics evaluated by object class. Overall, the network performed well on the test dataset, with 94.8% precision and 87.7% recall.

Table 2. Summary of YOLO network classification results.

Class	Images	Labels	Precision	Recall
Profile	500	939	0.966	0.950
Upper Support	500	1396	0.951	0.764
Lower Support	500	1724	0.881	0.852
Bearing	500	1317	0.952	0.920
Gear	500	821	0.991	0.901
Overall	500	6197	0.948	0.877

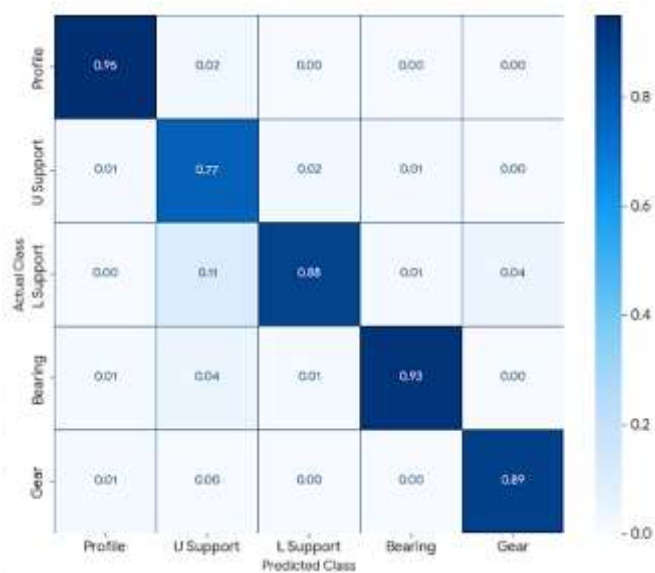


Fig. 16. Normalized confusion matrix for multi-class part classification using the YOLO network. Rows indicate ground-truth classes and columns indicate predicted classes. Values are expressed as percentages to improve readability and comparison across classes.

A clearer interpretation of the detection and classification results may be made by enhancing confusion matrix Fig. 16. As shown in the revised confusion matrix presented here, class names have been included. The rows have been marked as ground-truth classes, while columns indicate classes that have been predicted through a YOLO network.

For the object detection task, the confusion matrix describes the detection or non-detection of parts within regions of interest. In other words, True Positives (TP) represent the correctly identified parts, False Positives (FP) denote incorrectly identified parts where actually no parts are present, while False Negatives (FN) classify incorrectly identified parts. In addition, because background regions are not identified, the values for True Negatives (TN) are not used.

To classify each object in the image and associate it with its corresponding class within these five categories of mechanical parts, we use a multi-class confusion matrix. This measure will prove effective for testing the network's capacity for accurate discrimination among these

five categories. In terms of Fig. 16 and the five types of mechanical parts, we see relatively higher diagonal dominance, thus proving and validating our claim that this network is capable of accurate differentiation among these five categories.

4.2 Simulation-Based Validation and Real-World Transferability

Initially, the proposed inspection framework's validity was established through a simulation environment that is of a highly realistic nature. This is because the primary focus of digital twin technology is to offer a means of quick development, testing, and optimization of robot technology prior to implementation, so that there is less chance of monetary loss and machinery haltage.

Although large scale experiments were not carried out in real-world scenarios for this specific study, preliminary qualitative evaluation for such scenarios considering actual images of different mechanical parts was made. The YOLO network showed promising behavior for geometrically similar objects, and hence, it is confirmed that domain randomization approach is successfully implemented.

Comprehensive validation in a real-world scenario would be conducted in future work, mainly through hardware in loop, depth sensor integration, and extensive quantitative benchmarking. Such features are anticipated to augment realization of enhancing applicability of the proposed digital twin framework.

5 Conclusions

This work successfully developed a digital twin of a robotic inspection system capable of detecting and classifying mechanical parts in a simulated environment. The YOLO network, trained on a synthetic dataset with domain randomization, demonstrated high precision (94.8%) and recall (87.7%) on the test images, validating the effectiveness of the approach. The integration of IsaacSim and ROS Noetic enabled seamless communication between the robot and the vision system, supporting tasks such as hand-eye calibration and target acquisition. The results indicate that the proposed framework can significantly reduce the time and cost associated with developing robotic inspection systems, offering a reliable platform for training, testing, and validating machine vision algorithms before deployment in real industrial environments. Future work may explore the inclusion of depth sensing, dynamic part placement, and real-world transfer learning to further improve system performance.

Even though this study is focused on simulation-based evaluation, the incorporation of synthetic to real generalization analysis, along with some preliminary evaluation on real images, suggests that the possibility exists to deploy these trained models on actual inspection devices as well.

References

1. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world. In Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 23–30 (2017).
2. N. Subramanyam, C.V. Kumar, A.R. Reddy, P. Chandrakanth, Image diagnosis using CNN deep learning model. In: AIP Conference Proceedings, vol. 2802, p. 120005. AIP Publishing LLC (2024).
3. A. Ullah, M. Younas, M. S. Saharudin, Digital twin framework using real-time asset tracking for smart flexible manufacturing system. *Machines* 13, 37 (2025).

4. A. J. Kovari, A framework for integrating vision transformers with digital twins in Industry 5.0 context. *Machines* **13**, 36 (2025).
5. Ö. B. Çapunaman, P. Farrokhsiar, S. G. Bilén, J. P. Duarte, B. Gürsoy, Vision-based sensing and digital twin technologies in conformal 3D concrete printing. *Constr. Robot.* **9**, 4 (2025).
6. T. R. Saravanan, S. Suvitha, D. Banavath, S. Gogula, J. Upadhyay, M. Sudhakar, AI and machine learning integration in medical assistive robotics. In: *Fostering Cross-Industry Sustainability with Intelligent Technologies*; IGI Global, pp. 130–151 (2023).
7. G. Wang, C. Zhang, S. Liu, Y. Zhao, Y. Zhang, L. Wang, Multi-robot collaborative manufacturing driven by digital twins: Advancements, challenges, and future directions. *J. Manuf. Syst.* **82**, 333–361 (2025).
8. J. Li, G. Zhou, C. Zhang, J. Hu, F. Chang, A. Matta, Defining a feature-level digital twin process model by extracting machining features from MBD models for intelligent process planning. *J. Intell. Manuf.* **36**, 3227–3248 (2025).
9. G. Vinci-Carlavan, D. Rossit, A. Toncovich, A digital twin for operations management in manufacturing engineering-to-order environments. *J. Manuf. Syst.* **42**, 100679 (2024).
10. H. Z. Yu, W. Li, D. Li, L. J. Wang, Y. Wang, Enhancing additive manufacturing with computer vision: A comprehensive review. *Int. J. Adv. Manuf. Technol.* **132**, 5211–5229 (2024).
11. C. O'Donovan, C. Giannetti, C. J. Pleydell-Pearce, Revolutionising the sustainability of steel manufacturing using computer vision. *Procedia Comput. Sci.* **232**, 1729–1738 (2024).
12. I. Yousif, J. Samaha, J. Ryu, R. Harik, Safety 4.0: Harnessing computer vision for advanced industrial protection. *Manuf. Lett.* **41**, 1342–1356 (2024).
13. M. T. B. Touhid, E. Zhu, M. V. Ehteshamfara, S. Yang, Evaluation of digital twin synchronization in robotic assembly using YOLOv8. *Int. J. Adv. Manuf. Technol.* **134**, 871–885 (2024).
14. B. Doroszuk, P. Bortnowski, M. Ozdoba, R. Król, Calibrating the digital twin of a laboratory ball mill for copper ore milling: Integrating computer vision and discrete element method and smoothed particle hydrodynamics simulations. *Minerals* **14**, 407 (2024).
15. E. Kozin, Operational management of production for car maintenance and repair using digital twin technology. In *The Future of Industry: Human-Centric Approaches in Digital Transformation* (Springer, Cham, 2024), pp. 205–218.
16. K. B. Prakash, A. Amarkarthik, M. Ravikumar, P. Manoj Kumar, S. Jegadheeswaran, Optimizing performance characteristics of blower for combustion process using Taguchi based grey relational analysis. In: *International Conference on Advances in Materials Research*, pp. 155–163 (2019).
17. M. Balaji, S. N. Dinesh, S. V. Vetrivel, P. M. Kumar, R. Subbiah, Augmenting agility in production flow through ANP. *Mater. Today Proc.* **47**, 5308–5312 (2021).
18. T. Ravichandra, P. Murugeswari, M. Revathi, S. Senthilkumar, D. S. Rathore, M. Sudhakar, Future of machine learning and robotics in digital technology for hospitality. In: *Cutting-Edge Technologies for Business Sectors*; IGI Global, pp. 401–428 (2023).
19. N. Senthil Kannan, R. Parameshwaran, P. T. Saravanakumar, P. M. Kumar, M. L. Rinawa, Performance and quality improvement in a foundry industry using fuzzy MCDM and lean methods. *Arab. J. Sci. Eng.* **47**, 15379–15390 (2022).

20. P. Chandra Kanth, M.S. Anbarasi, A generic framework for data analysis in privacy-preserving data mining. In: Computational Intelligence in Data Mining, pp. 653–661. Springer Singapore, Singapore (2019).
21. R. Chitharaj, H. Perumal, M. Almeshaal, P. Manoj Kumar, Optimizing performance of a solar flat plate collector using Box–Behnken design. *Sustainability* 17, 461 (2025).
22. B. A. Kumar, R. Saminathan, M. Tharwan, M. Vigneshwaran, P. S. Babu, S. Ram, P. M. Kumar, Study on the mechanical properties of a hybrid polymer composite using egg shell powder based bio-filler. *Mater. Today Proc.* 69, 679–683 (2022).
23. K. R. Kunduru, Y. D. Dwivedi, R. Aruna, G. R. Thippeswamy, S. Selvakumar, M. Sudhakar, Elevating performance for enhancing AI-powered humanoid robots through innovation. In: Applied AI and Humanoid Robotics for the Ultra-Smart Cyberspace; IGI Global, pp. 85–119 (2023).