Received: 22 May 2024 | Revised: 26 May 2025 | Accepted: 24 September 2025 | Published online: 18 December 2025

RESEARCH ARTICLE

Artificial Intelligence and Applications
2025, Vol. 00(00) 1-15
DOI: 10.47852/bonviewAIA52023472

BON VIEW PUBLISHING

Deep Learning System for Object Detection and Collision Free Handling in Industrial Robots

Ashish Chouhan[1], Shivam Shandilya[2], Pravanjan Nayak[3], Debasish Mishra[4], and Surjya Kanta Pal[3,*]

[1] *Department of Electronics and Communication Engineering, Birla Institute of Technology, India*

[2] *Department of Electrical and Electronics Engineering, Birla Institute of Technology, India*

[3] *Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, India*

[4] *Department of Data Science and Information Systems, IFMR GSB, Krea University, India*

Abstract: This article presents a deep learning system designed to detect objects on a shop floor, classify them, estimate their pose, and direct a robot to pick them up without collision in real time. This system consists of specific algorithms devised for each of these tasks and has been implemented and tested in an industrial robot. Data collection involved capturing frames of several texture-less cuboid-shaped objects using a low-cost web camera. A convolutional neural network analyzes these frames to detect the object, which is then passed to the image processing module for pose estimation, followed by a sequence scheduling algorithm to determine its grasping. The system achieves 95% accuracy for object detection and 100% for object grasping. The practicality of the solution is confirmed through testing under various conditions such as fluctuating lighting, different colors, undefined orientation, and non-uniform spacing between the objects. The explainability of the model is demonstrated by successfully testing it in these situations. Results show the solution is agnostic to object orientation and background. FurtherFurthermore, it is computationally efficient, accurate, and cost-effective, as it utilizes a webcam, minimizing the overall cost. Additionally, the efficacy of the object detection model is found to be superior when compared to state-of-the-art algorithms.

Keywords: robot, recognition, classification, grasping, pose estimation, collision-free

1. Introduction

Industrial robot manipulators are widely used for autonomous pick-and-place activities in factories and other settings [1, 2]. Most of these robots are hard-coded, i.e., they lack intelligence which does not allow them to adapt to any change during tasks. On the contrary, humans are good at adaptation because of their reasoning abilities. Therefore, robots are equipped with vision-guided devices helping them to perceive the environment [3]. In this context, the importance lies in processing the sensed information to decipher meaningful details. This leads to the intelligence of robots in the intended tasks. This article presents one such artificially intelligent (AI) solution that employs a low-cost vision sensor whose information is processed in real-time with the developed algorithm, leading to detection, classification, pose estimation, and feedback to a robot to grasp an object from the shop floor. The following presents a background of the activities involved in the solution alongside the existing solutions.

To pick up an object, the robot manipulator needs to identify the correct location of the object. This is done by using suitable object detection algorithms, where the main objective is to identify the precise location of the object of interest. Object detection can be done using traditional methods or by using deep learning (DL) algorithms. Yunardi et al. [4] applied a traditional method of using HSV color space to segment out the boxes present on the conveyor belt. The boxes were successfully detected by using a Laplacian filter which determined the largest contour. However, this detection solution may be susceptible to illumination changes because of the HSV color space, and therefore, it may not be apt for an industrial environment. DeGol et al. [5] proposed a colored marker named ChromaTag that rejects initial false detections. It worked faster than other tags with higher detection accuracy. The tag is beneficial because it can be attached to warehouse boxes to detect and localize. However, implementing this method on a large scale is difficult. Jakubovic and Yang [6] performed feature matching between a primary image referred to as the source and a secondary image that was referred to as the target. The following four state-of-the-art algorithms were explored in this work for comparison: Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Binary Robust Invariant Scalable Keypoints (BRISK), and Oriented fast and Rotated Brief (ORB). Among these, SURF was found to be more robust and accurate than others in extracting features to detect the correct box.

Nonetheless, these feature extraction methods fail to work with a texture-less object. This problem can be dealt with by applying DL algorithms that can achieve higher accuracy than traditional object detection algorithms. Zhao et al. [7] applied the convolutional neural network (CNN) algorithm to detect objects stacked on a shop floor and found the CNN model to miss targets when the objects were stacked non-uniformly, objects with interference on their surface, and objects blocking the view of another object. To deal with these challenges, the DL model was augmented with two additional attributes. The first

***Corresponding author:** Surjya Kanta Pal, Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, India. Email: skpal@mech.iitkgp.ac.in

attribute detected the edges of the object by using the Sobel filter to learn the edge features. The second attribute aligned the features of the objects during pooling. Although the faster-RCNN model with these two attributes outperformed the previous models in terms of recall, precision, and average precision, the frame rate was still lower due to the two-stage architecture of the model when compared with single-stage detectors like You Only Look Once (YOLO).

YOLO is a single-stage object detection algorithm that employs a neural network to divide an object into regions, and subsequently predict bounding boxes and probabilities for each region. In contrast to region-based CNNs, like RCNN, YOLO weights bounding boxes by the predicted probabilities, facilitating faster processing with a single neural network. This efficiency in processing makes YOLO faster than RCNN, which utilizes multiple networks. The faster processing speed makes YOLO suitable for applications in the manufacturing industry. Kozamernik et al. [8] developed a DL-based system that aims to ensure the safety of humans and robots sharing a workspace. The system employs the YOLOv3 model, which analyzes images of the workspace to detect the presence of humans within the robot workspace. The system provides feedback to the robot based on this detection to ensure a safe working environment. He et al. [9] developed a DL-based system, named "Smart Hand" to aid humans in performing pick-and-place operations. The system comprised an RGB-D camera, a motion sensor, and an accelerometer that were used to detect objects and track movements. The detection mechanism employed the YOLOv3 model, which enabled the system to provide suggestions to humans on the selection of target objects and to alert them if they chose or placed an object in the wrong location. The smart systems developed in Kozamernik et al. [8] and He et al. [9] can be leveraged to create human-assisting systems that enhance efficiency and accuracy in complex tasks.

Recent advancements in algorithmic research have shown that the YOLOv4 model achieves greater detection accuracy than its predecessor, YOLOv3. Although both models share a similar fundamental architecture, the key differentiating factor is the feature extraction method. Gao et al. [10] utilized the YOLOv4 model to detect and locate objects moving on the conveyor belt. They also utilized the particle filter algorithm to refine the trajectory of the moving objects. By integrating these two technologies, they were able to improve the accuracy and efficiency of object detection and tracking, thereby demonstrating the potential of such approaches in automating manufacturing activities. Wong et al. [11] employed the YOLOv4 model and the rapidly exploring random tree (RRT) algorithm to detect and locate objects, plan the optimal path, and pick up objects within a constrained environment. To enhance the reliability of DL-based systems in industrial pick-and-place operations, Kijdech and Vongbunyong [12] explored the use of the YOLOv5 model to detect and localize objects on the shop floor. The study specifically focused on detecting transparent bottles with three different colored labels, using images captured from an RGB-D camera. Although the research showcases the potential of the YOLOv5 model, limitations still exist from an industrial application perspective with respect to lighting conditions, defects on the bottle, and its pose. Yamtuan et al. [13] utilized the YOLOv5 model to detect objects for controlling a robot. However, the study did not account for lighting conditions during model development. Another study developed a YOLO-GGCNN model to detect and localize objects. In this case, the detection was carried out using the YOLOv3, which was trained with images captured using an RGB-D camera. Additionally, researchers have explored other algorithms for object detection and localization. For instance, Yue et al. [14] developed a dual-channel Siamese framework to detect objects using template-matching criteria. This study used a ResNet50 model to extract two sets of features, each from the RGB-D and depth data. Miao et al. [15] proposed the Improved Nanodet model to detect objects, which combines the power of convolution target detection network, YOLO, and Shufflenet networks.

Apart from object detection applications, YOLO models have also been explored for their ability to detect irregularities or anomalies in various industrial applications. For example, Li et al. [16] developed a method to detect surface defects on a steel strip using the YOLO model. This detection method was later improved by Xu et al. [17] through integration with the K-Means++ algorithm. This integration enabled the selection of the most optimal anchor box, which improved the accuracy of positioning. These studies demonstrate the versatility of YOLO models in detecting anomalies and irregularities in industrial settings, highlighting their potential to enhance quality control.

Post detection and localization, it is essential to determine the best grasping pose for an object. It is decided based on object structure and robotic gripper to be used for grasping. With developments in DL, a large number of methods are developed that predict grasping pose for objects placed on a planar workspace. Chu et al. [18] proposed a DL model that predicted graspable locations of unseen objects when fed with an RGB-D image. The accuracy in the Cornell dataset was reported to be 96% and 96.1% in image-wise and object-wise splits, respectively. It is important to note that this method requires depth cameras, which are more expensive and computationally intensive than RGB cameras. Shin et al. [19] developed an algorithm that used Mask-RCNN to segment out objects kept on a plane. They used a distance metric to determine the smallest distance between two locations in the object. For this, several straight lines were found from the location of the center of gravity at 30°. Although it had a higher accuracy in predicting pose, it was computationally more expensive. In contrast to these works, Upadhyay et al. [20] showed the possibility of estimating 6D pose by using ORB and perspective-n-point (PnP) models. Despite the model being computationally efficient, the application scope is limited due to its reliance on features present on the surface of the object only. Additionally, objects in industrial settings are mostly texture-less, which further lessens the application scope of the system. Ma et al. [21] used the single shot detector (SSD) algorithm for object detection and employed PnP to estimate the pose of the object. The features utilized by this detection and estimation system are solely based on the object characteristics. On the other hand, Massa et al. [22] used a different approach by comparing the image of an object with its corresponding computer-aided design (CAD) model to extract features. The pose was estimated based on these features that had a higher similarity with the view of the input image. However, this method has a major drawback as it requires prior knowledge of CAD models of an object, which makes it impractical for industrial applications that involve a large variety and quantity of objects. In addition to single-shot detection methods, Zhang and Zhang [23] employed a two-stage detector, the Mask-RCNN model, for object detection and segmentation. The segmented images were then used to determine the pose of the object by orienting a minimum circumscribed rectangle to the detected object. However, this approach has a disadvantage as two-stage detectors require more computational resources. On the other hand, developing customized image processing algorithms can facilitate pose estimation, which is explained in a later section of this paper. Wang et al. [24] and Lee et al. [25] also used the YOLOv3 model for object detection and pose estimation. Nevertheless, it should be noted that the YOLOv3 model is computationally more expensive.

The literature review reveals that there are extensive applications of DL algorithms in automating manufacturing activities. However, the review also highlights the need for continuous research and development efforts to overcome the limitations of DL-based systems in industrial settings. One of the key limitations is that the objects in an industrial setting are usually texture-less and suffer from fluctuations in lighting conditions, resulting in noisy images. Another limitation is the lack of research on the detection of texture-less objects. Moreover, the use of

expensive depth cameras increases the overall cost of maintenance, which is another limitation.

To address these limitations, this article proposes the application of an advanced DL model named YOLOR for object detection [26]. YOLOR is a state-of-the-art model that outperforms previous ones in terms of precision, recall, map0.5, and F1 score with an optimum detection rate. The proposed model includes image acquisition by a low-cost webcam and analysis in the developed model, leading to real-time object recognition. The analysis excludes the use of expensive depth camera but still achieves higher accuracy due to the customized algorithms developed in this work. In addition, the article presents a novel algorithm to estimate the pose of objects on a planar workspace. The algorithm does not rely on either CAD models of the object or DL models, as found in the prior research. Instead, the pose estimation algorithm comprises morphological operations carried out on the object. Moreover, a sequence scheduling algorithm is developed to allow for collision-free grasping. These algorithms are integrated to form an AI solution, which is successfully tested on an industrial robot. The following lists the major contributions of this article:

1) An accurate YOLOR object detection and classification model is developed that is agnostic to object orientation and background. The model has an average accuracy of 95%, with an operation time less than 1 s.
2) A computationally efficient image processing algorithm is developed to estimate the pose of texture-less objects, with an operation time less than 1 s.
3) An accurate image processing algorithm is developed to enable a collision-free work environment for a robot. The algorithm has an accuracy of 100%, with an operation time of less than 1 s.
4) A real-time end-to-end framework is developed that includes the implementation of the proposed algorithms in an industrial robot, with superior and reliable results. The total operation time is less than 10 s when the robot is moving at 50% of its rated speed.
5) A real-time end-to-end intelligent framework is developed for industrial applications, such as autonomous pick and place, sorting, detection, and similar activities.

The strength of this study lies in the development of a cost-effective AI solution for handling industrial robot pick-and-place operations. The robot workspace is captured using a low-cost webcam, and the captured data is processed intelligently to compute the position and orientation of objects using image processing algorithms. Moreover, this study effectively addresses the challenge of handling texture-less objects. While prior research often relies on feature extraction techniques to determine object orientation, these methods are impractical for texture-less objects. This issue has been resolved by the YOLOR model by leveraging implicit and explicit features for inferencing.

The remainder of this paper is organized as follows. Section 2 presents the details of the methods used for developing the solution. The experimental details and robot specifications are also elaborated in this section. The results obtained by applying these methods are presented in Section 3. It also discusses the results of a few comparative studies. Section 4 lists the details of case studies performed for testing the solution in an industrial environment and the obtained results. The conclusions derived from this study, along with limitations and future scope, are listed in Section 5.

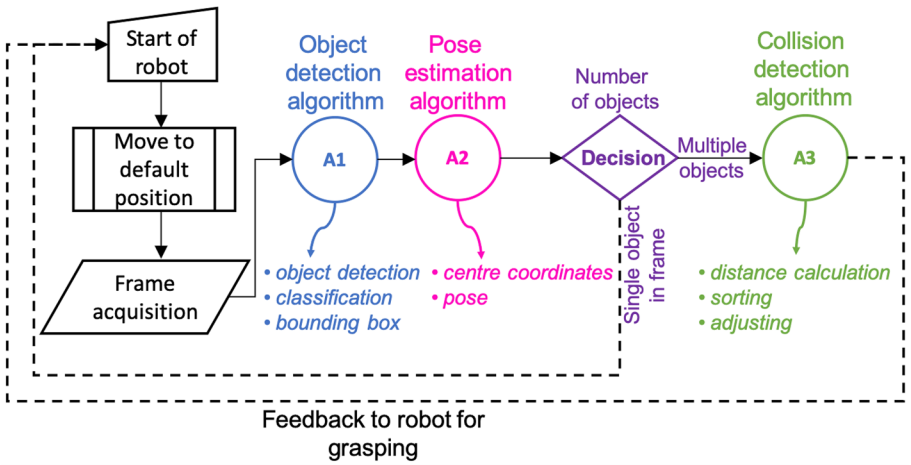
2. Methodology

Figure 1 presents the flowchart of the solution. The default position of the robot is defined in the robot's control panel. Therefore, each time a test is initiated, the robot moves to this position. A webcam is fixed on the robot arm that continuously captures pictures of the workspace. These frames are input to the object detection model (A1), as drawn in the schematic picture. This algorithm detects and classifies the boxes and creates bounding boxes around each. Using these bounding boxes, the center coordinates are calculated, and the pose is estimated in A2. Depending on the objects in the current frame, a decision is taken on the number of objects. If a single object is detected, the computed coordinates and pose are translated to the robot coordinate system. These coordinates are fed to the robot control panel in real time. Upon receiving the feedback, the robot arm actuates to pick that object by orienting itself to the object's pose and transports it to a destination. Otherwise, in the presence of multiple objects, a collision detection algorithm (A3) takes these objects as input and calculates the distance of each from the destination. Accordingly, the algorithm sorts those boxes present in the frame, and picks the one which satisfies both the conditions—(a) minimum distance from the destination and (b) clearance from other boxes. In the later sections, the details of each algorithm are described first and then the experimental details are presented.

2.1. Object detection and localization

A YOLOR model is utilized for object detection, which is a part of the YOLO series [26]. This algorithm has been proven in the prior

Figure 1
Flowchart of the solution



research having a faster inference speed and higher detection accuracy. In a YOLO framework, the input image is divided into S × S grid cells. For each cell, B bounding boxes are predicted by the model. Each bounding box has the following parameters:

1) p_c tells whether the predicted bounding box contains any object or not,
2) b_x and b_y represent center coordinates of a bounding box,
3) b_w and b_h represent width and height of a bounding box, respectively,
4) c_1, c_2, …, c_n represent number of classes for which the model is trained, and their values suggest whether an object of a class is present in a bounding box or not.

The YOLO models discard bounding boxes with $p_c = 0$, and those with $p_c = 1$ belongs to the highest-class probability. However, this can lead to multiple bounding boxes belonging to the same class. Non-max suppression addresses this issue by computing intersection over union (IoU) among similar bounding boxes. This is the basic functioning of YOLO models, starting from the YOLOv1 model.

However, the YOLOv1 model had lower detection accuracy for smaller objects and faced problems in localizing objects with unusual aspect ratio. To deal with this problem, the YOLOv2 model was introduced with the concept of anchor boxes. These are a predefined set of rectangular shapes that the model learns during the training phase using the K-means clustering algorithm. Bounding boxes formed by this model are relative to the anchor boxes. The YOLOv2 model has five anchor boxes by default, which were reduced to three in the YOLOv3 model. Additionally, the YOLOv3 model introduced a neck in the architecture, with multiscale feature maps that allowed it to detect objects of different sizes.

Later, the YOLOv4 model was introduced with CSP Darknet53 used in the backbone network, SPP+PAN in the neck network, and the same head as in the YOLOv3 model. To improve its performance, two new methods were added: Bag of Freebies and Bag of Specials. The first method increases training time without affecting the inference time and employs methods like data augmentation, Dropblock regularization, and CIoU loss. The second method increases the inference time by a small amount using an activation function. Later, the YOLOv4 model was further tweaked to develop several versions namely, Scaled-YOLOv4, YOLOv4-Large, and YOLOv4-Tiny. These resulted either by scaling up or scaling down the YOLOv4 model architecture. While scaling up increases the accuracy and lowers the speed, the scaled-down version increases the speed but reduces the accuracy.

The YOLOR model is an advanced extension of the Scaled-YOLOv4 model, incorporating both explicit and implicit knowledge into a unified network. This innovation allows YOLOR to achieve higher performance by leveraging a combination of learned feature representations and latent contextual knowledge. Traditionally, the objective function of a neural network can be expressed as:

$$y = f_\theta(x) + \in \tag{1}$$

where x represents the input data, $f_\theta(x)$ is the function learned by the neural network parametrized by θ, and ϵ is the error term. The output y is the prediction of the neural network. YOLOR introduces a modification to this framework presented in Equation (1) by integrating both explicit and implicit knowledge. The modified equation is given as:

$$y = f_\theta(x) + \in + g_\varphi(\in_{ex}(x), \in_{im}(z)) \tag{2}$$

In this updated formulation presented in Equation (2), ϵ_{ex} represents the explicit error derived from the inputs x, and ϵ_{im} refers to the implicit error derived from the latent code z, which captures deeper, high-level features of the input. The function g_φ combines these errors, explicit and implicit, into a unified learning signal, with the parameter set φ controlling this operation.

The explicit knowledge refers to the features and patterns learned by the neural network directly from the inputs, such as edges, textures, and object contours in the case of object detection. The explicit knowledge is encoded in the lower to mid-layers of the network, where fine-grained details are learned through traditional training techniques. On the other hand, the implicit knowledge refers to the high-level, abstract patterns captured in the deeper layers of the network. These are not directly tied to specific inputs but are rather latent, learned representations that provide contextual or semantic information about the input data. The use of implicit knowledge is a powerful approach to improve generalization, especially in complex environments where explicit features alone may not suffice for accurate prediction.

Previous research has shown that models incorporating both explicit and implicit knowledge can outperform those replying solely on explicit features. The combination allows the model not only to make predictions based on the immediate data but also to incorporate context or higher-order relationships between features, thus improving task generalization and model robustness. The architecture of YOLOR incorporates three key components that contribute to its performance improvements. The first component is the alignment of kernel space, which ensures that the kernels used in convolutional layers align well with the spatial features in the input data, optimizing feature extraction and preserving important details. The second component is the refinement of predictions, which is accomplished by leveraging both explicit and implicit knowledge at various stages of the network. This enables more accurate localization and classification, especially in challenging scenarios where objects are small, occluded, or in complex backgrounds. The third component is the creation of the CNN, which efficiently processes the explicit and implicit knowledge.

Moreover, the YOLOR model introduces several optimizations compared to the YOLOv4 model. These optimizations make it more efficient and better suited for real-time applications, especially in industrial settings. One of the key optimizations in YOLOR is the smaller input size, which reduces the computational burden and increases inference speed while maintaining high detection accuracy. This change accelerates the model performance by working with lower resolution inputs, making it ideal for real-time applications such as automated inspections in manufacturing environments. Furthermore, YOLOR uses fewer number of parameters in the network stem, backbone depth and out channels, and neck out channels. These reductions contribute to a more lightweight model, which is crucial for deployment on edge devices. This optimization makes YOLOR well-suited for industrial applications where low-latency decision-making are essential. Real-time performance is crucial for robotic manufacturing environments, where fast and accurate detection has a direct impact on throughput and quality control. Moreover, these optimizations lead to a cost-effective system deployment, as it requires fewer hardware resources compared to previous YOLO models. Besides, one of the significant updates in YOLOR is the replacement of the Mish activation function used in YOLOv4 [27] with SiLU [28]. The SiLU activation function has been shown to improve model performance, especially in terms of training stability and gradient flow. This update allows YOLOR to maintain both high accuracy and computational efficiency, making it more robust for deployment in real-world industrial applications.

In the present work, the pre-trained YOLOR model was carefully manipulated to learn and generalize the present problem. Our dataset consisted of 2 classes (yellow- and orange-colored boxes). In industrial settings, boxes can be of different colors. Thus, two contrasting colors are chosen. Transfer learning is employed for model learning. Around 20 images of objects in the workspace were captured using the webcam. A dataset of 3283 images was created from these 20 images by using

two data augmentation techniques—rotation and changing illumination. Out of these 3283 images, 2541 images were used for model training, 495 images for validation, and the remaining 247 images were used for testing. The validation set is samples of data that were held back from model training and were utilized to estimate the model performance over time. Accordingly, hyperparameters were tuned for accurate results.

To enable the system to be applicable for industrial settings, the dataset was carefully tweaked so that the model should be able to generalize the actual shopfloor situation. The industry settings are harsh environment with low lighting conditions and even the lighting may fluctuate in certain locations. Contrastingly, DL models developed are fit for a laboratory environment. Thus, they would fail when applied in an industrial workplace. To tackle the changes in illumination, brighter and darker images of the same scene were included in the dataset as presented in Figure 2.

An original image of boxes is presented in Figure 2(a), and (b) is the darker variant which was created by decreasing the γ value of the original image. Figure 2(c) presents the brighter variant obtained by increasing the γ value. The width and height of the input image were a multiple of 32, as desired by the YOLOR model. To maintain the frame rate, the captured frames were resized to default dimensions of input image, i.e., 640 × 640. Once an object is detected by the model, the following 7 parameters were obtained:

1) x_{min}, y_{min} – refers to x and y coordinates of upper left corner of a bounding box,
2) x_{max}, y_{max} – refers to x and y coordinates of upper right corner of a bounding box,
3) a confidence score – suggests how confident the model is that an object is contained in a bounding box,
4) a class – represents the number assigned to a class,
5) a name – name of the class.

This work contains two classes namely—yellow and orange boxes, with numbers 0 assigned to yellow-colored boxes, and 1 to orange boxes, respectively. These 7 parameters are utilized to detect and classify objects on the workspace. Furthermore, the bounding boxes' parameters are utilized to localize objects. A bounding box is drawn around a detected object, as presented in Figure 3 with its associated parameters. The center (C_x,C_y) of the box is calculated by finding the center of the corresponding bounding box as presented in Equations (3) and (4).

$$c_x = \frac{(x_{max}+x_{min})}{2} \tag{3}$$

$$c_y = \frac{(y_{max}+y_{min})}{2} \tag{4}$$

In order to evaluate the model, two metrics were used: mean average precision (mAP) and average accuracy. The mAP metric

Figure 2
Images of different illuminations: (a) original, (b) darker variant of original image, and (c) brighter variant of original image

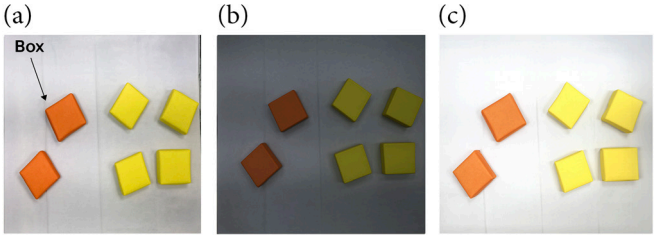
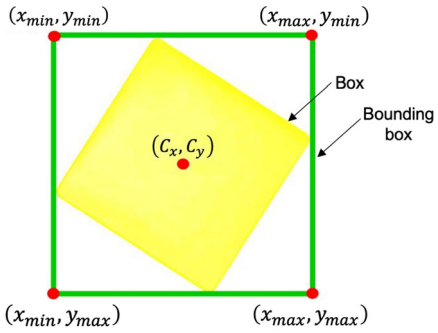


Figure 3
Picture of a box with bounding box and coordinates



assesses the precision of the model by comparing the predicted bounding box with the ground truth bounding box. The mathematical expression for calculating mAP is given in Equation (5), where n represents the number of classes and AP denotes the average precision of each class. This score measures how accurately the model classifies objects. During the training process, mAP was utilized to track the learning curve of the model after each epoch.

$$\text{mAP} = \frac{1}{n}\sum_{i=1}^{n}\text{AP}_i \tag{5}$$

Next, average accuracy was calculated as the mean of the confidence scores predicted by the model for each class. This is represented mathematically in Equation (6), where C_i indicates the confidence score of the i-th prediction made by the model for a specific class and N denotes the total number of predictions. The confidence score is derived from the objectness score and class prediction.

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}c_i \tag{6}$$

When the YOLOR model processes an input image, it divides the image into grids. For each grid, several anchor boxes are created based on the model predictions. These anchor boxes serve as initial estimates for object locations, after which an objectness score is computed for each anchor box. The objectness score, ranging from 0 and 1, indicates the probability that a particular box contains an object. Subsequently, the model makes a class prediction for each anchor box, which is also represented as a value between 0 and 1. Using the objectness score and class prediction, multiple confidence scores are calculated. Finally, through non-maximum suppression, the anchor box with the highest confidence score is selected for that particular object. The average accuracy is calculated separately for each class. In this study, the model was developed on a GTX 1650 Ti 6 GB GPU machine with a VRAM of 5 GB for inference.

2.2. Grasp pose estimation

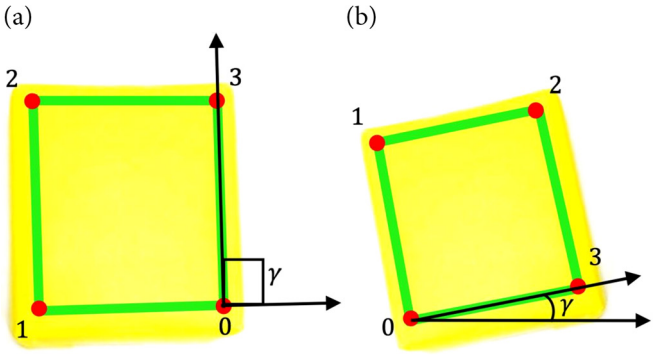
To estimate the grasp pose of a box, it was required to compute its yaw with respect to the webcam fixed on robot. The bounding boxes drawn by the object detection model were extracted as region of interest (RoI). The goal was to get a perfect contour of the target box so that a minimum fitting rectangle can be drawn around it to compute its yaw. The minimum fitting rectangle is drawn using an inbuilt function in OpenCV framework. This function draws a rectangle around a given contour such that the area of the rectangle enclosing the contour is minimum. It also returns the angle of the rectangle drawn

which ranges from 0^o to 180^o. The contour of the target object can be found by using methods like segmentation using HSV color space, background subtraction, edge detection, or by using segmentation using DL. Segmentation using DL is computationally expensive and is prone to illumination changes with HSV color space. Thus, they were not utilized. Background subtraction is a good candidate for contour formation, but even a slight movement in camera position would render incorrect results. In addition, background subtraction is also affected by illumination changes. Therefore, edge detection method was selected to form the contours. A Canny edge detector, which is proved to be the best edge detection algorithm [29], was utilized to determine cuboid edges from the RoIs.

Another challenge in this work was faced owing to the use of an inexpensive webcam to capture the images. This was intentionally considered a part of the solution to reduce the operational and maintenance cost. The images acquired from webcam were of poor resolution and thus, edges were not detected clearly. Figure 4(a) shows an original, and its edge detected image is presented in Figure 4(b). Clearly, the result is poor. Also, parts of some boxes were inside the RoI resulting in multiple edges. This problem is addressed by employing a few morphological operations that help in removing these imperfections. The edges were made more prominent by dilating the RoI for 5 iterations using a kernel of size 5 × 5. A binary image is obtained by dilation which is presented in Figure 4(c). Black patches are formed that represent the target box and its surrounding boxes. To remove the black patches of the surrounding boxes, the middle part of the RoI is pasted in a white background as shown in Figure 4(d), and the binary image is inverted. This results into a binary image with a white patch in the middle, as presented in Figure 4(e). This image represents the detected box in our case. Therefore, the largest contour is detected and a minimum area fitting rectangle is drawn around it as depicted in Figure 4(f).

To compute the yaw of the rectangle drawn and presented in Figure 4(f), the corner points of the box are ordered in a clockwise manner, starting with the lowermost point. This process is presented in Figure 5(a). In the case of two points being at the lowermost positions, the rightmost point is considered as the starting point, which is presented in Figure 5(b). In this case, the angle between the line joining

Figure 5
Calculation of yaw of box: (a) straight position and (b) angular position



the starting and end points with the horizontal is also calculated. The range of this angle is constrained between 0 and 90°.

As the present work also deals with multiple objects in a single frame, an estimate of the correct grasping pose for a target object was essential for the robot's actuation. In addition, collision-free actuation was also essential. x and y axes of all boxes in a scene were drawn using their yaw angles, found previously, as shown in Figure 6(a). The distance of all boxes from a target object is calculated using their center coordinates. If the distance of an object is lesser than a threshold value determined experimentally, then the midpoint of that object is considered for further calculations. This is to avoid collision with the gripper. The threshold region is shown as a circle in Figure 6(b).

This threshold was the minimum distance between the center of two boxes such that the plate of the gripper just fits in the gap between these boxes. Once all such objects are identified, a line joining their center with the target box center is drawn. The angles of all these lines are found with respect to the x-axis of the target object as shown in Figure 6(c). Accordingly, the target box was divided into 4 regions which are presented in Figure 6(d). If the calculated angle lies in Region 1 or Region 3, then the object cannot be picked from those regions. Then, Regions 2 and 4 are checked. If these regions were also occupied,

Figure 4
Process of edge detection and contour formation: (a) detected box, (b) edge detection without applying morphological operation, (c) edge detection after dilation, (d) transformation by patch formation, (e) inverted image after transformation, and (f) contour detection and fitting rectangle

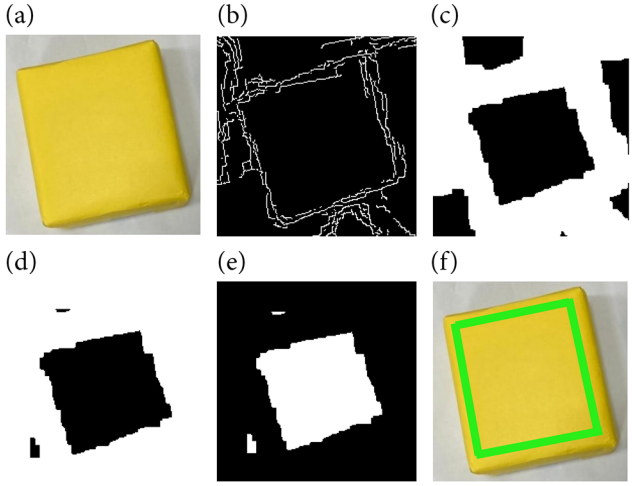
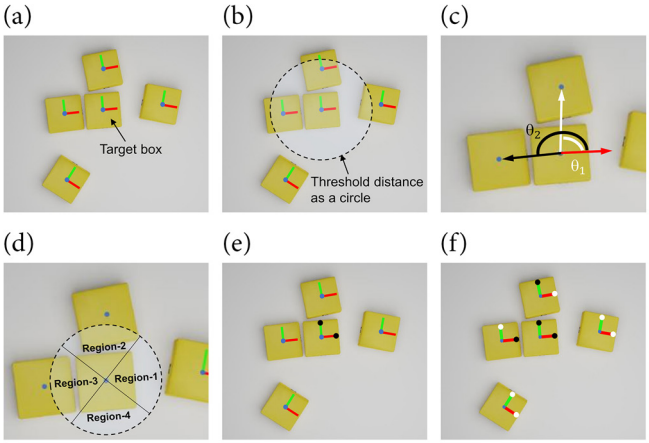


Figure 6
Process of pose estimation and collision avoidance in multiple objects in a scene: (a) detection of target box, (b) threshold distance estimation, (c) position of all objects with respect to target box, (d) region-based segmentation from target box, and (e) and (f) black and white dots representation for objects that cannot and can be picked



then the target object cannot be picked at the present moment. A black dot is used to represent that the box cannot be picked from that region, which is otherwise indicated with a white dot. In case the box can be picked from both set of regions, then Regions 1 and 3 are selected for grasping by default as the gripper has to rotate by at least an angle from its default position. Figure 6(e) shows the black dots which are present in both axes of the target box as angle θ_1 lies in Region 2 and θ_2 in Region 3. This process is repeated for all boxes as shown in Figure 6(f).

After identification of the grasping region, the grasping angle is calculated. Table 1 presents the calculation of grasping angles for different regions, where γ is the yaw angle of the box. The grasp pose is constrained between −45° and 45° if the object is to be picked from Regions 1 and 3. For Regions 2 and 4, the angle is constrained between 45° and 135°. To evaluate the efficiency of the pose estimation algorithm, the deviation between the actual and estimated yaw angles were recorded.

2.3. Grasping

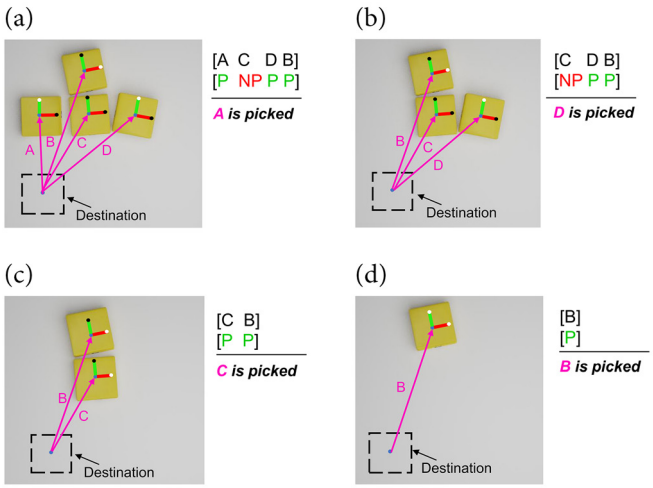
A sequence scheduling algorithm is devised that decides the sequence of picking the boxes. As the boxes are picked one at a time, the total time of a cycle is always constant. It does not depend on the sequence of picking the boxes. Therefore, the boxes are picked in increasing order of their distances from the destination. The destination is considered to be the bottommost and leftmost point in a captured frame. Figure 7 schematically details this algorithm. In the figure, 'A', 'B', 'C', and 'D' represent distances of boxes from the destination.

These are calculated by using distance formula between the mid-point of boxes and mid-point of destination. With the distance values, they are sorted in increasing order. Also, it is tracked which of the boxes present in a scene can be picked. The boxes that can be picked are indicated with 'P' or otherwise are indicated with 'NP'. In Figure 7(a), the sorted list is represented inside square brackets to the right of the figure. The first box in the list is 'A' and it can be picked up, hence it is transferred to the destination. Next box is 'C', as presented in Figure 7(b). However, it cannot be picked. The algorithm moves to the next box which is 'D' and it is transferred to the destination. Likewise, boxes 'C' and 'B' are transferred next as shown in Figure 7(c) and (d), respectively. The sequence scheduling algorithm was evaluated by calculating the percentage of successful pick-ups by the robot with respect to the total number of trials.

Table 1
Grasp pose estimation

	Formula for grasping pose	
Region of grasping	$0° \le \gamma \le 45°$	$45° < \gamma \le 90°$
Regions 1 and 3	γ	γ − 90°
Regions 2 and 4	γ + 90°	γ

Figure 7
Process of sequence scheduling algorithm: (a) to (d) sequence of boxes transferred from workspace to destination



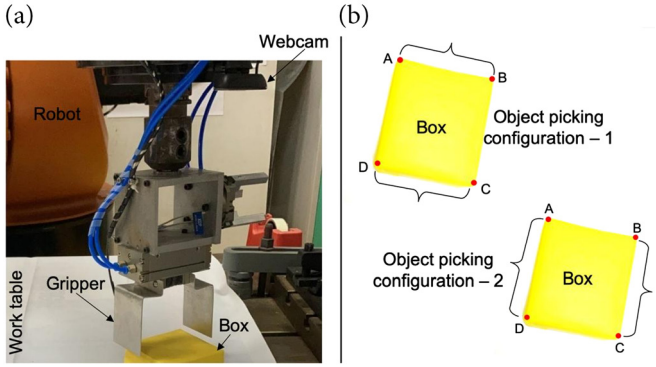
The experimental setup is shown in Figure 8 that consists of a 6-DoF industrial robot (KUKA KR500 R2380 MT), a parallel plate pneumatic gripper attached to its end effector, a webcam fixed to robot, and a work table to hold the boxes. The gripper system was connected with a robot controller so that the boxes can be clamped for picking and unclamped for placing. The boxes are laid on a white sheet at different locations and at various orientations without stacking. The color of the sheet is decided such that there is a contrast between objects and sheet. The webcam is connected to a local machine that collects RGB images. The default position of the robot is selected in a manner such that the camera lens is parallel to the workspace surface covering a maximum area. As the camera lens is parallel to the workspace, the z-coordinate of all objects is known. This step also makes the pitch and roll values of all objects almost the same. Hence, for a successful grasp, only x- and y-coordinates and yaw of an object is required. Considering the gripper and cuboid structure of objects, they can be picked only by two configurations as shown in Figure 8.

The communication between robot and local machine running the algorithms is established using OPC-UA protocol. This is a machine-to-machine communication protocol that is widely used in industrial automation. The robot is treated as the server and the client is the local machine. The server and client machines are connected through an Ethernet cable for data transmission. The OPC software gathers information about the robot variables which are listed as node-ids.

In order to feed information to the robot controller, the grasp pose coordinates which are computed in terms of pixels were transformed into robot coordinates. The x, y, and z coordinates of the robot are in millimeters. The image resolution is set at 720 × 1280 pixels so as to get more pixels for better precision. As the camera position is fixed, it captured a fixed length and breadth of the workspace. They are 710 mm in length and 830 mm in width. The corresponding values in pixels are 914 × 1054 as presented in Figure 9.

The distance per pixel along the length was found by dividing 710 with 914, and 830 with 1054 for breadth. The origin of this new

Figure 8
(a) Experimental set up and (b) possible configurations for object picking



coordinate system is shifted to a point just below the gripper center on the white workspace. The pixel coordinate of this point was (621,744). It was also observed that when the robot moved along the x-axis of pixel-based coordinate system, y-coordinates of the robot coordinate system changed. Similar relation was also observed between y pixels of the pixel-based coordinate system and x-coordinate of the robot coordinate system. At origin, x- and y-coordinates of the robot were 1719.74 mm and 184.5 mm, respectively. Hence, a linear relation was formed between the pixel-based coordinate system and the robot coordinate system, which is presented in Equations (7) and (8).

$$X = (744 - Y_{pixel}) \times \left(\frac{710}{914}\right) + 1719.74 \quad (7)$$

$$Y = (621 - X_{pixel}) \times \left(\frac{830}{1054}\right) + 184.50 \quad (8)$$

where, X and Y are x- and y-coordinates of the coordinate system in millimeters, respectively, and X_{pixel} and Y_{pixel} are x- and y-coordinates of the pixel-based coordinate system in pixels, respectively. By using Equations (7) and (8), some offset in both x and y directions were observed. Therefore, the differences were determined experimentally by performing several tests. Using the obtained differences, a linear equation was devised by taking the difference as dependent variable and corresponding pixel coordinates as independent variable for calculating the offsets. Thus, the final equations are presented in Equations (9) and (10).

$$X = (744 - Y_{pixel}) \times \left(\frac{710}{914}\right) + 1719.74 + (Y_{pixel} \times 0.0123) - 7.97 \quad (9)$$

$$Y = (621 - X_{pixel}) \times \left(\frac{830}{1054}\right) + 184.50 + (X_{pixel} \times 0.0286) - 2.2148 \quad (10)$$

The angles of both coordinate systems are in the same units (o), while the reference points were different. For 0° in the pixel-based coordinate system, the corresponding angle was −117.39° in the robot coordinate system. Thus, the required angle was calculated using Equation (11).

$$Angle_{robot} = -117.39 + Angle_{pixel} \quad (11)$$

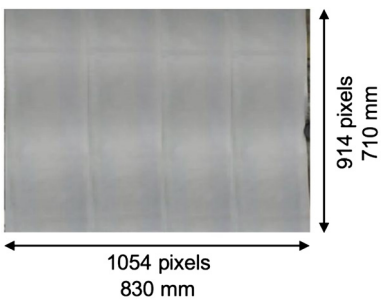
where, $Angle_{robot}$ is the yaw value in the robot coordinate system and $Angle_{pixel}$ is the yaw value in the pixel-based coordinate system.

3. Results

3.1. Object detection

As reported, a DL model is developed for object detection and classification. Prior research communicates the use of traditional methods for object detection. Therefore, a comparative assessment of these two methods is performed and their results are discussed.

Figure 9
Dimension of sheet in millimeters and pixels



3.1.1. Detection of box using feature matching-based algorithms

Experiments are conducted using SIFT [30], SURF [31], and ORB [32] algorithms to detect the boxes present in a scene. Firstly, a source image is created whose features are compared with features extracted from the target image. If a certain number of features match, then the target image is said to contain the object of the source image. An image of the box taken from a webcam was given as input to these algorithms to extract feature points. The results of the experiments are shown in Figure 10. The extracted features are marked in green. Figure 10(a) shows features extracted by SIFT, Figure 10(b) shows the same obtained by SURF, and Figure 10(c) shows the ones extracted by ORB. Only a single feature is extracted by the SURF algorithm which is also insufficient for a feature matching process. Many feature points are extracted by SIFT and ORB algorithms at the edges and corners of the box. However, these feature points give false results in the feature matching process. This proves they are not capable of detecting texture-less objects.

3.1.2. Detection of box using DL model

As reported, the objective was to develop a generalized model that would be suited for industrial settings. Therefore, the DL model was trained with an augmented dataset containing images of varying intensities. In order to compare the usefulness of such a model, DL models were also developed with—(a) as-captured images from a webcam, (b) brighter variant, and (c) darker variant. The obtained results are compared with that of the hybrid model, which is trained with a balanced dataset.

Prior to this, the accuracy metrics of the hybrid model are presented. The hybrid model was trained for 15 epochs. Figure 11 presents the loss as a function of the number of epochs. It suggests how well a model is able to locate an object's center and the predicted bounding box covers an object. The training loss is found to be decreasing with increasing number of epochs. Overfitting of the model was prevented by the early stopping criterion which allows training only up to a certain point. With only 15 epochs of training, satisfactory results were obtained on the validation set. The nature of the plots indicates both training and validation losses do not converge. Therefore, they are not over-trained and do not suffer from overfitting.

Figure 12 presents objectness loss which is a probability measure suggesting that an object exists in a proposed RoI. This loss measure of training and validation sets shows that the model got better at predicting the RoI with increasing epochs.

Figure 13 presents classification loss which suggests how well the model is able to predict the correct class of a given object. After the first epoch itself, the loss decreased dramatically indicating that the model found suitable weights. The training loss converged within a few epochs only, indicating its generalizability to the present situation. However, the validation loss is not as smooth as the training loss, which is probably due to the unseen lighting conditions in the images of the validation set. Therefore, it took a few more epochs to converge.

Figure 10
Feature extraction by traditional object detection algorithms: (a) SIFT, (b) SURF, and (c) ORB

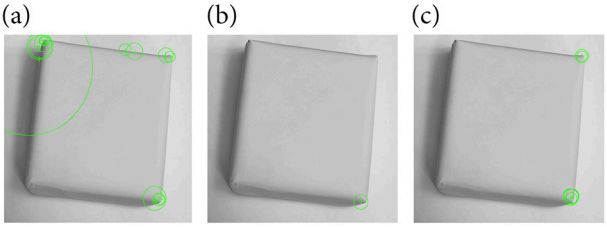


Figure 11
Loss of hybrid model: training and validation sets

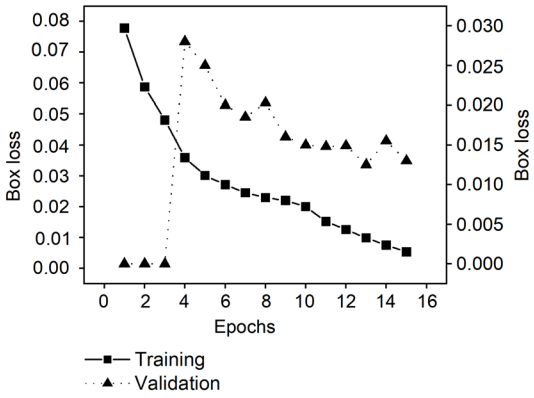


Figure 12
Objectness loss of hybrid model: training and validation sets

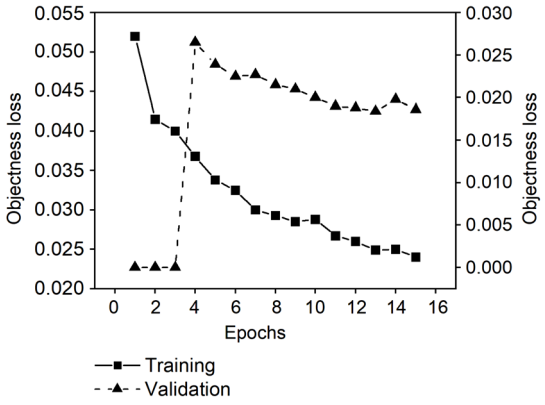


Figure 13
Classification loss of hybrid model: training and validation sets

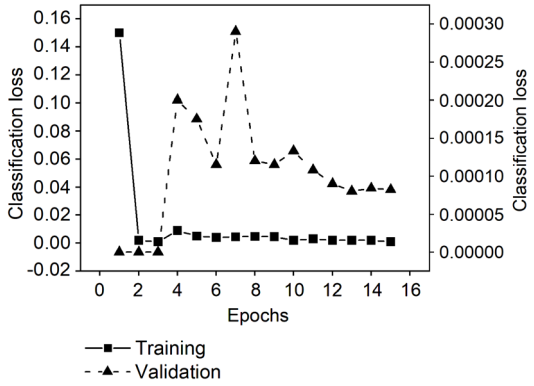
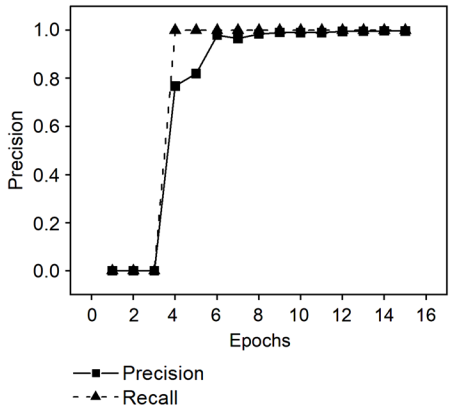


Figure 14 presents the precision and recall of the model. Precision is calculated as the ratio of correctly predicted boxes (true positives) to the total number of predictive positive boxes (true positives and false positives). It indicates how many of the predicted positives are actually correct. Recall is calculated as the ratio of correctly predicted boxes (true positives) to the total number of actual positive boxes (true positives and false negatives). It measures the model's ability to detect all relevant instances of a certain class. Within a few epochs, both the precision and recall of the model converge to 1, indicating high performance in classification.

Figure 14
Hybrid model accuracy: precision and recall



The mean average precision (mAP) which compares the detected box to the ground-truth bounding box is shown in Figure 15. The model detection accuracy is measured by the mAP score, ranging from 0 to 1. Only a few epochs were required for the hybrid model to converge.

The accuracy for detecting yellow boxes is determined to be 94% and that of orange boxes is 92%. This is due to the greater number of yellow boxes (10,684) as compared to the orange boxes (5863) in the dataset. The usefulness of this hybrid model is assessed by comparing its accuracy with three other models, as stated earlier. These models are the ones that are developed with—(a) as-captured images from a webcam, (b) brighter variants of as-captured images, and (c) darker variants of as-captured images. All the models were trained with 2000 number of images. A testing dataset was made that contained 210 images with illumination changes. These images were made by altering the lighting condition of the scene. Figure 16 compares their performances in terms of mAP@0.5:0.95 and precision, which shows the superiority of the hybrid model over others.

3.2. Center coordinate, yaw, and grasp pose estimation

The midpoint and pose of the object are found successfully by using the devised algorithm. The calculated yaw of the boxes has an error of ±3°. This error is pretty small and thus is acceptable because it does not create any problem while grasping. The predicted bounding boxes, estimated midpoints, and yaw of multiple objects in a frame are presented in Section 4. The grasp pose of the boxes was also estimated successfully by using midpoints and corresponding yaw value of those boxes. The region of grasping of all boxes in a random scene is shown in Figure 17. Black dot represents that the box cannot be gripped from that particular region whereas white dot indicates that the box can be gripped from that region. Some boxes had both dots as black indicating that those boxes cannot be picked at that particular moment.

3.3. Comparison of model performance and benchmarking

A comprehensive comparative study was executed to validate the effectiveness of the YOLOR model compared to other YOLO versions for object detection. This involved evaluating YOLOv8 and YOLOv9 using the same machine, dataset, and training duration. The models were assessed based on accuracy and framerate metrics, which are presented in Table 2. The YOLOR model demonstrated a superior accuracy rate of 95% compared to YOLOv8 at 89% and YOLOv9 at 80%. Although YOLOR has a lower framerate than other YOLO models as listed in Table 2, its estimates are more accurate. These findings highlight the

Figure 15
Hybrid model accuracy: mAP@0.5 and mAP@0.5:0.95

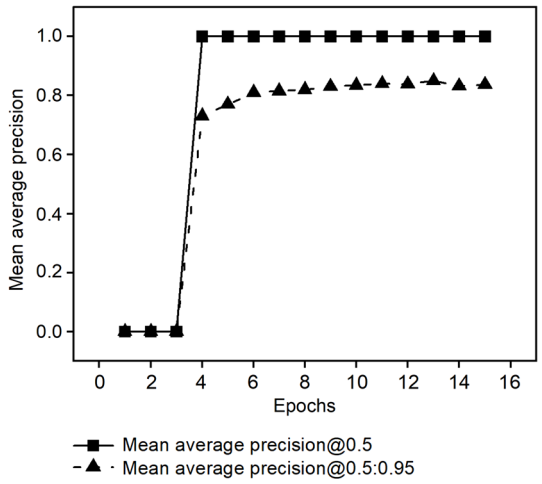
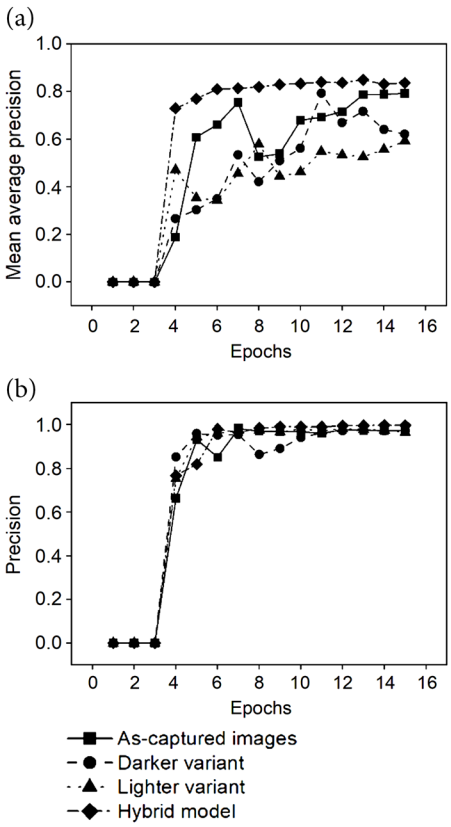


Figure 16
Performance comparison among hybrid model and normal models: (a) mean average precision and (b) precision



potential of the YOLOR model for efficient and precise object detection, particularly in industrial settings where rapid detection is crucial for safe and effective operations. Furthermore, the inference speed is 40 ms which was sufficient for processing 640 × 640 resolution images.

Furthermore, state-of-the-art algorithms from the literature were considered to develop models for object detection. This was done to compare their performance with the YOLOR model and to present the seamless integration within the DL system developed in this study.

Figure 17
Estimation of region of grasping of boxes in a random scene

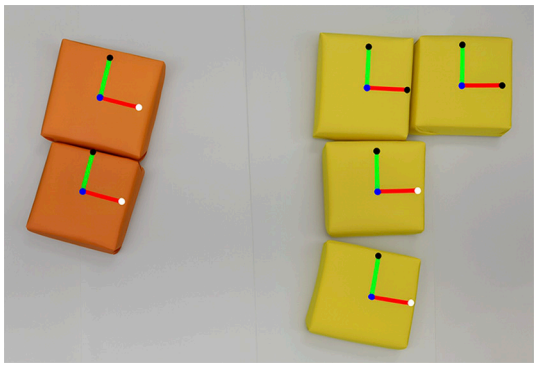


Table 2
Performance of YOLOR vs other YOLO versions

Sl. no.	Model	Average accuracy (%)	Framerate (FPS)
1	YOLOv8	89	67
2	YOLOv9	80	50
3	YOLOR	95	24

Table 3 lists the accuracy obtained with the state-of-the-art models and YOLOR model. The average accuracy of YOLOR is comparable to the latest methods in the literature. Additionally, these models provide bounding boxes post-detection, allowing for the extraction of detected images to predict grasp points, reflecting the ease of integrating newer models into existing systems to extract the detected images. These extracted images can then be sent through our system to predict grasp points, showcasing the plug-and-play solution with seamless integration of newer models.

3.4. Model augmentation

To expand the application scope of the developed DL-based system, experiments were conducted to augment the ability of the DL model to recognize objects of different shapes. In this experiment, circular-shaped objects were utilized to apply the entire DL pipeline. The YOLOR model was trained with approximately 3000 images for 30 epochs to detect the circular-shaped objects, achieving an accuracy rate higher than 90%.

Table 3
Benchmarking

Sl. no.	Reference	Model	Average accuracy (%)
1	[13]	YOLOv5	85
2	[14]	DCSPose	94
3	[33]	YOLOv3 + GG-CNN	94
4	[15]	Improved Nanodet	94
5	[34]	Cycle-GAN + Mask-RCNN	96
6	This study	YOLOR	95

While circular-shaped objects can be picked up from any yaw angle, a feasibility test was conducted in this study where the image of the detected object was passed through the DL pipeline. Figure 18 presents the detection results, confirming the generalizability of the DL-based system developed in this work to objects of any shape. The figure also shows that the angle predicted by the system was 0°, highlighting the potential of the system to be applied to objects of varying shapes.

These findings demonstrate that the developed DL-based system can be adapted to recognize objects of various shapes, thereby expanding its application potential. This has important implications for manufacturing industries, where objects of different shapes are encountered, and the ability to detect them accurately and efficiently is essential for smooth functioning of the manufacturing processes.

4. Testing of the Solution – Implementation in a Robot

The YOLOR model, retrained with ~3000 augmented images, achieved over 90% detection accuracy. The developed DL-based system, including contour extraction and grasp analysis, was highly effective in operation and efficient in computation. The total pipeline, including object detection, pose estimation, collision checking, sequence scheduling, and object picking, is accomplished within 10 s when the robot moves at 50% of its rated speed. The inference time for object detection using a GTX 1650 Ti 6 GB GPU with 5 GB VRAM usage was on an average 40 ms per frame of 640 × 640 resolution. Pose estimation and collision avoidance rely on lightweight image processing operations. These are linear-time operations with respect to the number of pixels in consideration O(p), where p is the number of pixels. The sequence scheduling algorithm uses simple distance-based sorting, resulting in O(nlogn) complexity, where n is the number of detected objects. In addition, the solution is implemented in Python using standard open-source libraries such as PyTorch for deep learning and OpenCV for image processing. This makes the system highly accessible, easy to modify, and portable across platforms, facilitating rapid deployment and maintenance in industrial settings. To test the credibility of the developed solution, several experiments were conducted at varying configurations, and those are presented in this section.

4.1. Presence of single object in camera frame

In this experiment, it was assumed that the robot's workspace consisted of a single box only. Figure 19(a) presents a picture of this workspace, where two boxes can be observed. When the program started,

Figure 18
Detection of circular-shaped object and its grasping parameters

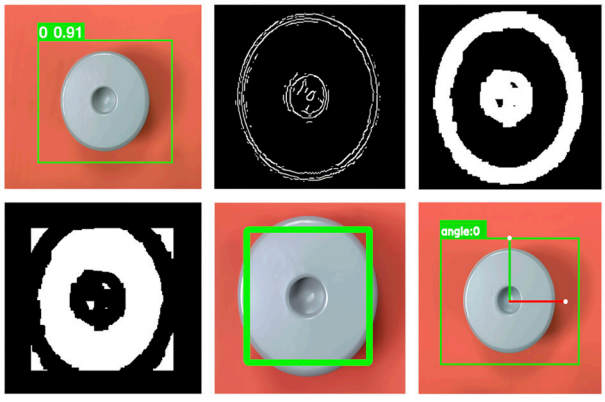
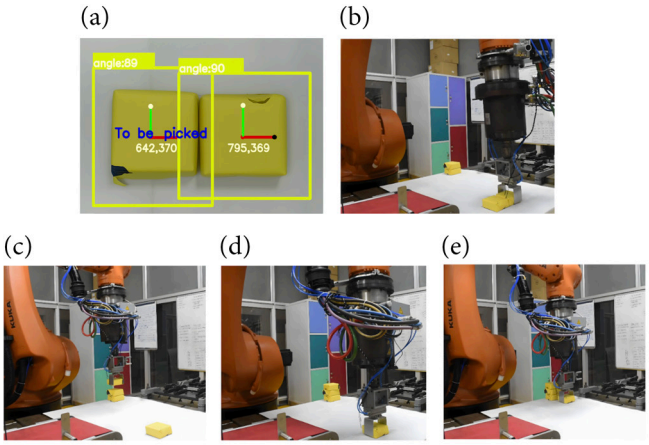


Figure 19
Experiment with single boxes in the workspace: (a) image acquisition and object recognition, (b) gripping, and (c, d, e) picking and stacking



the robot arm moved to the designated location, where the webcam captured frames of the workspace. These frames were then analyzed, leading to the detection and classification of the boxes. The detection results are illustrated in Figure 19(a). Then, the program estimated the pose and calculated the distance to each box in the workspace. Out of the two boxes, the one positioned on the left was determined to be closer and therefore selected for pick up. The distance measurements can also be observed in Figure 19(a). After making this decision, the robot moved to grasp the selected box. It oriented itself appropriately to ensure a proper grasp, as shown in Figure 19(b, d). Finally, the box was transported to its destination, depicted in Figure 19(c, e).

This experiment was repeated several times to ensure the repeatability. The average time of picking the box is 5 s. However, detection and estimation processes were accomplished in milliseconds.

4.2. Presence of multiple objects in a structured form

In this experiment, it was assumed that the robot's workspace consisted of multiple boxes arranged in an orderly structure, as illustrated in Figure 20. The operation is depicted in Figure 20, where the pick-and-place sequence is from left to right, and the robot's movement is from top to bottom correspondingly. When the program began, the robot arm moved to the designated location, and the analysis commenced, as presented sequentially in Figure 20. All the boxes were successfully transported from the workspace to the destination without any collisions with other boxes. This experiment demonstrated the potential of the developed system for use in industrial applications, where objects are typically arranged in a structured manner and need to be picked up and placed efficiently.

4.3. Presence of multiple objects in an unstructured form

An additional experiment was conducted in which the boxes were arranged randomly within the workspace. This random configuration of the boxes is presented in Figure 21. The figure also depicts the sequence of the pick-and-place operations as determined by the developed system, moving from left to right, while the robots' movements proceed from top to bottom. The results demonstrate the system's efficiency in handling these dynamic conditions in real time, all without any collisions.

Figure 20
Experiment with multiple boxes in the workspace arranged in a structured manner

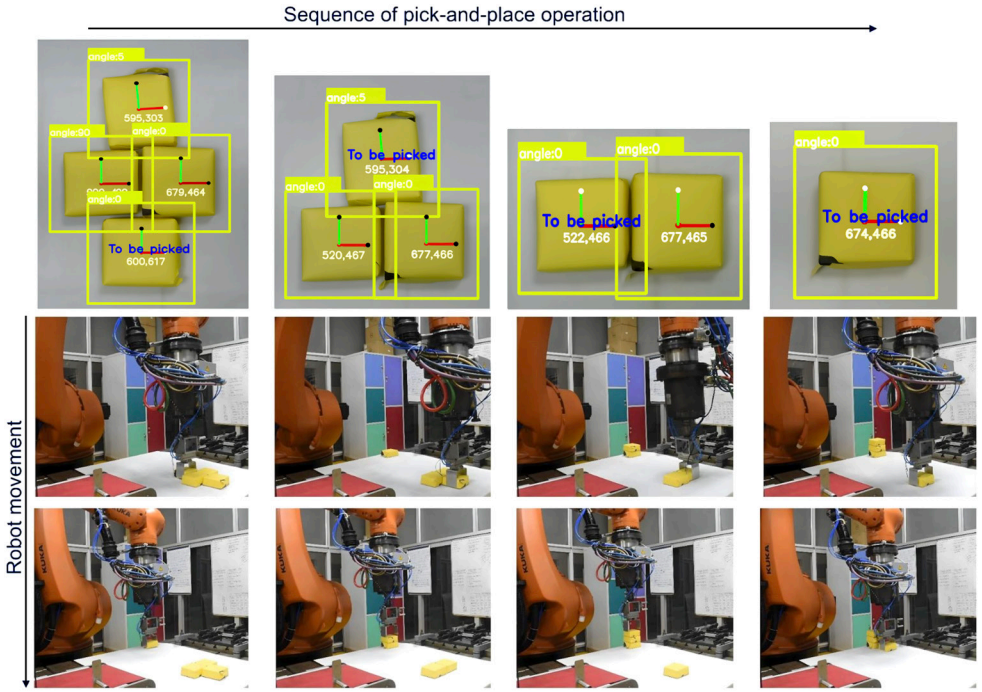
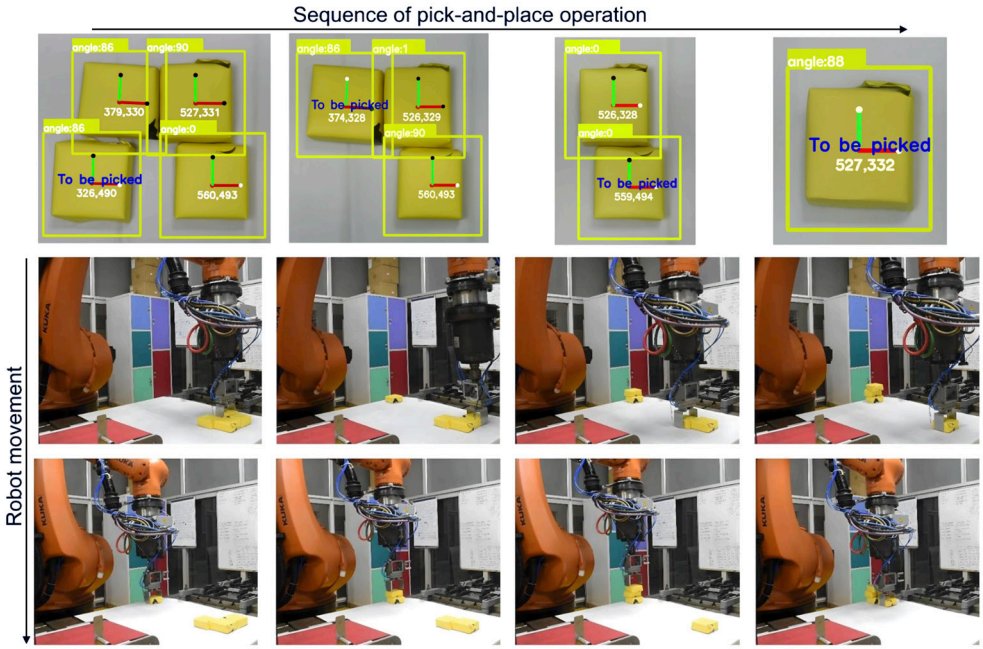


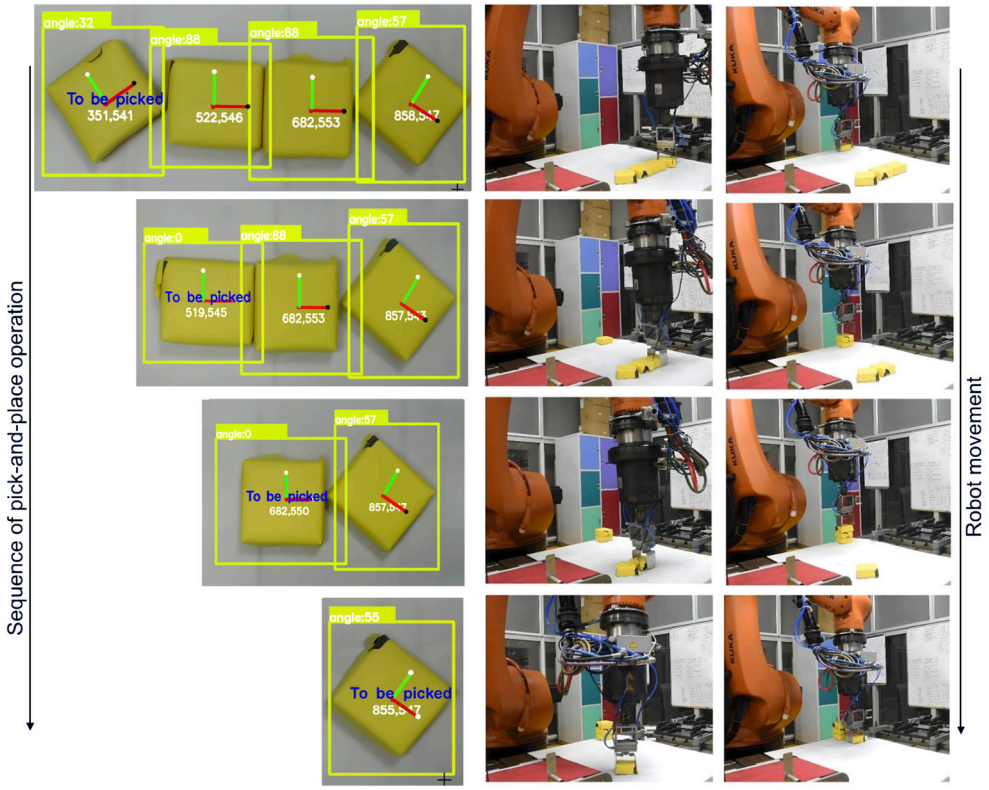
Figure 21
Experiment with multiple boxes in the workspace arranged in an unstructured manner



4.4. Presence of multiple objects in a complex pattern

In this experimental set, multiple boxes were arranged in a complex order within the workspace, as depicted in Figure 22. When the program started, the system determined the sequence of pick-and-place operations, which are presented from top to bottom in Figure 22, along with the corresponding robot movements. The results indicate that the system effectively handled the given scenario and successfully transported the boxes to their destination without any collisions. This demonstrates the efficiency of the system in accurately picking up the boxes while avoiding any contact with the surrounding boxes.

Figure 22
Experiment with multiple boxes in the workspace arranged in a complex pattern



5. Conclusion

In this paper, a solution is presented that automates pick-and-place operations of texture-less objects by using an object detection algorithm, a classification and pose estimation algorithm, accurate robotic grasping, and a collision avoidance algorithm. These algorithms are integrated into a real-time DL solution tested in an industrial environment. The obtained results clearly demonstrate that a hybrid DL model is necessary for effectively handling the noise typically present in images captured on the industrial shop floor. Our experiments show that the hybrid model developed in this work outperforms conventional models that struggle with such noise. Furthermore, the collision avoidance algorithm aids the industrial robot in correctly picking objects without causing damage, thereby enhancing overall efficiency. In addition, the scheduling algorithm, which employs a distance metric, helps optimize the transfer of jobs from the workspace, facilitating the industrial robot to complete tasks more quickly.

While the developed system has been proven successful in handling texture-less objects, future work will aim to extend its application to irregular-shaped objects. Currently, the system can manage an average of six objects, processing them in ~40 ms. Further experiments are necessary to evaluate system performance with higher object densities and objects with rapid movements.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Information

The data that support the findings of this study are openly available in GitHub at https://github.com/AshishChouhan85/Dataset.

Author Contribution Statement

Ashish Chouhan: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft. **Shivam Shandilya:** Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft. **Pravanjan Nayak:** Methodology, Validation, Data curation. **Debasish Mishra:** Methodology, Formal analysis, Investigation, Writing – review & editing, Visualization. **Surjya Kanta Pal:** Conceptualization, Resources, Writing – review & editing, Supervision.

References

[1] Zhang, L., Cai, Z., Yan, Y., Yang, C., & Hu, Y. (2024). Multi-agent policy learning-based path planning for autonomous mobile robots. *Engineering Applications of Artificial Intelligence*, *129*, 107631. https://doi.org/10.1016/j.engappai.2023.107631

[2] Saleh, N. I., & Ijab, M. T. (2023). Industrial Revolution 4.0 (IR4.0) readiness among industry players: A systematic litera-

ture review. *Artificial Intelligence and Applications*, *1*(2), 70–85. https://doi.org/10.47852/bonviewAIA2202336

[3] Jiang, Y., Liu, G., Huang, Z., Yang, B., & Yang, W. (2024). Geometry perception and motion planning in robotic assembly based on semantic segmentation and point clouds reconstruction. *Engineering Applications of Artificial Intelligence*, *130*, 107678. https://doi.org/10.1016/j.engappai.2023.107678

[4] Yunardi, R. T., Winarno, & Pujiyanto. (2015). Contour-based object detection in Automatic Sorting System for a parcel boxes. In *International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation*, 38–41. https://doi.org/10.1109/ICAMIMIA.2015.7507998

[5] DeGol, J., Bretl, T., & Hoiem, D. (2017). ChromaTag: A colored marker and fast detection algorithm. In *IEEE International Conference on Computer Vision*, 1481–1490. https://doi.org/10.1109/ICCV.2017.164

[6] Jakubovic, A., & Velagic, J. (2018). Image feature matching and object detection using brute-force matchers. In *International Symposium ELMAR*, 83–86. https://doi.org/10.23919/ELMAR.2018.8534641

[7] Zhao, K., Wang, Y., Zhu, Q., & Zuo, Y. (2022). Intelligent detection of parcels based on improved faster R-CNN. *Applied Sciences*, *12*(14), 7158. https://doi.org/10.3390/app12147158

[8] Kozamernik, N., Zaletelj, J., Košir, A., Šuligoj, F., & Bračun, D. (2023). Visual quality and safety monitoring system for human-robot cooperation. The *International Journal of Advanced Manufacturing Technology*, *128*(1–2), 685–701. https://doi.org/10.1007/s00170-023-11698-2

[9] He, Y., Fukuda, O., Sakaguchi, D., Yamaguchi, N., Okumura, H., & Arai, K. (2020). Development of a practical tool in pick-and-place tasks for human workers. *International Journal of Advanced Computer Science and Applications*, *11*(4), 780–786. https://doi.org/10.14569/IJACSA.2020.01104101

[10] Gao, M., Cai, Q., Zheng, B., Shi, J., Ni, Z., Wang, J., & Lin, H. (2021). A hybrid YOLOv4 and particle filter based robotic arm grabbing system in nonlinear and non-Gaussian environment. *Electronics*, *10*(10), 1140. https://doi.org/10.3390/electronics10101140

[11] Wong, C. C., Chen, C. J., Wong, K. Y., & Feng, H. M. (2023). Implementation of a real-time object pick-and-place system based on a changing strategy for rapidly-exploring random tree. *Sensors*, *23*(10), 4814. https://doi.org/10.3390/s23104814

[12] Kijdech, D., & Vongbunyong, S. (2023). Pick-and-place application using a dual arm collaborative robot and an RGB-D camera with YOLOv5. *IAES International Journal of Robotics and Automation*, *12*(2), 197–210. https://doi.org/10.11591/ijra.v12i2.pp197-210

[13] Yamtuan, K., Radomngam, T., & Prempraneerach, P. (2023). Visual servo kinematic control of delta robot using YOLOv5 algorithm. *Journal of Robotics and Control*, *4*(6), 818–831. https://doi.org/10.18196/jrc.v4i6.19102

[14] Yue, Z., Han, Z., Yang, X., & Liu, L. (2024). DCSPose: A dual-channel Siamese framework for unseen textureless object pose estimation. *Applied Sciences*, *14*(2), 730. https://doi.org/10.3390/app14020730

[15] Miao, D., Wang, Y., Yang, L., & Wei, S. (2023). Foreign object detection method of conveyor belt based on improved nanodet. *IEEE Access*, *11*, 23046–23052. https://doi.org/10.1109/ACCESS.2023.3253624

[16] Li, J., Su, Z., Geng, J., & Yin, Y. (2018). Real-time detection of steel strip surface defects based on improved YOLO detection network. *IFAC-PapersOnLine*, *51*(21), 76–81. https://doi.org/10.1016/j.ifacol.2018.09.412

[17] Xu, Y., Zhang, K., & Wang, L. (2021). Metal surface defect detection using modified YOLO. *Algorithms*, *14*(9), 257. https://doi.org/10.3390/a14090257

[18] Chu, F. J., Xu, R., & Vela, P. A. (2018). Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters*, *3*(4), 3355–3362. https://doi.org/10.1109/LRA.2018.2852777

[19] Shin, H., Hwang, H., Yoon, H., & Lee, S. (2019). Integration of deep learning-based object recognition and robot manipulator for grasping objects. In *16th International Conference on Ubiquitous Robots*, 174–178. https://doi.org/10.1109/URAI.2019.8768650

[20] Upadhyay, R., Asi, A., Nayak, P., Prasad, N., Mishra, D., & Pal, S. K. (2023). Real-time deep learning–based image processing for pose estimation and object localization in autonomous robot applications. The *International Journal of Advanced Manufacturing Technology*, *127*(3–4), 1905–1919. https://doi.org/10.1007/s00170-022-09994-4

[21] Ma, Y., Zhu, W., & Zhou, Y. (2022). Automatic grasping control of mobile robot based on monocular vision. The *International Journal of Advanced Manufacturing Technology*, *121*(3–4), 1785–1798. https://doi.org/10.1007/s00170-022-09438-z

[22] Massa, F., Russell, B. C., & Aubry, M. (2016). Deep exemplar 2D-3D detection by adapting from real to rendered views. In *IEEE Conference on Computer Vision and Pattern Recognition*, 6024–6033. https://doi.org/10.1109/CVPR.2016.648

[23] Zhang, S., & Zhang, Y. (2023). Determination method of stable grasping parameters for irregular sheet sorting. The *International Journal of Advanced Manufacturing Technology*, *128*(5–6), 2075–2085. https://doi.org/10.1007/s00170-023-12052-2

[24] Wang, Z., Xu, Y., He, Q., Fang, Z., Xu, G., & Fu, J. (2020). Grasping pose estimation for SCARA robot based on deep learning of point cloud. The *International Journal of Advanced Manufacturing Technology*, *108*(4), 1217–1231. https://doi.org/10.1007/s00170-020-05257-2

[25] Lee, Y. J., Lee, S. H., & Kim, D. H. (2022). Mechanical parts picking through geometric properties determination using deep learning. *International Journal of Advanced Robotic Systems*, *19*(1), 17298814221074532. https://doi.org/10.1177/17298814221074532

[26] Wang, C. Y., Yeh, I. H., & Liao, H. Y. M. (2023). You only learn one representation: Unified network for multiple tasks. *Journal of Information Science and Engineering*, *39*(3), 691–709. https://doi.org/10.6688/JISE.202305_39(3).0015

[27] Misra D., & Landskape. (2020). Mish: A self regularized non-monotonic activation function. In *British Machine Vision Conference*, 1–14.

[28] Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3–11. https://doi.org/10.1016/j.neunet.2017.12.012

[29] Ansari, M. A., Kurchaniya, D., & Dixit, M. (2017). A comprehensive analysis of image edge detection techniques. *International Journal of Multimedia and Ubiquitous Engineering*, *12*(11), 1–12. https://doi.org/10.14257/ijmue.2017.12.11.01

[30] Kher, H. R., & Thakar, V. K. (2014). Scale invariant feature transform based image matching and registration. In *Fifth International Conference on Signal and Image Processing*, 50–55. https://doi.org/10.1109/ICSIP.2014.12

[31] Muthugnanambika, M., & Padmavathi, S. (2017). Feature detection for color images using SURF. In *4th International Conference on Advanced Computing and Communication Systems*, 1–4. https://doi.org/10.1109/ICACCS.2017.8014572

[32] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, 2564–71. https://doi.org/10.1109/ICCV.2011.6126544

[33] Sun, R., Wu, C., Zhao, X., Zhao, B., & Jiang, Y. (2023). Object recognition and grasping for collaborative robots based on vision. *Sensors*, *24*(1), 195. https://doi.org/10.3390/s24010195

[34] Yoon, J., Han, J., & Nguyen, T. P. (2023). Logistics box recognition in robotic industrial de-palletising procedure with systematic RGB-D image processing supported by multiple deep learning methods. *Engineering Applications of Artificial Intelligence*, *123*, 106311. https://doi.org/10.1016/j.engappai.2023.106311

How to Cite: Chouhan, A., Shandilya, S., Nayak, P., Mishra, D., & Pal, S. K. (2025). Deep Learning System for Object Detection and Collision Free Handling in Industrial Robots. *Artificial Intelligence and Applications*. https://doi.org/10.47852/bonviewAIA52023472