International Journal of Transport Development and Integration
Vol. 9, No. 3, September, 2025, pp. 665-676
Journal homepage: http://iieta.org/journals/ijtdi

Real-Time Object Detection for Forklift Automated Guided Vehicles Using Deep Learning

Ryham Ibrahim Khalil*, Naktal Moiad Edan

Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul 41002, Iraq

Corresponding Author Email: ryham.22csp14@student.uomosul.edu.iq

https://doi.org/10.18280/ijtdi.090319

Received: 26 June 2025
Revised: 14 August 2025
Accepted: 25 August 2025
Available online: 30 September 2025

Keywords:
real-time system, artificial intelligence, deep learning, object detection, Forklift AGV, Raspberry Pi 4, YOLOX

ABSTRACT

Automated Guided Vehicles (AGVs) are increasingly used in industrial and logistics operations for material handling, offering benefits such as reduced human error, improved efficiency, and lower operational costs. This study presents the design and implementation of a real-time intelligent management system for Forklift AGVs based on deep learning techniques. The core of the system is an optimized version of YOLOv3, termed YOLOX, enhanced with Adaptive Spatial Feature Fusion (ASFF) and advanced data augmentation strategies. The ASFF module employs spatially adaptive weights (α, β, γ) to dynamically integrate multi-scale features across the Feature Pyramid Network, improving the detection of small, occluded, and overlapping objects. The system is trained on a combined Pascal VOC dataset using mix-up and label smoothing to enhance generalization and model robustness. It is deployed on embedded hardware, including Raspberry Pi 4, enabling real-time processing of visual data and sensor inputs under various lighting and environmental conditions. Evaluation results indicate that the model achieves a high mean Average Precision (mAP) of 94.17%, with real-time confidence scores reaching 98.1% in natural lighting and 94.3% in dim conditions. The system effectively detects and classifies a wide range of objects—including static, dynamic, small, distant, and partially occluded—in complex scenes. The proposed solution demonstrates robust real-time performance and adaptability, making it suitable for deployment in resource-constrained environments. It offers a scalable and intelligent framework for autonomous AGV navigation, contributing to safer and more efficient material transportation in real-world applications.

1. INTRODUCTION

The development of Automated Guided Vehicles (AGVs) represents a significant technological milestone. AGVs, which are intelligent robots used for automatic material handling in industrial settings and ports [1], have seen their traditional methods overshadowed by modern technologies such as deep learning and advanced sensors. These traditional methods are often time-consuming, error-prone, and unsafe. Improving automation is essential for industrial competitiveness, operational efficiency, and sustainability. AGVs face challenges such as misclassification, low inference speed, and potential collisions [2, 3]. Effective real-time object detection depends on accuracy and speed [4, 5], with many detection networks focusing on accuracy while neglecting computational complexity. Consequently, enhancing AGV performance is a key research area [6].

Recent advances in Convolutional Neural Networks (CNNs) [7, 8] offer promising solutions to complex computer vision problems [9], improving speed and accuracy compared to older methods. However, real-time neural networks require substantial computing power, which poses challenges when integrating them with embedded systems such as Arduino and Raspberry Pi [10]. The RetinaNet model, featuring multiple convolutional layers [11], and the Faster Region Convolutional Neural Networks (RCNN) model, which utilizes a Region Proposal Network for feature acquisition, represent notable object detection approaches [12]. However, these methods often suffer from slow detection speeds.

In contrast, the You Only Look Once (YOLO) model offers rapid inference with single-stage detection [13]. YOLOv3, widely adopted for its efficiency, employs Darknet-53 for direct predictions without needing additional proposal generation. Despite its advancements, YOLOv3 struggles with detecting small, densely distributed, or occluded objects under varying lighting conditions [5, 14]. Real-time communication protocols, such as Web Real-Time Communication (WebRTC), offer low latency but face limitations in certain applications [15, 16]. To address these issues, an improvement to the YOLOv3 model is proposed, along with constructing and applying a smart control architecture for AGVs to enable object detection in autonomous driving environments. The developed model achieves a dual-directional improvement in object detection accuracy and speed during autonomous driving. The system operates in real-time using the Secure Shell (SSH) protocol for communication, ensuring efficient and secure data exchange between the hardware and software components. The key contributions of this study are summarized as follows:

(1) Design and implementation of an integrated hardware-

software AGV system using Raspberry Pi 4 for real-time object detection and autonomous control.

(2) Development of an enhanced YOLOX-based model to improve object detection accuracy in complex environments.

(3) Utilization of mix-up and label smoothing techniques to augment the dataset, improve generalization, and reduce classification errors.

(4) Application of Adaptive Spatial Feature Fusion (ASFF) with adaptive weights (α, β, γ) to improve multi-scale feature integration and small object detection.

(5) Experimental validation of the system's real-time performance in dynamic indoor environments under varied lighting and obstacle scenarios.

The structure of this paper is organized as follows: Section 2 reviews related work, Section 3 presents the methodology, Section 4 describes the monitoring and control system and its integration with the Forklift AGV, Section 5 summarizes the results and evaluation metrics, Section 6 discusses the findings, and Section 7 concludes the paper.

2. LITERATURE REVIEW

Following the challenges outlined in the introduction, several studies have focused on improving object detection models, particularly those based on the YOLO architecture, in the context of autonomous systems. While these studies have achieved measurable improvements in detection accuracy and speed, many of them present limitations that hinder their suitability for real-time deployment on embedded AGV platforms. The following section presents selected studies and critically examines their approaches and relevance to the current work.

In the study [17], an enhanced algorithm for YOLOv3, named YOLO MFE, aimed to resolve the difficulty of extracting features at multiple scales in YOLOv3. This enhancement was achieved through multi-scale normalization combined with the Generalized Intersection over Union (GIOU) loss, aiming to enhance prediction precision. The proposed model was conducted using dataset named Pascal VOC. When comparing the modified model to YOLOv3 using the mAP metric, YOLOv3 recorded a mAP of 81.04%, whereas the improved model reached 83.04%, demonstrating an improvement of 1.66%. Nonetheless, the use of GIOU as a loss function, instead of the traditional box loss function for bounding box localization, did not result in overall better outcomes. In the study [18], to improve the detection of the lower part of the human body and support the autonomous tracking of AGV, a system based on the Single Shot Detector (SSD) model was introduced. The system addressed the issue of poor feature extraction performance in the traditional SSD model due to the use of Visual Geometry Group (VGG16) and the lack of complexity in the training data. An improved model, referred to as Residual Network (ResNet50), in which the input dimensions were increased to 448×448 pixels to improve accuracy. The experimental outcomes showed that improved detection accuracy by 7% and mAP of 85.1% as compared to the baseline model. However, the system still needs to be optimized for performance when operating in environments where the data is diverse or complex, as well as the proposed model is trained to detect only five classes, which impacts efficiency in various scenarios. The study [19] compared YOLOv3, YOLOv4, and YOLOv5 using the Pascal VOC dataset with a fixed input size of 416×416. YOLOv3 achieved the highest accuracy (77%), while YOLOv5 recorded the fastest inference time. Despite minor improvements to YOLOv5, YOLOv3 remained the most accurate. However, the experiments were based on relatively simple detection tasks, without addressing more complex challenges such as occlusion, dense scenes, or small object detection.

In the study [20], a multiple class deep SoRT and G-RCNN were introduced to develop detection and tracking in video streams. The integrated approach achieved a mAP of 80.6% and demonstrated promising performance with respect accuracy and speed. Nonetheless, evaluation primarily relied on positive RoI samples, and the loss function did not sufficiently capture background or ambiguous regions—factors that significantly affect AGV perception in uncontrolled environments. Reference [21] presented a modified YOLOv3 architecture by employing atailored convolutional deep neural networks (CDNNs) and additional layers to enhance small object detection. The modified YOLOv3 model achieved an accuracy of 58.80% mAP, compared to 55.3 and 57.9 for YOLOv3-416 and YOLOv3-608, respectively. However, the image resolution was reduced to 256×256 to accelerate training, which may have resulted in the loss of critical information and a decline in the model effectiveness in recognizing fine-grained details. In reference [22], an enhancement of the SSD model was proposed through integration with Spiking Neural Networks (SNN), aiming to improve the detection of dark objects while reducing computational overhead. Utilizing the VGG16 backbone, the model reached a mAP of 66.01%. Despite this improvement, the method was not evaluated in embedded or real-time scenarios, raising concerns about its suitability for AGV-based deployment. In reference [23], the Tiny YOLOv3, based on DCNN and Darknet-53 as a backbone for feature extraction, was utilized to develop the Vehicle and Pedestrian Detection (VaPD) system for real-time detection using the TensorFlow library on the Google Colab environment, with a Raspberry Pi 4, and the Pascal VOC dataset. A fixed input size of 416 × 416 pixels was used for the images. Concerning the metric of mAP, based on the result obtained at the dataset, the detection system yielded an accuracy 77.5% Therefore, both the accuracies in terms of AP and mAP portrayed high values. However, the VaPD system build based on the Tiny YOLOv3 model has limitation in detection of occluded or overlapping objects because often it misclassified.

Collectively, the reviewed studies demonstrate meaningful progress in object detection; however, most do not fully address the constraints of real-time AGV applications, particularly regarding small object detection, computational efficiency, and hardware compatibility. The objective of this research is to resolve these challenges through development of a lightweight, optimized YOLOv3-based detection system integrated within AGV control architecture, suitable for real-time autonomous operation on embedded platforms.

3. METHODOLOGY

The proposed system was developed through a structured methodology encompassing hardware configuration, software implementation, and full system integration into an AGV prototype. This section outlines the experimental setup and implementation details across both hardware and software domains.

3.1 Hardware configuration

The hardware framework consists of two main controllers:

• Main Controller: A high-performance computing unit configured with an Intel® Core™ i7-11800H (11th Gen) processor @ 2.30 GHz, 32 GB of memory, and 512 GB of NVMe SSD storage, used primarily for object detection and decision-making tasks.

• Sub-controller (Raspberry Pi 4): Responsible for data acquisition and basic control tasks. It is equipped with a Cortex-A72 quad-core processor, 8 GB RAM, and dual-band wireless capability. The Pi Camera module captures 5 MP images and streams real-time video at 1080p/30 FPS to the main controller for further processing. Additional components include:

• Ultrasonic Sensor (HC-SR04): Used for obstacle detection within a range of 2 cm to 400 cm. The sensor is interfaced with the Raspberry Pi GPIO pins and operates through a voltage divider to ensure compatibility with 3.3V logic.

• Relay Modules: Used for controlling actuator responses in the prototype. All components were mounted onto a Forklift AGV robot. This setup enables real-time testing of object detection, classification, and motion control capabilities in a semi-structured environment.

3.2 Software implementation

The software environment was developed using Python 3.11 and OpenCV2 [24], in conjunction with the YOLOX object detection framework. Model training and testing were performed using the Pascal VOC dataset. VOC 2012 includes 11,530 images containing 27,450 annotated objects, and VOC 2007 consists of 9,963 images with a total of 24,640 labeled objects [25]. Both VOCs contain eleven object classes relevant to AGV operations. Input images were resized to 608×608 pixels, and bounding boxes were normalized accordingly. The training process utilized Darknet53 as the backbone network within a CUDA-accelerated PyTorch environment, leveraging pre-trained weights for optimized convergence. The model's effectiveness was assessed using the mAP metric and inference latency to ensure real-time viability on the embedded system.

3.3 System integration

Following hardware and software development, the full system was deployed on the Forklift AGV prototype. This integration facilitated the real-time transfer of data from sensors and cameras to the detection model, allowing the AGV to recognize, classify, and respond to objects and activities autonomously. The systems behavior was evaluated in various indoor scenarios to evaluate detection accuracy, decision speed, and system robustness.

3.4 Object detection

In recent years, deep learning (DL) methods have been widely applied to object detection, as they can extract fine-grained visual features and progressively construct more abstract semantic representations. These DL-based methods enable a hierarchical representation of data, enhancing the object detection process. For multi-classification tasks, DL-based object detection outperforms conventional detection methods in terms of speed, accuracy, and robustness. The ongoing advancement of neural networks using convolutions (CNNs) [8] has significantly advanced object detection, with contemporary deep CNN-based object detectors such as SSD, R-CNN, and YOLO being essential to this advancement [26]. To efficiently extract features, these detectors employ DL methods. Nevertheless, CNNs require constructing their network structure and optimizing weight parameters through training [7, 14]. YOLOv3 is an improved variant of YOLO and YOLOv2. This network directly uses a single feed-forward method to estimate class probabilities and bounding box offsets from complete images CNN, eliminating the need to generate region proposals or sample features [27]. In YOLOv3, the input is divided into grid cells of size S×S. It is the grid cell's responsibility to detect an object when its center point is inside its borders. Every cell in the model computes the object scores associated with B bounding boxes and forecasts the location data for these bounding boxes [28]. For every object, the score is calculated as Eq. (1):

$$Confidence\ Score = \Pr(object) \times IOU_{pred}^{truth} \tag{1}$$

where, $\Pr(object)$ indicates the probability that an item is inside the box, and IOU_{pred}^{truth} is the intersection over union between the anticipated bounding box and the ground truth bounding box. Class prediction involves assigning probabilities to multiple categories to determine the most likely classification of an object detected within an image. Four coordinates are predicted by the model for every bounding box: x, y, w, and h. These coordinates are typically expressed as offsets relative to the grid cell containing the bounding box [27, 28]. The bounding box's expected locations, represented as (b_x, b_y, b_w, b_h), are given in Eqs. (2)-(5) are computed using the sigmoid function to constrain the values between 0 and 1.

$$b_x = \sigma(t_x) + c_x \tag{2}$$

$$b_y = \sigma(t_y) + c_y \tag{3}$$

$$b_w = p_w\, exp^{t_w} \tag{4}$$

$$b_h = p_h\, exp^{t_h} \tag{5}$$

where, (σ) denotes the sigmoid function, (t_x, t_y, t_w, t_h) represents the predicted values, and (c_x, c_y) the coordinates of the top-left corner of the grid cell are the coordinates. (p_w, p_h) refers to the dimensions of the anchor box. Figure 1 shows YLOLv3 architecture.

YOLOX is a single-stage object detection framework that is tailored for real-time applications. Important elements including residual modules, skip connections, and up-sampling procedures are incorporated into its structure. To provide prediction outputs, the model uses tiny 1×1 kernels and only convolutional layers. The detecting head has a kernel dimension of 1×1×255 and is based on the formula (B×(5+C))×1. Compared to previous iterations of the YOLO series, YOLOX offers better detection accuracy while operating at 30 frames per second (FPS) [28, 29]. The YOLOX model's detection pipeline consists of three main steps.

(1) Input Stage: To guarantee compatibility with the ensuing convolutional layers and preserve uniformity in spatial processing across the network, raw input images are resized to 608×608×3 (height×width×channels).

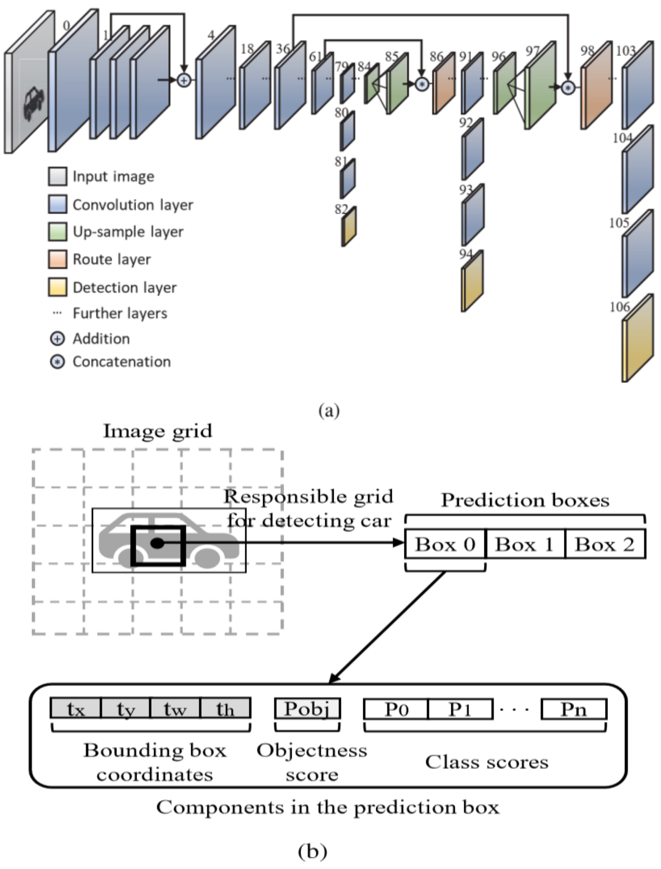


Figure 1. YOLOv3 architecture [30]

(2) Feature Extraction Stage: A deep convolutional network called the Darknet-53 backbone, which is intended to recognize hierarchical visual patterns, receives the scaled images. At 76×76, 38×38, and 19×19 spatial resolutions, this stage generates three unique feature maps that correspond to various receptive fields. From the original RGB channels to the 32, 64, 128, 256, 512, and finally 1024 filters, the feature maps are progressively deepened as the signal moves through the layers. These layers gradually pick up visual characteristics at low and high levels. Convolutional processing, up-sampling, down-sampling, and spatial-weighted feature fusion are all used in the ASFF method [31] to improve contextual representation once the acquired multi-scale features have been fused via an FPN.

(3) Prediction Stage: The detecting head uses the improved feature maps produced by the FPN and ASFF modules to make item predictions. Three anchor-based bounding boxes are produced by the model for every geographic grid cell. Multiple parameters obtained from the fused multi-scale features are encapsulated in each bounding box as Eq. (6):

$$Number\ of\ Filters = (5 + C) \times B \tag{6}$$

where, B indicates the quantity of anchor or boundary boxes that are used in the model, *5* represents the value of predictions for each bounding box $(p_c, b_x, b_y, b_w, b_h)$ and C represents the class probabilities.

3.4.1 Image augmentation

During pre-processing, data augmentation methods were used to improve the model's performance on dataset that were unbalanced. Specifically, image mix-up and label smoothing were utilized to improve the model generalization capacity and promote linear behaviour between training examples. Two examples are chosen at random for the image mix, X_i, Y_i and X_j, Y_j [32], and the creation of a new instance through linear interpolation, by the following Eqs. (7) and (8):

$$\hat{x} = \lambda x_i + (1-\lambda)x_j \tag{7}$$

$$\hat{y} = \lambda y_i + (1-\lambda)y_i \tag{8}$$

In these equations, X_i, Y_i and X_j, Y_j are two randomly selected samples from the training data, and $\lambda \in [0,1]$ is a value drawn from the Beta (β, β) distribution. This newly generated example ($\hat{x}$, $\hat{y}$) is then used in mix-up training. Moreover. Label smoothing regularizes the output distribution by softening the ground truth labels in the training data, thereby enhancing the model generalization ability. This technique introduces controlled noise to the actual class values, limiting the model capacity to overfitting and thus improving its overall classification accuracy [33].

3.4.2 Data augmentation methods

During testing and validation, this method was utilized to improve predictions for cases where the object in the image is too small. The images were resized using a randomly chosen interpolation technique from among the popular methods and then normalized [34].

3.4.3 ASFF

In the standard YOLOv3 model features through the FPN are fused in a top-down fashion to integrate deep and shallow feature information [29]; however, features at different scales remain interdependent and mutually constrained [35]. The ASFF process includes two key steps as follows [36]:

- Feature Resizing

In YOLOv3, for a given feature level l ∈{1,2,3} associated with a feature map X^l, the feature maps from all other levels X^n where n ≠ l, are spatially adjusted to conform to the resolution of *(X^l)*. This alignment is essential to enable effective multi-level feature aggregation. As YOLOv3 features across the three levels differ in resolution and channel numbers, up-sampling and down-sampling techniques are modified accordingly for each scale. followed by an interpolation to upscale the resolution.

- Adaptive Fusion

Concatenation along the channel dimension is performed once the tree-adjusted feature maps have been resized. After concatenation, the feature map is normalization through the soft-max activation function, generating weight vectors *(α), (β)*, and *(γ)*, which are then employed to combine the feature maps adaptively. The representation of the feature vector at spatial location *$(i,\ j)$* on the feature map that was moved from level n to level l is *$(x_{ij}^{n\to l})$*. At level l, the multi-scale feature aggregation can therefore be expressed as Eq. (9) [36, 37]:

$$y_{ij}^{l} = \alpha_{ij}^{l} . x_{ij}^{1\to l} + \beta_{ij}^{l} . x_{ij}^{2\to l} + \gamma_{ij}^{l} . x_{ij}^{3\to l} \tag{9}$$

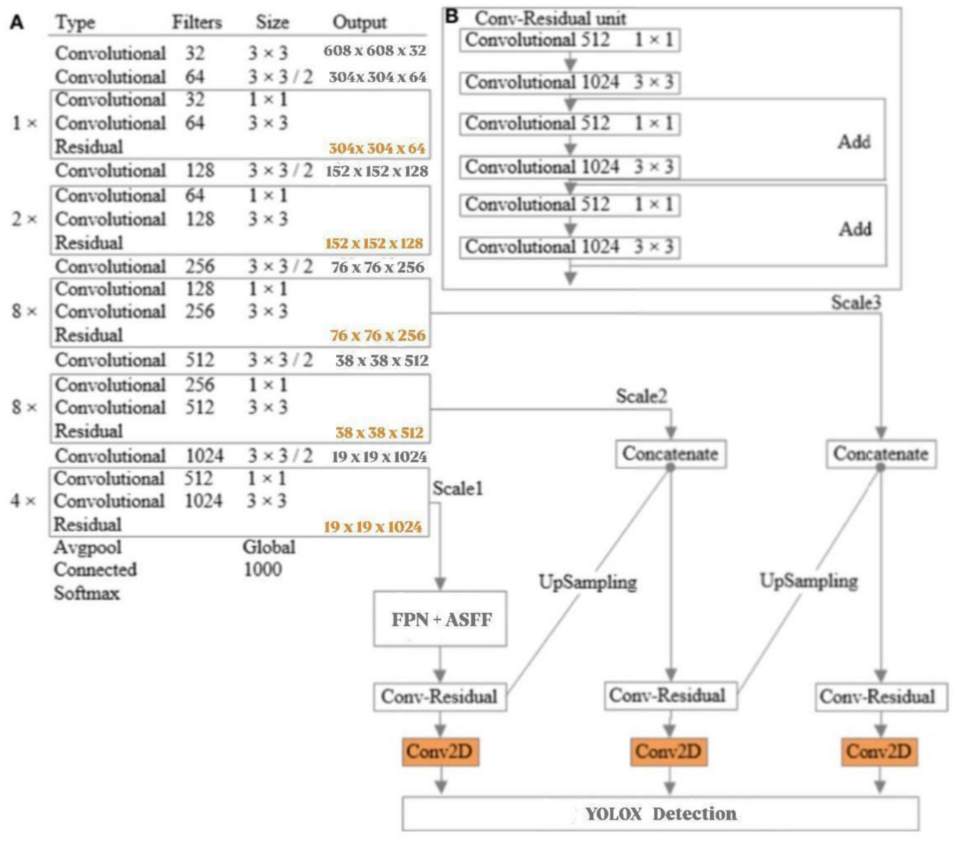
where, *(y_{ij}^l)* indicate the output feature vector at *(i, j)* among channels, and weights *(α_{ij}^l), (β_{ij}^l)*, and *(γ_{ij}^l)* show how important feature mappings are spatially at three levels for level l, adaptively learnt, with shared across channels. Based on previous studies, the constraint *($\alpha_{ij}^l + \beta_{ij}^l + \gamma_{ij}^l = 1$)* is enforced, with values α_{ij}^l, β_{ij}^l, γ_{ij}^l *∈ [0,1]* as Eq. (10) [36, 37]:

$$\alpha_{ij}^{l} = \frac{e^{\lambda_{\alpha_{ij}}^{l}}}{e^{\lambda_{\alpha_{ij}}^{l}} + e^{\lambda_{\beta_{ij}}^{l}} + e^{\lambda_{\gamma_{ij}}^{l}}} \tag{10}$$

where, (α_{ij}^{l}, β_{ij}^{l}) and (γ_{ij}^{l}) are determined using the softmax function, with the control parameters being ($\lambda_{\alpha_{ij}}^{l}$), ($\lambda_{\beta_{ij}}^{l}$), and ($\lambda_{\gamma_{ij}}^{l}$). Utilising 1×1 convolution layers, the weight scalar maps (λ_{α}^{l}), (λ_{β}^{l}), and (λ_{γ}^{l}) are calculated from ($x^{1\to l}$), ($x^{2\to l}$), and ($x^{3\to l}$) and are therefore learnable through standard backpropagation. This approach ensures each pixel in the fused feature map is a weighted average of corresponding pixels in the rescaled maps, uses adaptively learnt weights to improve detection accuracy and better integrate multi-level features. Table 1 and Figure 2 show Darknet-53 with ASFF in YOLOX architecture.

Table 1. YOLOX darknet-53 architecture

Layer	Filters	Size/Stride	Repeat	Output Size
Image	-	-	-	608×608
Conv	32	3×3/1	1	608×608
Conv	64	3×3/2	1	304×304
Conv	32	1×1/1	Conv×1	304×304
Conv	64	3×3/1	Conv×1	304×304
Residual	-	-	Residual×1	304×304
Conv	128	3×3/2	1	152×152
Conv	64	1×1/1	Conv×2	152×152
Conv	128	3×3/1	Conv×2	152×152
Residual	-	-	Residual×2	152×152
Conv	256	3×3/2	1	76×76
Conv	128	1×1/1	Conv×8	76×76
Conv	256	3×3/1	Conv×8	76×76
Residual	-	-	Residual×8	76×76
Conv	512	3×3/2	1	38×38
Conv	256	1×1/1	Conv×8	38×38
Conv	512	3×3/1	Conv×8	38×38
Residual	-	-	Residual×8	38×38
Conv	1024	3×3/2	1	19×19
Conv	512	1×1/1	Conv×4	19×19
Conv	1024	3×3/1	Conv×4	19×19
Residual	-	-	Residual×4	19×19



A	Type	Filters	Size	Output
	Convolutional	32	3 × 3	608 x 608 x 32
	Convolutional	64	3 × 3 / 2	304x 304 x 64
	Convolutional	32	1 × 1	
1 ×	Convolutional	64	3 × 3	
	Residual			304x 304 x 64
	Convolutional	128	3 × 3 / 2	152 x 152 x 128
	Convolutional	64	1 × 1	
2 ×	Convolutional	128	3 × 3	
	Residual			152 x 152 x 128
	Convolutional	256	3 × 3 / 2	76 x 76 x 256
	Convolutional	128	1 × 1	
8 ×	Convolutional	256	3 × 3	
	Residual			76 x 76 x 256
	Convolutional	512	3 × 3 / 2	38 x 38 x 512
	Convolutional	256	1 × 1	
8 ×	Convolutional	512	3 × 3	
	Residual			38 x 38 x 512
	Convolutional	1024	3 × 3 / 2	19 x 19 x 1024
	Convolutional	512	1 × 1	
4 ×	Convolutional	1024	3 × 3	
	Residual			19 x 19 x 1024
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2. YOLOX architecture

4. MONITORING AND CONTROL SYSTEM

The Forklift AGV system relies on a distributed control strategy that ensures synchronized performance between two core processing units. This structure was developed to support real-time navigation, object detection, and responsive decision-making during autonomous movement in dynamic environments [38]. At the operational level, the Sub Controller—implemented using a Raspberry Pi 4—plays a pivotal role in managing sensory data. It continuously captures video via a Pi Camera and collects distance measurements through an ultrasonic sensor. These data streams are prepared for transmission and sent over a secure Wi-Fi connection using the SSH protocol, relying on a static IP address for consistent communication. Once the connection is established, the Sub Controller streams live visual and distance information to the Main Controller without interruption. On the receiving end, the Main Controller, equipped with high processing capabilities, initializes a graphical user interface (GUI) and activates the pretrained YOLOX-based object detection model. Incoming video frames are analyzed at 30 frames per second, enabling accurate and timely object recognition. In parallel, the system interprets real-time distance data from the ultrasonic sensor to assess the proximity of potential obstacles and determine the necessary control actions. System responses follow a tiered logic based on predefined distance thresholds, calibrated through repeated experiments. When the measured distance to an object exceeds the first threshold (D1), the AGV proceeds at its normal speed. If the distance falls between D1 and a lower critical threshold (D2), the system initiates an immediate directional adjustment to avoid potential contact. In scenarios where the object lies within the critical zone (D_object ≤ D2), the AGV halts immediately to avoid collision. The effectiveness of this control design has been demonstrated in three real-world evaluation scenarios. In the first case, the AGV maintained uninterrupted navigation along a path free of obstacles, validating system stability and the absence of false detections. In the second, a moderate-distance obstacle (approximately 120 cm) triggered a successful course adjustment without stopping the vehicle.

This intermediate scenario is graphically illustrated in Figure 3, which highlights the system's detection and responsive adjustment mechanism. The third case represented a critical situation, where the AGV encountered an obstacle within less than 70 cm, prompting an immediate and controlled stop. These evaluations highlight the system's ability to interpret environmental feedback and react appropriately, confirming its suitability for autonomous operation in practical settings.

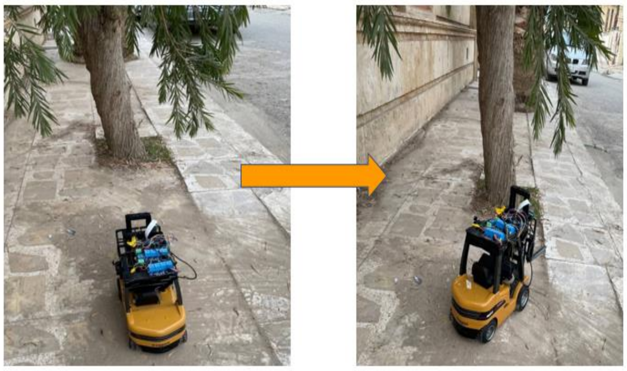


Figure 3. Forklift AGV path adjustment to avoid obstacles

5. RESULTS ANALYSIS AND EVALUATION METRICS

5.1 Experiment and operating environment

To assess the suggested YOLOX-based object identification model effectiveness in relation to AGV navigation, experiments were conducted using a well-structured and reproducible computational environment. The development and training processes were carried out within the Anaconda distribution, utilizing the PyCharm IDE and Python 3.11. The hardware configuration comprised a 13th Generation Intel® Core™ i7-1335U processor operating at 1.30 GHz, accompanied by 16 GB of system memory. The implementation relied on both TensorFlow and PyTorch frameworks to support efficient model construction and training flexibility. The training and testing dataset were generated through the combination of two benchmark datasets: Pascal VOC 2007 and Pascal VOC 2012. Specifically, the VOC 2012 dataset contributed 11,530 labeled images with 27,450 object annotations, while the VOC 2007 dataset provided 9,963 images containing 24,640 labeled objects. This combination resulted in a comprehensive dataset of 21,493 images and 52,090 object instances, representing a substantial and diverse collection of visual scenes. To ensure balanced learning and validation, 30% of the images were used for testing, while the remaining 70% were used for training. The large-scale dataset contributed significantly to improving model generalization and robustness across various object classes and environmental conditions. The training process utilized pre-trained Darknet53 weights as the initialization backbone. A total of 100 epochs were executed, comprising 8,275 iterations, with a 0.001 starting learning rate and a batch size of 16. To enhance convergence, an exponential decay strategy was applied every 20 epochs, reducing the learning rate by a factor of 0.9. 0.0005 weight decay regularization was also employed to prevent overfitting. Input images were resized dynamically between 320×320 and 608×608 pixels in order to reduce overfitting and enhance data variability, the model incorporates Mix-up and ASFF, supporting multi-scale object detection. Standard measures, namely Precision, Recall, and mAP, are used to evaluate performance. The IoU threshold was set to 0.8, ensuring that only predictions with substantial overlap with ground truth were accepted as true positives. An NMS threshold of 0.5 and a confidence score of 0.8 were applied.

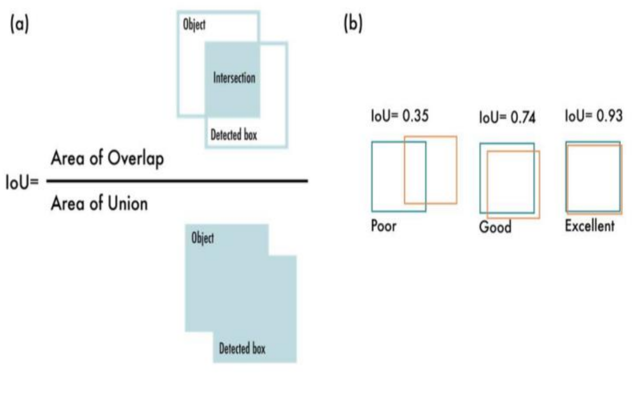


Figure 4. Intersection over union [28]

As illustrated in Figure 4, IoU was used to measure detection accuracy by comparing predicted and ground truth boxes. The high IoU threshold, while reducing false positives, also presents a more rigorous standard, thereby balancing precision and recall in a meaningful manner. Figure 5 shows the loss value (object, box, class).

To examine robustness, the model was evaluated under different conditions, including variable lighting, partial occlusions, and dynamic backgrounds, to detect in real-world environments encountered by AGVs. Results indicate that the model retained high detection accuracy and decision reliability despite such environmental fluctuations, confirming its applicability to autonomous navigation in practical deployment scenarios.

5.2 Evaluation metrics

According to experiments, three metrics, Precision, Recall, and mAP, have been identified to assess the YOLO model's effectiveness in object detection.

Precision: the ratio of True Positive (TP) cases to all positive forecasts, quantifies how accurate positive predictions [34]. It is determined utilizing the Eq. (11):

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

Recall: measured as the percentage of actual positive predictions out of all potential positives, identifying missed positive predictions. It is measured utilizing Eq. (12) [34], where the number of TPs accurately predicted positive samples, while False Positive (FP) tracks incorrectly projected negative samples to be positive. Conversely, False Negatives (FN) tallies incorrectly projected positive samples to be negative. Due to the abundance of irrelevant background regions in images, True Negative (TN) is disregarded in evaluation, as these regions do not affect performance assessment. These metrics are crucial for evaluating classification model performance.

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

Mean Average Precision (mAP): known as the average precision (AP) across all detection categories, computed by averaging the AP values for each class. It provides a comprehensive assessment of model performance, calculated as the sum of AP values for all classes divided by the number of classes (N) in total [34], as Eq. (13):

$$mAP = \frac{1}{N}\sum_{i=0}^{N} AP \tag{13}$$

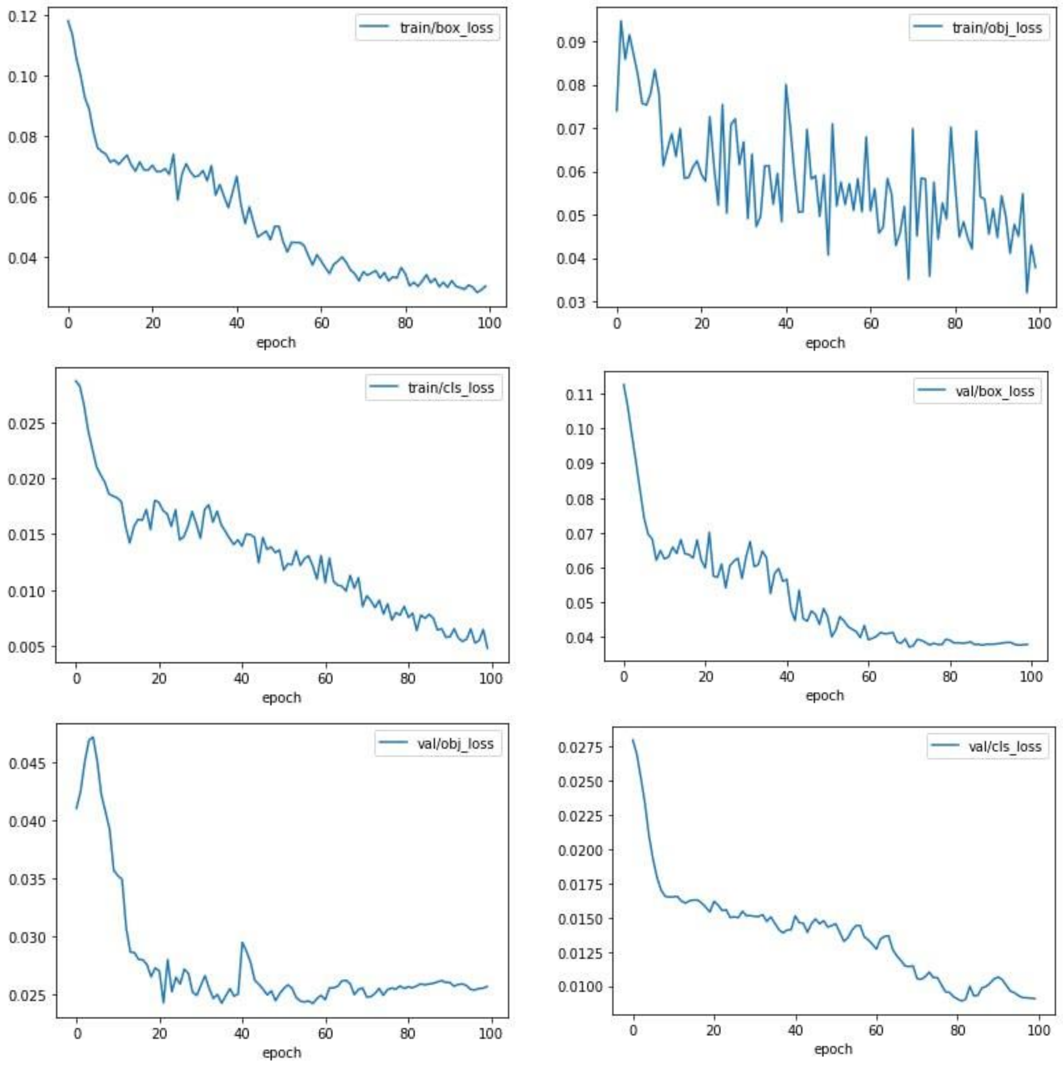


Figure 5. Results training and validation loss object, box, class vs. each epoch

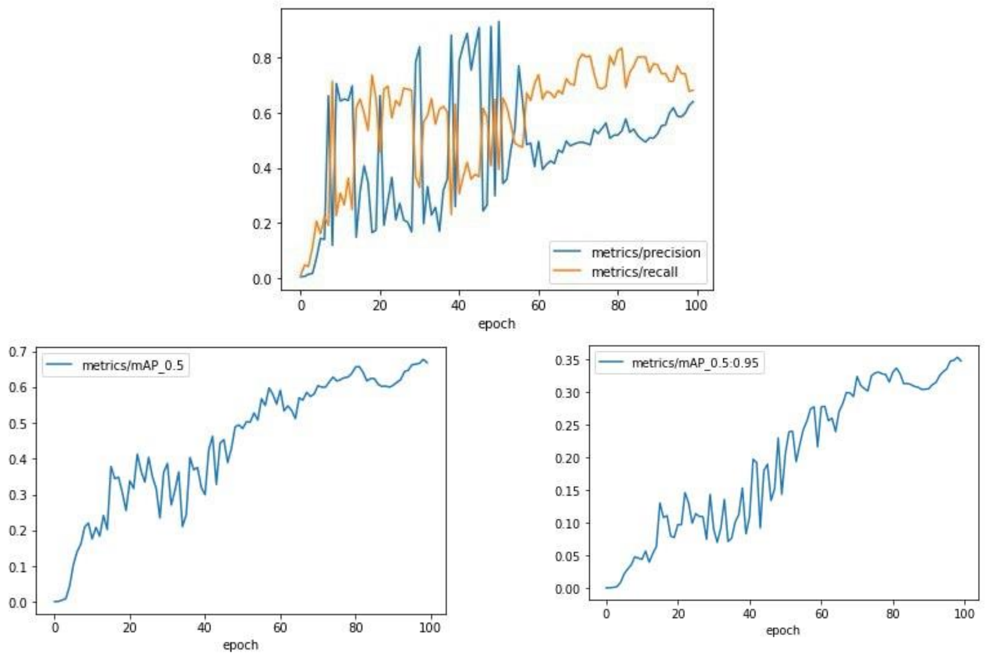


Figure 6. Evaluation metrics Precision, Recall, mAP@0.5, and mAP@0.5: 0.95

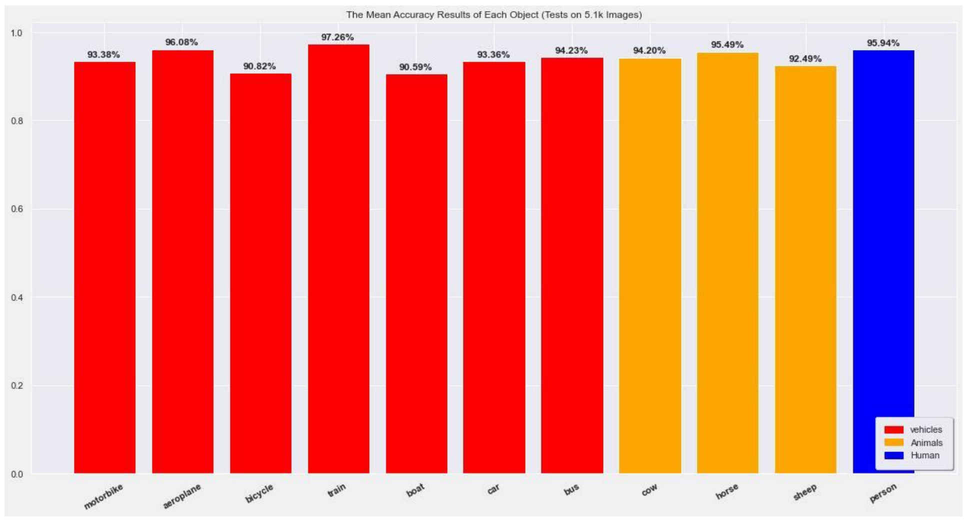


Figure 7. Average precision of dataset classes

The precision-recall curve's area under the curve is represented by this, where AP is computed over recall values at 0 and 1. mAP@0.5:0.95 averages mAP across IoU thresholds from 0.5 to 0.95, whereas mAP@0.5 denotes mAP at an IoU threshold of 0.5 itself. mAP also varies with confidence thresholds. Figure 6 shows the analysis of Recall, Precision, mAP@0.5:0.95 and mAP@0.5 for YOLOv3 on the Pascal VOC validation dataset. The results demonstrate significant improvements in performance metrics. Precision increased from 0.0039 at epoch 1 to 0.6405 at epoch 100, while recall rose from 0.0088 to 0.6815. Both mAP@0.5 and mAP@0.5:0.95 improved from 0.0002 to 0.3472 and from 0.0010 to 0.6685, respectively. The AP values, which indicate precision at various recall levels, showed high accuracy across all classes, as shown in Figure 7.

6. DISCUSSION

The system presented in this work demonstrates a successful optimization of the YOLOv3 model to YOLOX. Although YOLOv3 may exhibit marginally lower accuracy than some advanced detection models, it retains advantages in high real-time detection speed and low computational demands. This balance renders YOLOv3 particularly suited for applications requiring rapid response, even on hardware with limited processing capacity. In order to improve the system's small item detection capabilities, targeted modifications were applied, incorporating image enhancement methods such as mix-up, and label smoothing. These techniques improved the model generalization capacity, enhanced classification accuracy, regulated output distribution, and reduced the risk of overfitting. Further augmentation steps, including interpolation and normalization, were implemented to boost prediction reliability. Structurally, the inclusion of the ASFF was crucial in refining the FPN. ASFF enabled the dynamic integration of spatial information, significantly enhancing feature extraction and representation across diverse image scales. The system was rigorously trained and validated on the Pascal VOC dataset, employing three distinct loss functions to measure prediction errors. This approach led to error reductions between ground-truth and predicted bounding boxes by 0.0879%, 0.0361%, and 0.0239% for the training set, and 0.1088%, 0.0153%, and 0.0188% for the validation sets, respectively. Importantly, these improvements were achieved without overfitting, as illustrated in Figure 5, highlighting the model's robustness. Comparative performance analyses further demonstrate that the optimized system achieved notable improvements over all single-stage detection models on the VOC dataset, which encompassed 11 object categories Table 2. The model's accuracy, measured via mAP, was compared against both single-stage and two-stage detection frameworks. While some models reduce image resolution to increase processing speed, this system maintained a resolution of 608×608 pixels, balancing accuracy with computational efficiency Table 3. Finally, measures including mAP@0.5, recall, accuracy, and mAP@0.5:0.95 were contrasted with advanced YOLO versions, with results presented in Table 4. According to the results, the YOLOX model significantly outperformed the reviewed detection models in Section 2, achieving an mAP score of approximately 94.17%. The deployment of the optimized model on a Forklift AGV robot allowed for a comprehensive evaluation, affirming the efficiency and usefulness of the system in real-world, real-time object identification applications. Figure 8 shows real-time object detection scenarios using the Forklift AGV system.

Table 2. Results of comparison detection for various classes on the VOC dataset

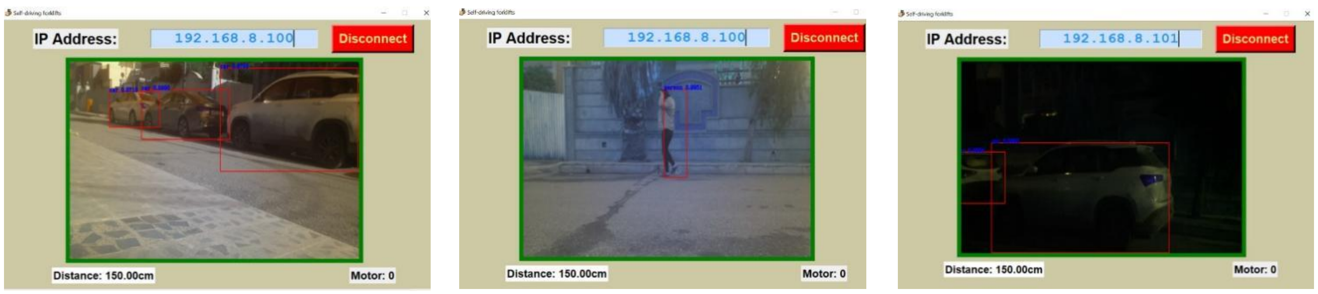
Model	Person	Car	Train	Motorbike	Bus	Bicycle	Airplane	Boat	Cow	Sheep	Hours
YOLOv3 [17]	75.3	65.6	84.5	75.0	82.1	73.2	71.5	74.5	87.9	88.7	55.9
YOLOv4 [17]	51.3	65.5	41.0	75.0	62.1	62.1	83.2	41.0	67.9	58.7	67.6
YOLOv5 [17]	60.5	74.7	72.1	66.9	66.9	73.4	70.6	44.3	42.2	34.9	67.6
YOLOv3 [33]	89.0	92.0	85.0	89.0	95.0	90.0	-	62.0	64.0	66.0	90.0
YOLO-ESFM [39]	88.8	91.0	89.1	90.0	85.7	88.1	89.5	72.9	91.6	86.0	84.8
SSD [20]	70.0	79.6	77.4	74.4	73.1	75.3	70.2	54.5	68.5	66.6	80.0
RFENet-YOLOv8 [4]	89.5	92.9	90.4	91.4	89.8	92.4	90.9	80.8	89.8	84.4	92.5
YOLOX	95.9	93.3	97.2	93.3	94.2	90.8	96.0	90.5	94.2	92.4	95.4

Table 3. Comparison accuracy mAP for detection models on VOC dataset

Model	Input Size	Base Network	Framework	mAP %	Type	Year
Modified YOLOv3 [13]	416×416	Darknet-53	One stage	83.04	Real-time	2023
YOLOv3 [13]	-	Darknet-53	One stage	81.04	-	2023
YOLOv3 [33]	-	Darknet-53	One stage	58.80	Not real-time	2021
SSD (SCOD) [20]	-	VGG16	One stage	66.01	Real-time	2024
YOLOv3 [17]	416×416	Darknet-53	One stage	77.20	Real-time	2022
YOLOv4 [17]	416×416	CSPDDarknet-53	One stage	54.90	Real-time	2022
Fast RCNN [20]	-	VGG16	Two-stage	70.00	Not real-time	2024
RCNN [31]	1000×600	ZFNet	Two-stage	80.50	-	2023
YOLO-ESFM [39]	640×640	Darknet-53	One stage	87.0	Not real-time	2024
RFENet-YOLOv8 [4]	640×640	ResNet-50	One stage	82.9	Not real-time	2025
YOLOX	608×608	Darknet-53	One stage	94.17	Real-time	

Table 4. Comparative metrics for detecting the VOC dataset

Model	mAP@0.5	mAP@0.5:0.95	Precision	Recall	Year
YOLOv5 [40]	45.1	20.9	55.4	48.7	2022
MobileNetv3 YOLOv5s [21]	55.3	32.6	-	-	2024
Ghost-C3M YOLOv5 [40]	44.4	20.8	56.2	47.0	2022
Ghost-C3 YOLOv5 [40]	46.6	21.5	57.0	48.3	2022
MobileNetv3 YOLOv5s [21]	56.1	35.4	-	-	2024
Ghost-C3SE YOLOv5 [40]	45.3	20.8	56.7	48.2	2022
YOLOX	66.8	34.7	64.0	68.1	-



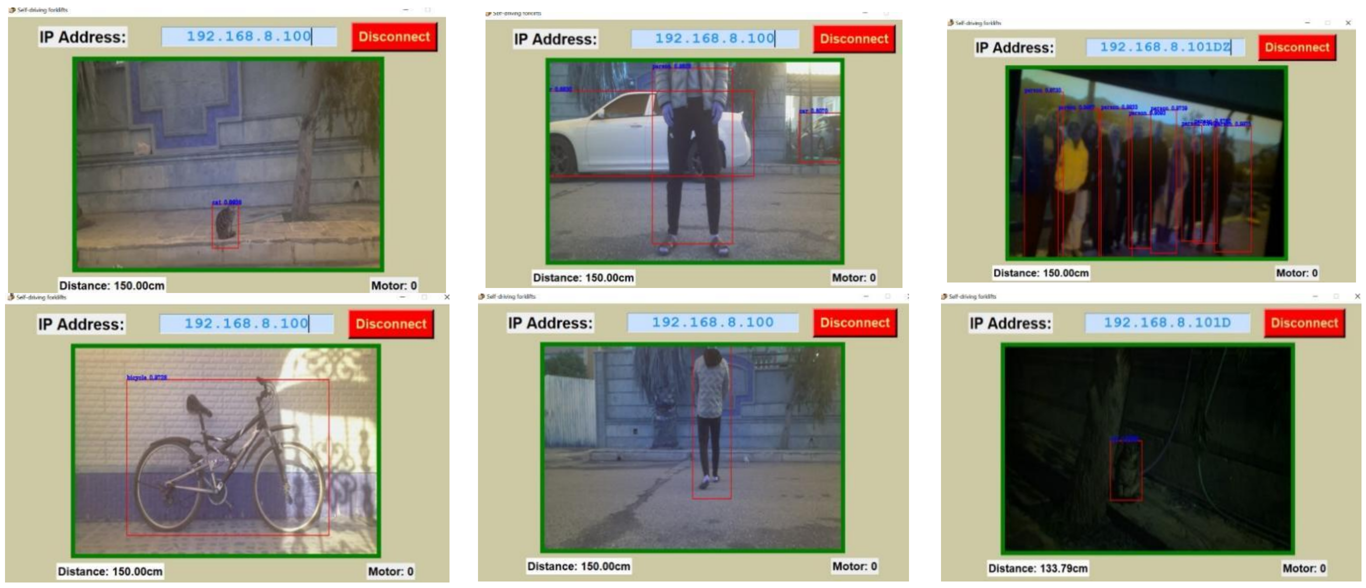


Figure 8. Real-time object detection scenarios using the Forklift AGV system

7. CONCLUSIONS

This study presents a real-time control system for Forklift AGVs that combines deep learning-based object detection with adaptive motion handling. The system integrates an improved YOLOX model supported by ASFF, where spatial weights (α, β, γ) contribute to better feature representation across different scales. Experimental results showed that the system can accurately detect small, distant, overlapping, and partially visible objects under various lighting conditions, achieving a detection precision exceeding 97%. The Forklift AGV demonstrated stable navigation across three obstacle scenarios—clear paths, medium-range objects, and close obstacles—with consistent responses such as stopping or re-routing. The system maintained reliable real-time performance even on resource-constrained hardware, confirming its applicability in industrial environments. However, the current implementation is limited to reactive navigation without predefined path planning within a fixed indoor industrial layout. Future work will focus on extending the system to support multi-AGV coordination and integrating additional sensing capabilities to enable site-level path planning.

ACKNOWLEDGMENT

This work is supported by the University of Mosul, which has provided the necessary academic resources that contributed to the completion of this research.

REFERENCES

[1] Verma, P., Olm, J.M., Suárez, R. (2024). Traffic management of multi-AGV systems by improved dynamic resource reservation. IEEE Access, 12: 19790-19805. https://doi.org/10.1109/ACCESS.2024.3362293

[2] Zhang, D.X., Chen, C., Zhang, G.Y. (2024). AGV path planning based on improved A-star algorithm. In 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, pp. 1590-1595. https://doi.org/10.1109/IAEAC59436.2024.10503919

[3] Clavero, C., Patricio, M.A., García, J., Molina, J.M. (2024). DMZoomNet: Improving object detection using distance information in intralogistics environments. IEEE Transactions on Industrial Informatics, 20(7): 9163-9171. https://doi.org/10.1109/TII.2024.3381795

[4] Li, Z.H., Dong, Y.S. (2025). Refined feature enhancement network for object detection. Complex & Intelligent Systems, 11: 13. https://doi.org/10.1007/s40747-024-01622-w

[5] Zou, Z.X., Chen, K.Y., Shi, Z.W., Guo, Y.H., Ye, J.P. (2023). Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3): 257-276. https://doi.org/10.1109/JPROC.2023.3238524

[6] Cheng, G., Yuan, X., Yao, X.W., Yan, K.B., Zeng, Q.H., Xie, X.X., Han, J.W. (2023). Towards large-scale small object detection: Survey and benchmarks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(11): 13467-13488. https://doi.org/10.1109/TPAMI.2023.3290594

[7] Sulaiman, N., Hasoon, S.O. (2023). Using intelligence techniques to automate Oracle testing. Al-Rafidain Journal of Computer Sciences and Mathematics, 17(1): 91-97. https://doi.org/10.33899/CSMJ.2023.179485

[8] Hamdy, R.A., Younis, M.C. (2023). Performance evaluation of artificial neural network methods based on block machine learning classification. Al-Rafidain Journal of Computer Sciences and Mathematics, 17(2): 111-123. https://doi.org/10.33899/csmj.2023.142250.1079

[9] Du, W.X. (2024). The computer vision simulation of athlete’s wrong actions recognition model based on artificial intelligence. IEEE Access, 12: 6560-6568. https://doi.org/10.1109/ACCESS.2023.3349020

[10] Rosero-Montalvo, P.D., Tözün, P., Hernandez, W. (2024). Optimized CNN architectures benchmarking in hardware-constrained edge devices in IoT environments. IEEE Internet of Things Journal, 11(11): 20357-20366. https://doi.org/10.1109/JIOT.2024.3369607

[11] Mahum, R., Al-Salman, A.S. (2023). Lung-RetinaNet: Lung cancer detection using a RetinaNet with multi-scale feature fusion and context module. IEEE Access, 11:

53850-53861. https://doi.org/10.1109/ACCESS.2023.3281259

[12] Song, P.H., Li, P.T., Dai, L.H., Wang, T., Chen, Z. (2023). Boosting R-CNN: Reweighting R-CNN samples by RPN's error for underwater object detection. Neurocomputing, 530: 150-164. https://doi.org/10.1016/j.neucom.2023.01.088

[13] Diwan, T., Anirudh, G., Tembhurne, J.V. (2023). Object detection using YOLO: Challenges, architectural successors, datasets and applications. Multimedia Tools and Applications, 82(6): 9243-9275. https://doi.org/10.1007/s11042-022-13644-y

[14] Ragab, M.G., Abdulkader, S.J., Muneer, A., Alqushaibi, A., Sumiea, E.H., Qureshi, R., Al-Selwi, S.M., Alhussian, H. (2024). A comprehensive systematic review of YOLO for medical object detection (2018 to 2023). IEEE Access, 12: 57815-57836. https://doi.org/10.1109/ACCESS.2024.3363300

[15] Edan, N., Mahmood, S.A. (2021). Multi-user media streaming service for e-learning based web real-time communication technology. International Journal of Electrical and Computer Engineering, 11(1): 567-574. https://doi.org/10.11591/ijece.v11i1.pp567-574

[16] Edan, N.M., Al-Sherbaz, A., Turner, S. (2017). WebNSM: A novel WebRTC signalling mechanism for one-to-many bi-directional video conferencing. In 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), San Jose, CA, USA, pp. 27-33. https://doi.org/10.1109/CIC.2017.00015

[17] Li, H., Yin, Z., Fan, C., Wang, X. (2023). YOLO-MFE: Towards more accurate object detection using multiscale feature extraction. In Sixth International Conference on Intelligent Computing, Communication, and Devices (ICCD 2023), Hong Kong, China, pp. 143-152. https://doi.org/10.1117/12.2682803

[18] Gao, X.B., Xu, J.H., Luo, C., Zhou, J., Huang, P.L., Deng, J.X. (2022). Detection of lower body for AGV based on SSD algorithm with ResNet. Sensors, 22(5): 2008. https://doi.org/10.3390/s22052008

[19] Francies, M.L., Ata, M.M., Mohamed, M.A. (2022). A robust multiclass 3D object recognition based on modern YOLO deep learning algorithms. Concurrency and Computation: Practice and Experience, 34(1): e6517. https://doi.org/10.1002/cpe.6517

[20] Pramanik, A., Pal, S.K., Maiti, J., Mitra, P. (2021). Granulated RCNN and multi-class deep sort for multi-object detection and tracking. IEEE Transactions on Emerging Topics in Computational Intelligence, 6(1): 171-181. https://doi.org/10.1109/TETCI.2020.3041019

[21] Santhanalakshmi, S.T., Khilar, R. (2023). A custom deep convolutional neural network CDNN-(with YOLO v3 based newly constructed backbone) for multiple object detection. Journal of Data Acquisition and Processing, 38(3): 1511.

[22] Ali, M., Yin, B., Bilal, H., Kumar, A., Shaikh, A.M., Rohra, A. (2024). Advanced efficient strategy for detection of dark objects based on spiking network with multi-box detection. Multimedia Tools and Applications, 83(12): 36307-36327. https://doi.org/10.1007/s11042-023-16852-2

[23] Falaschetti, L., Manoni, L., Palma, L., Pierleoni, P., Turchetti, C. (2024). Embedded real-time vehicle and pedestrian detection using a compressed tiny YOLO v3 architecture. IEEE Transactions on Intelligent Transportation Systems, 25(12): 19399-19414. https://doi.org/10.1109/TITS.2024.3447453

[24] Pulipalupula, M., Patlola, S., Nayaki, M., Yadlapati, M., Das, J., Sanjeeva Reddy, B.S. (2023). Object detection using You Only Look Once (YOLO) algorithm in Convolution Neural Network (CNN). In 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), Lonavla, India, pp. 1-4. https://doi.org/10.1109/I2CT57861.2023.10126213

[25] TensorFlow. (2022). VOC. https://www.tensorflow.org/datasets/catalog/voc#voc2007defaultconfig.

[26] Alsultan, O.K.T., Mohammad, M.T. (2023). A deep learning-based assistive system for the visually impaired using YOLO-V7. Revue d'Intelligence Artificielle, 37(4): 901-906. https://doi.org/10.18280/ria.370409

[27] Gu, H., Zhu, K., Strauss, A., Shi, Y., Sumarac, D., Cao, M. (2024). Rapid and accurate identification of concrete surface cracks via a lightweight & efficient YOLOv3 algorithm. Structural Durability & Health Monitoring, 18(4): 363-380. https://doi.org/10.32604/sdhm.2024.042388

[28] Terven, J., Córdova-Esparza, D.M., Romero-González, J.A. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4): 1680-1716. https://doi.org/10.3390/make5040083

[29] Salim, H., Mustafa, F.S. (2024). A comprehensive evaluation of YOLOv5s and YOLOv5m for document layout analysis. European Journal of Interdisciplinary Research and Development, 23: 21-33.

[30] Choi, J., Chun, D., Kim, H., Lee, H.J. (2019). Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. In 2019 IEEE/CVF International Conference on Computer Vision, Seoul, Korea (South), pp. 502-511. https://doi.org/10.1109/ICCV.2019.00059

[31] Liu, S., Huang, D., Wang, Y. (2019). Learning spatial fusion for single-shot object detection. arXiv preprint arXiv:1911.09516. https://doi.org/10.48550/arXiv.1911.09516

[32] Xu, M., Yoon, S., Fuentes, A., Park, D.S. (2023). A comprehensive survey of image augmentation techniques for deep learning. Pattern Recognition, 137: 109347. https://doi.org/10.1016/j.patcog.2023.109347

[33] Shen, L., Yu, J., Yang, H., Kwok, J.T. (2024). Mixup augmentation with multiple interpolations. arXiv preprint arXiv:2406.01417. https://doi.org/10.48550/arXiv.2406.01417

[34] Mohammed, E.A., Ali, A.J., Abdullah, A.M. (2024). Artificial intelligence-based helipad detection with convolutional neural network. NTU Journal of Engineering and Technology, 3(1): 18-25. https://doi.org/10.56286/ntujet.v3i1.799

[35] Wang, M., Li, K., Zhu, X., Zhao, Y. (2022). Detection of surface defects on railway tracks based on deep learning. IEEE Access, 10: 126451-126465. https://doi.org/10.1109/ACCESS.2022.3224594

[36] Liu, H., Du, J., Zhang, Y., Zhang, H. (2022). Performance analysis of different DCNN models in remote sensing image object detection. EURASIP Journal on Image and Video Processing, 2022(1): 9. https://doi.org/10.1186/s13640-022-00586-6

[37] Khalil, R.I., Edan, N.M. (2025). Development of automated guided vehicles using software engineering. In 2025 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, pp. 20-27. https://doi.org/10.1109/CSASE63707.2025.11054012

[38] Abdullah, D.B., Abood, I.N. (2023). Real time system scheduling approach: Survey. AL-Rafidain Journal of Computer Sciences and Mathematics, 17(1): 43-51. https://doi.org/10.33899/csmj.2023.179512

[39] Yan, F., Chen, K., Cheng, E., Qu, P., Ma, J. (2024). YOLO-ESFM: A multi-scale YOLO algorithm for sea surface object detection. ResearchGate. https://doi.org/10.21203/rs.3.rs-4623645/v1

[40] Wu, L., Zhang, L., Shi, J., Zhang, Y., Wan, J. (2022). Damage detection of grotto murals based on lightweight neural network. Computers and Electrical Engineering, 102: 108237. https://doi.org/10.1016/j.compeleceng.2022.108237

NOMENCLATURE

AGV	Automated Guided Vehicle
AP	Average precision for each class
mAP	Mean average precision across all classes
mAP@0.5	Mean average precision at IoU threshold of 0.5
mAP@0.5: 0.95	Mean average precision at IoU threshold of 0.5 to 0.95
IoU	Intersection over union
NMS	Non-maximum suppression at IoU thresholds 0.3 to 0.7
TP	True positive count
FP	False positive count
FN	False negative count
FPS	Frames per second
D1, D2	Distance thresholds for AGV control
ASFF	Adaptive spatial fusion features
FPN	Feature pyramid network

Greek symbols

l	Learning rate for model training
λ	Mix up parameter
σ	Sigmoid activation function
α	Weight for low-level spatial features
β	Weight for mid-level contextual features
γ	Weight for high-level semantic features

Subscripts

obj	Parameter to detection objects
det	Detection layer
cls	Classification layer