Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Generalized Simulation Study of Autonomous Obstacle Avoidance Based on Game Non-Player Character

Wenqian Shang[1,2*], Jinru Yang[1], Wenfeng Hu[2], Tong Yi[3], Linlin Zhang[1], Dunxing Liu[2]

[1] School of Software, Xinjiang University, Urumqi, 830046, China
[2] State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing, 100024, China
[3] The Key Lab of Education Blockchain and Intelligent Technology, Ministry of Education, Guangxi Normal University, Guilin, 541004, China
E-mail: shangwenqian@cuc.edu.cn

Received: 21 June 2025; **Revised:** 4 August 2025; **Accepted:** 8 September 2025

Abstract: With the rapid development of artificial intelligence technology, unmanned delivery services are increasingly used in restaurants, hotels and other scenes. In this study, for the autonomous obstacle avoidance problem in unmanned delivery service, a generalized simulation experiment method for autonomous obstacle avoidance based on game Non-Player Character (NPC) is proposed. Through simulation research and experimental verification, the obstacle avoidance strategy of the game NPC is innovatively applied to simulate and optimize the obstacle avoidance behavior of unmanned delivery robots in complex environments. This study uses an optimized You Only Look Once v5 (YOLOv5) model as the core of NPC visual processing, increasing processing speed to 21.7 ms. Compared with benchmark models such as YOLOv5, this model significantly improves the efficiency of simulation experiments while maintaining high recognition precision. Based on this model, this study designs and implements a series of efficient decision-making mechanisms to ensure that NPCs can effectively avoid obstacles in simulated scenarios. In addition, this study developed a path planning and decision-making system suitable for unmanned delivery robots, ensuring that robots can operate safely and efficiently in complex environments. The research results provide a solid technical foundation for the continuous improvement and widespread promotion of unmanned delivery services. They also promote the research and application of autonomous obstacle avoidance technology based on game NPCs in the field of unmanned delivery, laying an important theoretical foundation for the development of unmanned delivery technology.

***Keywords*:** Non-Player Character (NPC), You Only Look Once v5 (YOLOv5), decision algorithm, obstacle avoidance

MSC: 68T20, 68T40, 68T45

1. Introduction

In today's era, the rapid development of artificial intelligence technology has profoundly changed our lifestyle and work mode [1–3]. Especially in the service industry, the application of Artificial Intelligence (AI) is gradually subverting the traditional service model, bringing unprecedented convenience to consumers. Unmanned delivery services (During the delivery process, automated equipment such as driverless vehicles is used to complete delivery tasks through autonomous

DOI: https://doi.org/10.37256/cm.6520257549

operation), as a product of this change, provide innovative solutions to solve problems such as long queues, inefficient delivery, and touchless delivery in hotels. Whether in campus cafeterias or other scenarios, unmanned delivery services have become the key to improving service quality and efficiency [4, 5]. Therefore, exploring the technological innovation of unmanned delivery services in campus cafeterias, hotel delivery and other scenarios not only has practical application value, but also conforms to the development trend of modern service industry.

This study focuses on obstacle avoidance, a key issue in unmanned delivery services [6]. In complex dynamic environments (e.g., campus cafeterias), unmanned delivery robots need to face challenges such as dense crowds and diverse obstacles, which puts higher demands on the robot's perception, decision-making, and execution capabilities. To simulate this process, this study innovatively adopts the obstacle avoidance technique of game Non-Player Character (NPC) to simulate the obstacle avoidance behavior of unmanned delivery robots. Although the game industry has made significant progress in terms of graphical expressiveness, the realism and intelligence of NPCs are still insufficient [7]. Traditional NPCs usually rely on preset routes or behavioral trees to achieve obstacle avoidance and lack the randomness and responsiveness of real creatures. For this reason, this study introduces computer vision technology to endow NPCs with visual perception and decision-making ability closer to humans.

Computer vision technology aims to equip computers with human-like visual capabilities to extract information from images and use it for specific purposes [8]. By applying computer vision technology to game NPCs, this study realizes autonomous obstacle avoidance for NPCs in virtual scenes. Although the application of computer vision in games is not yet common, the concept is similar to autonomous driving technology in virtual worlds [9], except that it requires less safety. In this study, we use the Unreal 4 game engine [10] to build a virtual scene and the optimized YOLOv5 model [11] for obstacle recognition and decision making.

The main contributions of this study are as follows:

1. Integrate computer vision technology into game development, realize virtual life with computer vision as the "eye", and promote computer vision to be part of game implementation.

2. The benchmark YOLOv5 model is optimized for computational speed requirements to improve computational efficiency without significantly affecting performance.

3. The decision-making mechanism of NPC is designed and implemented, including the three links of distance measurement, result calculation and result transmission, and its feasibility is verified through experiments.

Through these studies, we not only provide new perspectives to enhance the realism of game NPCs, but also provide a reference for simulation studies on obstacle avoidance in unmanned delivery services.

This paper is organized as follows: Section 2 systematically reviews key technologies such as NPC intelligent design, YOLO model optimization, and attention mechanisms, providing a theoretical foundation for the research. Section 3 focuses on model optimization, proposing a Distance-Weighted Confidence metric (DWC) to address the issue of low confidence in detecting distant objects, and achieving a balance between speed and precision by introducing the Mobile One module and Convolutional Block Attention Module (CBAM) attention mechanism (inference time optimized from 15.1 ms to 15.6 ms). Section 4 designs a vision-based obstacle avoidance decision-making system, including ranging algorithms, dynamic path planning, and game engine interaction mechanisms. Section 5 validates the system's effectiveness through simulation experiments, demonstrating NPCs' real-time obstacle avoidance capabilities in complex scenarios. Finally, Section 6 summarizes the research contributions and outlines future directions, providing technical support for the intelligent deployment of unmanned delivery services.

2. Related work

2.1 *NPC design*

Marek and other researchers explored ways to enhance the intelligent performance of Non-Player Characters (NPCs) [12]. They comprehensively analyzed four mainstream NPC intelligence design strategies, including decision trees [13], genetic algorithms, reinforcement learning [14], and a hybrid strategy of genetic algorithms and reinforcement learning. It was found that the hybrid strategy performed the best in terms of comprehensive performance, which was not only

effective but also had high operational stability. However, the study did not address the field of computer vision and the experiments focused on the behavioral patterns of NPCs rather than visual recognition capabilities. The researcher points out that even in an optimal simulation environment, the behavior of NPCs is a compromise between real humans and virtual characters, and its purpose is to optimize character behavior in games to make it close to reality rather than an exact replica. This may be one of the reasons why computer vision techniques are not widely used for NPC implementation in current game development. In terms of specific implementation techniques for NPC, a variety of approaches have been developed. Research has shown that current popular techniques include path planning, state machines, affective models, behavioral learning, goal planning, decision trees, perceptual models, and group intelligence [15–19]. Among them, path planning techniques are particularly prevalent, which centers on finding the optimal path from the starting point to the end point [20]. In game design, path planning is a fundamental technique for NPC design, and its role is similar to that of GPS, which provides NPCs with the shortest path to the target location. For example, researchers such as P. Mirowski used path planning techniques in mazes to help the blob achieve navigation [21].

Although techniques such as path planning have been widely used in NPC design, these methods usually rely on preset rules or environmental information, and lack the ability to perceive and adapt to the dynamic environment in real time. In particular, the application of computer vision techniques in NPC design is still in its infancy, which provides an important innovative space for the research in this paper. By introducing computer vision technology into NPC design, this study aims to enhance the autonomous obstacle avoidance ability of NPC in dynamic environments and provide a new solution to the obstacle avoidance problem in unmanned delivery services.

2.2 *YOLO model*

You Only Look Once (YOLO) is an innovative single-stage target detection algorithm proposed by Joseph Redmon and other researchers [22, 23]. Unlike traditional two-stage target detection algorithms, YOLO transforms the detection task into a regression problem and makes predictions directly based on the whole image. Specifically, YOLO partitions the image into multiple cells, each of which predicts the confidence and category probability of the surrounding bounding box. During the training process, YOLO improves the detection performance by comparing the differences between the predicted and real boxes and optimizing the model parameters in combination with the classification and confidence loss. Thanks to its single-stage design, YOLO not only has excellent real-time processing capability, but also maintains high detection precision, and thus has been widely used in real-time demanding scenarios such as autonomous driving.

Since the introduction of YOLO, its development team has continuously released new versions, including YOLOv2, YOLOv3, and YOLOv4, with each generation improving in performance and efficiency [24–26]. On this basis, YOLOv5 further realizes the lightweighting of the model and achieves a good balance between model size and performance by streamlining or optimizing modules that have less impact on performance. This lightweight design makes YOLOv5 more convenient to deploy, which has led to a wide range of applications in various fields such as security surveillance, traffic and urban planning, sports analysis, etc. YOLOv5 not only inherits the advantages of its predecessor, but also significantly improves the detection precision and applicability, which further strengthens the leading position of YOLO series in the field of target detection [27, 28].

Following YOLOv5, the YOLO series of models underwent rapid iteration, with successive releases of YOLOv6, v7, v8, v9, v10, and v11. [29–31]. These new versions typically adopt innovative architectural designs, such as the RepVGG-style reparameterization technique used in YOLOv6 and v7, the anchor-free detection mechanism implemented in YOLOv8, the bidirectional connection structure proposed in YOLOv9, and the efficient modules developed in YOLOv10 and v11. At the same time, through optimization on large-scale benchmark datasets such as Common Objects in Context (COCO), the detection precision, inference speed, and generalization performance of the models are continuously improved [32]. However, the selection of a base model for a specific application scenario depends on the unique requirements of the target task and the expected model modification plan. The main objective of this study is to develop an efficient real-time obstacle detection system for specific gaming environments, requiring extremely low inference latency and ease of deployment. YOLOv5 was selected as the base model based on the following core considerations:

1. Mature modular implementation: Its well-developed, fully documented modular design based on PyTorch significantly lowers the technical threshold for integrating custom lightweight components.

2. Speed-precision balance: Provides excellent baseline performance on resource-constrained platforms, highly compatible with the real-time requirements of game NPC navigation tasks.

3. Stability during development: During the launch and core development stages of this study, YOLOv5 was the most stable, widely used, and community-supported version of the YOLO series, providing a reliable foundation for our targeted optimization.

Although the new version of YOLO has made significant progress, this study focuses on verifying the effectiveness of the proposed lightweight improvement scheme (YOLO-Mobile One Convolutional Attention (MOCA)) under the stringent requirements of game scenarios, which is implemented based on a proven and adaptable framework.

2.3 *Attention mechanism*

Attention mechanism is a classic research in the field of artificial intelligence, which aims to help models focus on important information that was easily ignored in the past [33]. After years of development, attention mechanisms have been widely used in the field of computer vision, mainly including channel attention mechanisms and spatial attention mechanisms.

2.3.1 *Channel attention mechanism*

In the feature map extraction process, not all channels contribute equally to the model. The channel attention mechanism enhances the feature representation by calculating the importance of each channel and boosting the weights of key channels.The Squeeze-and-Excitation (SE) module in SeNet is a classical implementation of the channel attention mechanism, which significantly improves the model performance by dynamically adjusting the channel weights [34]. In addition, Jiahui et al. proposed a dual-channel multiscale method based on the channel attention mechanism, which effectively aggregates the multiscale high-dimensional global features of each layer of the Convolutional Neural Network (CNN) by enhancing the Transformer's ability to extract the deep detail information, and achieved breakthroughs in the field of pedestrian re-identification [35].

2.3.2 *Spatial attention mechanism*

The spatial attention mechanism is similar to the channel attention mechanism, but it focuses on the importance of different regions in the image. By allowing the model to focus on important regions, the spatial attention mechanism can effectively improve the model's ability to capture key information. For example, the Spatial Transformer module enables neural networks to automatically focus on different regions of the input image by learning spatial transformations [36]. Yong et al. used the spatial attention mechanism to refine the spatial information of the feature maps, which significantly enhanced the performance of the model [37].

3. YOLOv5 model optimization

3.1 *Dataset and evaluation metrics*

This section describes the construction process of the dataset used for the experiment. In the preparation phase of the experiment, we found that the models trained on existing datasets (e.g., CoCo128) could not accurately recognize the objects in the game scene. Therefore, we decided to construct a homemade dataset suitable for game scenes.

3.1.1 *Data collection and labeling*

We first set up the data collection environment by randomly arranging the objects needed for the experiment in the game scene and paying attention to the sparse and dense matching between the objects (as shown in Figure 1). After the scene was set up, about 6,000 images were collected by operating the character to move around the scene and capturing images periodically. Subsequently, we screened these images, removed images that were too similar or too densely placed

objects, and finally retained 2,000 images for labeling. Due to the low impact of distant objects on obstacle avoidance, we only labeled near objects.

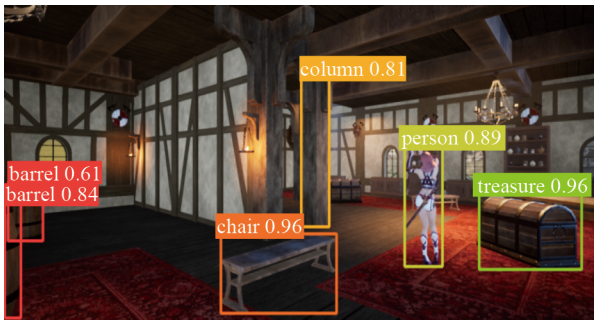


Figure 1. Data collection scenarios

3.1.2 *Dataset segmentation and training*

During the training process, we divided the dataset into training and test sets in the ratio of 4 : 1. After the training was completed, we performed test experiments on the model. Figure 2 shows the results of the detection using the homemade dataset. As can be seen from the figure, all objects in the near distance are correctly recognized, while objects in the far distance are not detected, which is consistent with the labeling of the dataset. Based on these results, we finally decided to use this dataset as the experimental data.

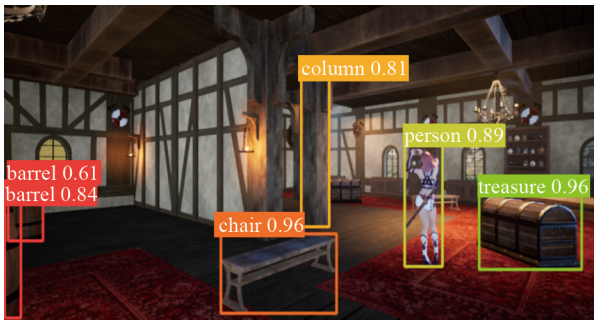


Figure 2. Homemade dataset test results

3.1.3 *Evaluation metrics*

To validate the performance of the improved YOLO model, this experiment will evaluate it based on metrics such as Precision (P), Recall (R), mean Average Precision (mAP), and computational speed. Precision is defined as the ratio of True Positives (TP) to the sum of true positives and False Positives (FP), i.e., the ratio of the number of correctly identified objects in the game scene to the sum of incorrectly identified objects and correctly identified objects; recall is defined as the ratio of true positives to the sum of true positives and False Negatives (FN); mAP is the average of the average precision across all categories. True positive rate refers to the proportion of correctly identified objects, false positive rate refers to the proportion of incorrectly identified objects, and false negative rate refers to the proportion of objects that the model failed to detect.

Precision is defined as follows:

$$\text{Precision} = \frac{TP}{TP+FP}. \tag{1}$$

The recall rate is defined as follows:

$$\text{Recall} = \frac{TP}{TP+FN}. \tag{2}$$

The average precision is defined as follows:

$$AP = \int_0^1 P(R). \tag{3}$$

The average precision mean is defined as follows:

$$mAP = \frac{1}{c}\sum_{j=1}^{c} AP_j. \tag{4}$$

In the field of object detection, confidence is a core concept in the YOLO series of models, directly reflecting the reliability of the model's detection results. However, traditional confidence assessment does not take into account the impact of target distance on detection quality. Objects at long distances, due to lower resolution and blurred features, typically result in lower confidence scores. If the average confidence score is directly used as an overall performance metric, the low confidence detection results of distant objects will significantly reduce the overall performance. To address this issue, this paper proposes a Distance-Weighted Confidence (DWC) metric to scientifically quantify the detection stability of the model at different distances:

$$\text{DWC} = \frac{\sum_{i=1}^{N} w_i \cdot c_i}{\sum_{i=1}^{N} w_i}, \; w_i = \frac{1}{d_i^2}, \tag{5}$$

where c_i is the confidence level of the *i-th* detection box, and d_i is the distance between the detection box and the camera. By assigning lower weights to distant objects (w_i is inversely proportional to d_i^2), DWC effectively balances the contributions of high-precision detection at close range and low-precision detection at long range, thereby avoiding the excessive impact of failed long-range detection on the overall evaluation.

3.2 *Latency optimization*

According to the experimental requirements, the real-time preformance of the model is one of the core objectives of this research. To improve the inference speed, model lightweighting is an effective solution. Lightweighting achieves the effect of fewer parameters, lower memory, and faster speed by reducing the number and size of model parameters.

3.2.1 *Optimization solutions*

Traditional methods usually improve the performance by increasing the model complexity, but this increases the inference time. In order to reduce the computation time, we choose to optimize the model structure. The residual branching structure in YOLO needs to wait for all branches to complete the computation before moving to the next module, which

leads to longer computation time. Therefore, reducing the number of branches and reconstructing the parameters is the key to improve the inference speed.

To this end, we introduce a lightweight module Mobile One in the backbone network [22]. The structure of the Mobile One module is borrowed from MobileNet-V1, which mainly consists of a 3×3 depthwise convolution and a 1×1 pointwise convolution (as Figure 3 shows). Its training process is divided into two parts:

1. Channel-by-channel convolution: each convolution kernel is responsible for one channel, the number of channels of the feature map is the same as the input, and each channel is convolved independently during the computation.

2. Point-by-point convolution: the results of the channel-by-channel convolution are blended using a $1 \times 1 \times M$ convolution kernel to generate a new feature map.

Mobile One uses a structural reparameterization technique to reduce parameter size and runtime by adding parallel branches for 3×3 channel-by-channel convolution and 1×1 point-by-point convolution during training, and then compressing multiple branches into a single branch. This process is shown in Figure 3.

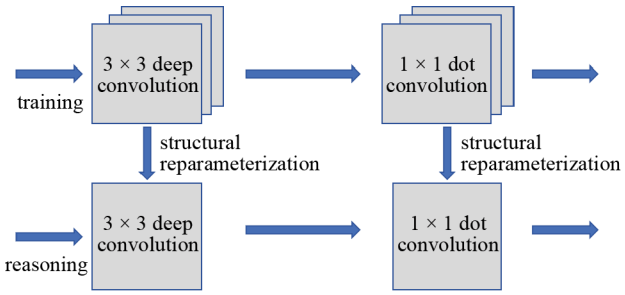


Figure 3. Schematic diagram of Mobile One module

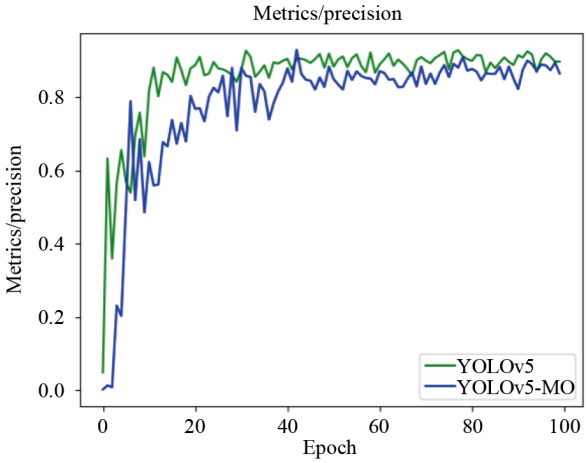
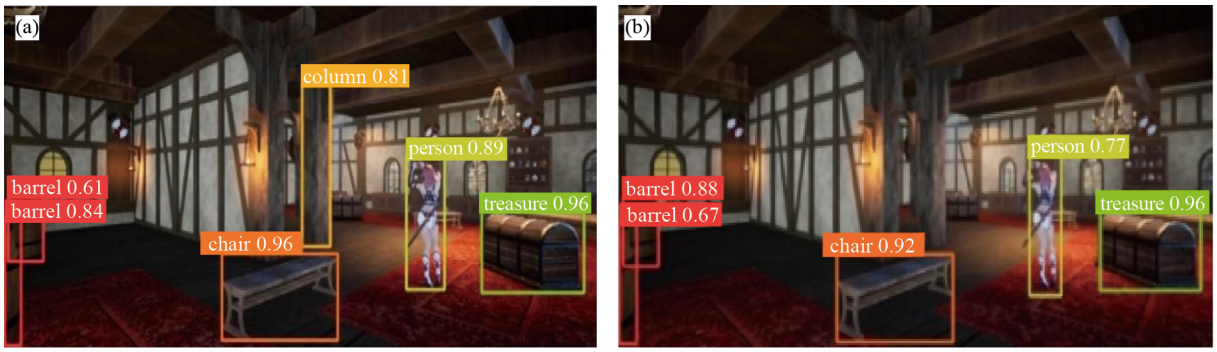


Figure 4. Graph of the first experiment

3.2.2 *Optimization experiments*

We introduce the Mobile One module in the benchmark model and conduct comparison experiments with the base YOLOv5 model. The experimental results in terms of precision (precision) are shown in Figure 4. From the figure, it can be seen that the model (hereinafter referred to as YOLO-Mobile One (MO)) after the introduction of the Mobile One module has a significant decrease in precision, and other performance metrics (e.g., recall, mAP, etc.) are also reduced accordingly.

Figure 5 illustrates the test results. Compared with the base model (Figure 5a), the target detection performance of the lightweight model (Figure 5b) has changed: firstly, the confidence level of some targets fluctuates; secondly, individual targets (especially objects at longer distances) fail to be recognized. Although these changes have limited impact on the simulation experiments, how to reduce the magnitude of the performance degradation is still an issue that needs to be further explored.



Original model New model

Figure 5. Test results of delay optimization experiments

In terms of computing speed, as shown in Table 1, the inference time of the lightweight model increased from 15.1 milliseconds to 15.6 milliseconds. Although the absolute difference is small, it is significant in the strict real-time requirements of unmanned delivery services:

1. Frame rate threshold impact: 15.1 ms corresponds to approximately 66.2 fps, while 15.6 ms corresponds to 64.1 fps. If the target system requires real-time performance of ≥ 60 fps, 15.6 ms still meets the threshold, while the improvement of 15.1 ms, though not crossing the threshold, can reduce latency fluctuations on resource-constrained devices.

2. Cost-effectiveness optimization: Speed improvement is accompanied by a decrease in precision (as shown in Table 1), indicating that the current lightweight strategy has limitations in terms of performance-speed trade-offs. Future research should focus on dynamic structural optimization and feature enhancement modules to minimize precision degradation while maintaining inference speed, such as through distance-adaptive branch adjustment or attention mechanisms to enhance distant feature extraction capabilities.

Table 1. Comparison of computational speed in the first experiment

Model version	Object detail	Inference time	Non-Maximum Suppression (NMS) time
Oder model	2 buckets, 1 chair	15.6 ms	5.0 ms
New model	2 buckets, 1 chair	15.1 ms	5.1 ms

3.3 *Optimization performance optimization*

In the previous experiments, although the computational speed was improved, however, the performance of the model showed a more significant fluctuation, and minimizing the magnitude of this fluctuation is the issue that will be explored in this subsection.

3.3.1 *Optimization solutions*

The problem revealed in the previous experiments was that the model was unable to properly recognize certain objects in the image. The crux of this result lies not only in the insufficient processing power of the model for the image itself,

which makes it impossible to notice other parts of the image that need to be paid attention to during the computation; it also lies in the fact that the feature extraction process does not elevate the status of those important channels, which leads to the underutilization of those channels that have a greater impact on the result, so that the extraction of the features is insufficient, and as a result, the recognition of the corresponding objects is not achieved. The result is that the recognition of the corresponding object is not realized. Due to the complexity and integration of the causes of the problem, it was decided to embed the hybrid attention mechanism in the improved YOLO-MO model to achieve further optimization of the model performance.

Among the studies on hybrid attention mechanisms, CBAM [23] is one of the more classical ones, and the main use of this module is to enhance the performance of convolutional neural networks. The original intention of its proposal is to try to deal with the limitations of traditional convolutional neural networks for different scales, shapes, and other information. CBAM mainly consists of two parts, namely, the channel attention module (C-channel) and the spatial attention module (S-channel), and the enhancement of the feature representations is realized by reasonably embedding the two modules into the CNN model, as Figure 6 shown. By dividing the modules into two structures, there are naturally two big steps in the computation process.

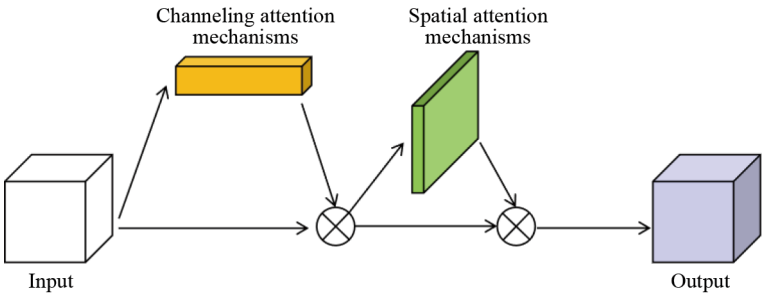


Figure 6. Test results of delay optimization experiments

The first step of the input image will go through the channel attention module, for each image, the first step is to perform global maximum pooling and global average pooling operations for each channel, calculate the maximum feature value and average feature value of each channel, and finally get two vectors; then the module inputs the two features into a shared fully-connected layer, which is mainly used to learn the attention weights of each channel, thus allowing the module to confirm which channel is more important, and also intersect the maximum feature vector and the average feature vector to get the final attention weight vector; in order to ensure that the weights are between 0 and 1, an activation function is needed to generate new weight vectors; use the new vectors to multiply them with each channel of the feature map to get the attention-weighted channel feature maps; and finally, the unimportant channels are suppressed, and the important channels are emphasized. Then it will enter the spatial attention module, in which the process is similar to the channel attention module, but with different dimensions.

When the input data passes through the above two modules, then the resultant features of the two modules are multiplied element by element to get the final attention feature. This enhanced attention feature will be used as an input to the subsequent network structure, which can be done to retain the key information while suppressing the unimportant information and noise to achieve higher model performance.

3.3.2 *Optimization experiments*

Based on the previous section, this optimization focuses on the optimization of the performance of the YOLO-MO model, with the goal of allowing the model to recognize objects that would otherwise be unrecognizable, while improving the confidence level of the individual objects to some extent. The ideal result is a decrease in speed but not to the original YOLO model, and an improvement in performance over YOLO-MO.

In the previous experiments, Mobile One was mainly added to the backbone part of the network, so the introduction of CBAM was mainly focused on the head part of the model, and in the subsequent adjustments, it was also introduced in the backbone as much as possible. Specific experimental results will be given in the next section, the precision of the model is not up to the level of the original model, but compared with the YOLO-MO model, it achieves better results, and the computational speed of the new model is shown in Table 2.

Table 2. Comparison of computational speed in the second experiment

Model version	Object detail	Inference time	NMS time
Oder model	2 buckets, 1 chair	15.6 ms	5.0 ms
New model	2 buckets, 1 chair	15.4 ms	5.1 ms

It can be seen that although the new model (hereafter referred to as YOLO-MOCA) is slower than the YOLO-MO model in terms of computational speed, it is still slightly faster than the YOLO model. Figure 7 shows the results of the new model in the test, compared with YOLO-MO, YOLO-MOCA has indeed achieved the optimization of the effect, the part that can be recognized in the original model but YOLO-MO fails to recognize, can be successfully recognized in the new model, and the confidence level of individual objects of the new model has been improved to a certain extent.

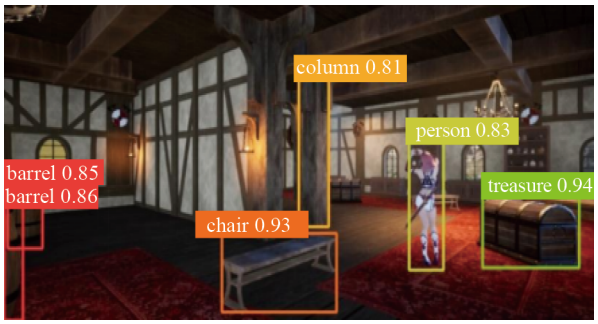


Figure 7. Test results of delay optimization experiments

3.4 *Analysis of experimental results*

In this experiment, the game scene dataset is divided according to the ratio of 4 : 1, in which the training set is 1,600 images and the test set is 400 images. In this experiment, an improved version of YOLOv5 was used. The detailed configuration of this model is shown in Table 3.

To uantitatively analyze the performance of the model and compare the detection effects of multiple scenarios in the experiments, precision, recall, mAP0.5, *obj_loss*, *box_loss*, *cls_loss* and other metrics are used as references to evaluate the model. In order to make the results clearer, a comparative experiment is used to compare with the original YOLOv5 model as well as YOLOv3 and YOLOv4 respectively, and in the data analysis will focus on analyzing the data of the v5 model as well as the improved model.

Figure 8 shows the precision change curves of each model after 100 epochs of iteration in the homemade dataset. The green curve is for the basic YOLOv5 model, which shows an up and down fluctuation around 0.88 after 20 epochs, and the maximum value reaches 0.92 after stable fluctuation; the blue line is for the first step of the improved model YOLO-MO, which has a more obvious decline compared with the past, and still has a big fluctuation around 40 epochs, and its maximum value is about 0.88 after relative stability; the red curve is for the final YOLO-MOCA model. The blue line is the curve of YOLO-MO, which has a more obvious decline compared with the past, and still has a big fluctuation after 40 epochs, and its maximum value is about 0.88 after relative stabilization; the red curve is the final YOLO-MOCA

model, which has a more stable overall trend, and its maximum value is about 0.90 at the end, which has a slight increase compared with YOLO-MO.

Table 3. Improving YOLOv5 model parameter configuration

Parameter category	Parameter name	Setpoint	NPC scene optimization instructions
Input settings	Image size	640 × 640	Balancing detection precision and real-time performance
	Multi-scale zoom range	± 50%	Zoom range adapts to NPC fixed camera angle (avoids distortion of distant targets)
Training strategy	Batch Size	8	Computer memory limitations
	Optimizer	AdamW	Add gradient clipping (max_grad_norm = 1.0) to prevent gradient explosion caused by violent shaking in dynamic environments
	Initial learning rate	1×10^{-2}	Grid search validation (range 1×10^{-3} to 1×10^{-2})
	Weight Decay	5×10^{-4}	Reducing overfitting in lightweight models
	Epochs	100	Increase epochs to address complex obstacle diversity
Data augmentation	Mosaic enhancement	0.8	Reduce the probability of distortion caused by close-range obstacle splicing
	Motion blur enhancement	0.2	Image blurring when simulating NPC movement
	Random rotation angle	± 15°	Restricting the rotation range maintains the physical plausibility of ground obstacles
Model architecture	Detector resolution	[80, 40, 20]	Enhanced small object detection
	NMS threshold	0.45	Stricter filtering reduces false positive rates for dense obstacles
	Detection confidence threshold	0.4	Balancing false negatives and false positives
Hardware configuration	GPU model	NVIDIA RTX 3060	–
	CUDA version	11.3	Compatible with PyTorch 1.10

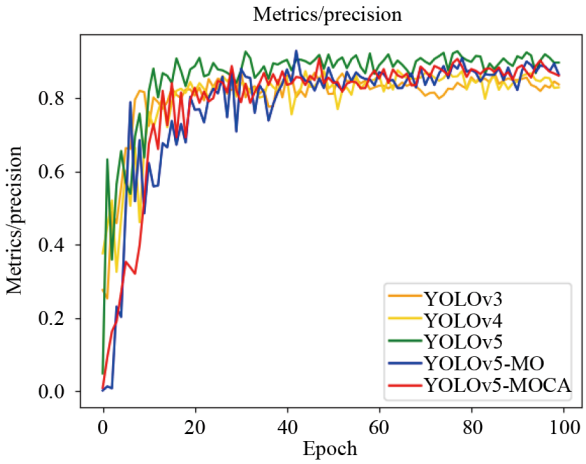


Figure 8. Precision change curve

Figure 9 shows the recall change curves of each model after iterating 100 epochs in the homemade dataset. The green curve is for the basic YOLOv5 model, which overall fluctuates up and down around around about 0.86, with a maximum value of about 0.88, just like the precision curve; the blue curve is for the YOLO-MO model, which can be seen in the figure that there are large fluctuations for the first 40 epochs, after which it gradually tends to a relatively stable state. The blue curve is the YOLO-MO model, from the graph, we can see that the first 40 epochs have large fluctuations, and after that, it tends to be relatively stable, with a maximum value of about 0.87; the red curve is the last YOLO-MOCA

model, the overall trend is more obvious compared with other models, and the overall trend is more obvious than the other models, and the model is gradually stabilized after 60 epochs, and ultimately exceeds the original YOLOv5 model, with a recall rate of about 0.9, and the integrated precision curve shows that this phenomenon is caused by the introduction of the attention mechanism. The introduction of the attention mechanism leads to this phenomenon, i.e., the recall rate increases while the precision rate decreases.

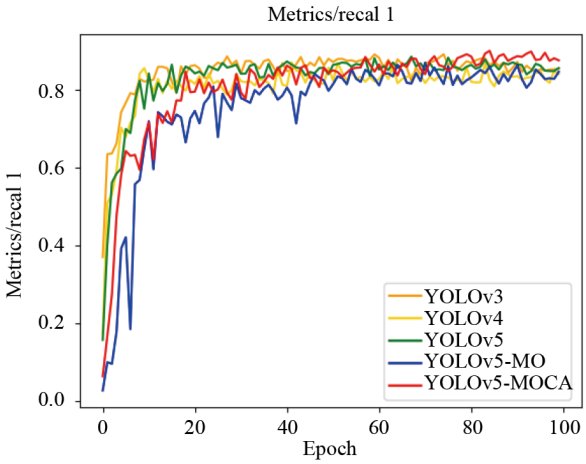


Figure 9. Recall change curve

Figure 10 shows the mAP 0.5 change curves of each model after iterating 100 epochs in the homemade dataset, the green curve is the initial YOLOv5 model, and from the figure, we can see that its convergence speed is a little bit faster than in the past, and the final maximum value reaches about 0.93; the blue curve is from the YOLO-MO model, and we can see that it fluctuates a lot in the first 40 iterations of the training, and the blue curve is from the YOLO-MO model, and we can see that in the first 40 iterations of training, its fluctuation is larger and its mAP is lower, and after that, it is at a higher level, and the maximum value is about 0.91; finally, the red curve is the mAP change curve of YOLO-MOCA, which is more stable than YOLO-MO, mainly because the fluctuation becomes smaller and converges earlier, and the final maximum value is on the same level as that of YOLO-MO, and it reaches about 0.91.

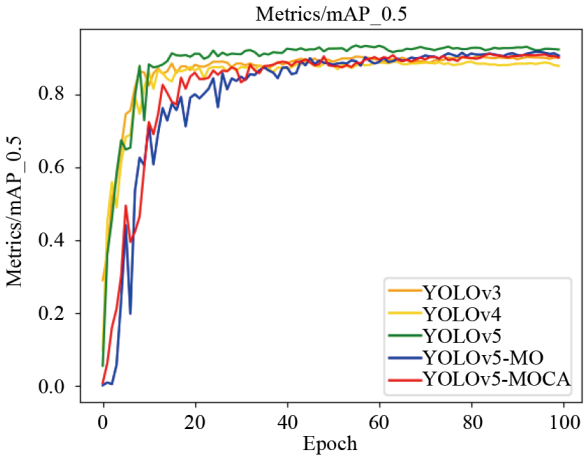


Figure 10. mAP_0.5 change curve

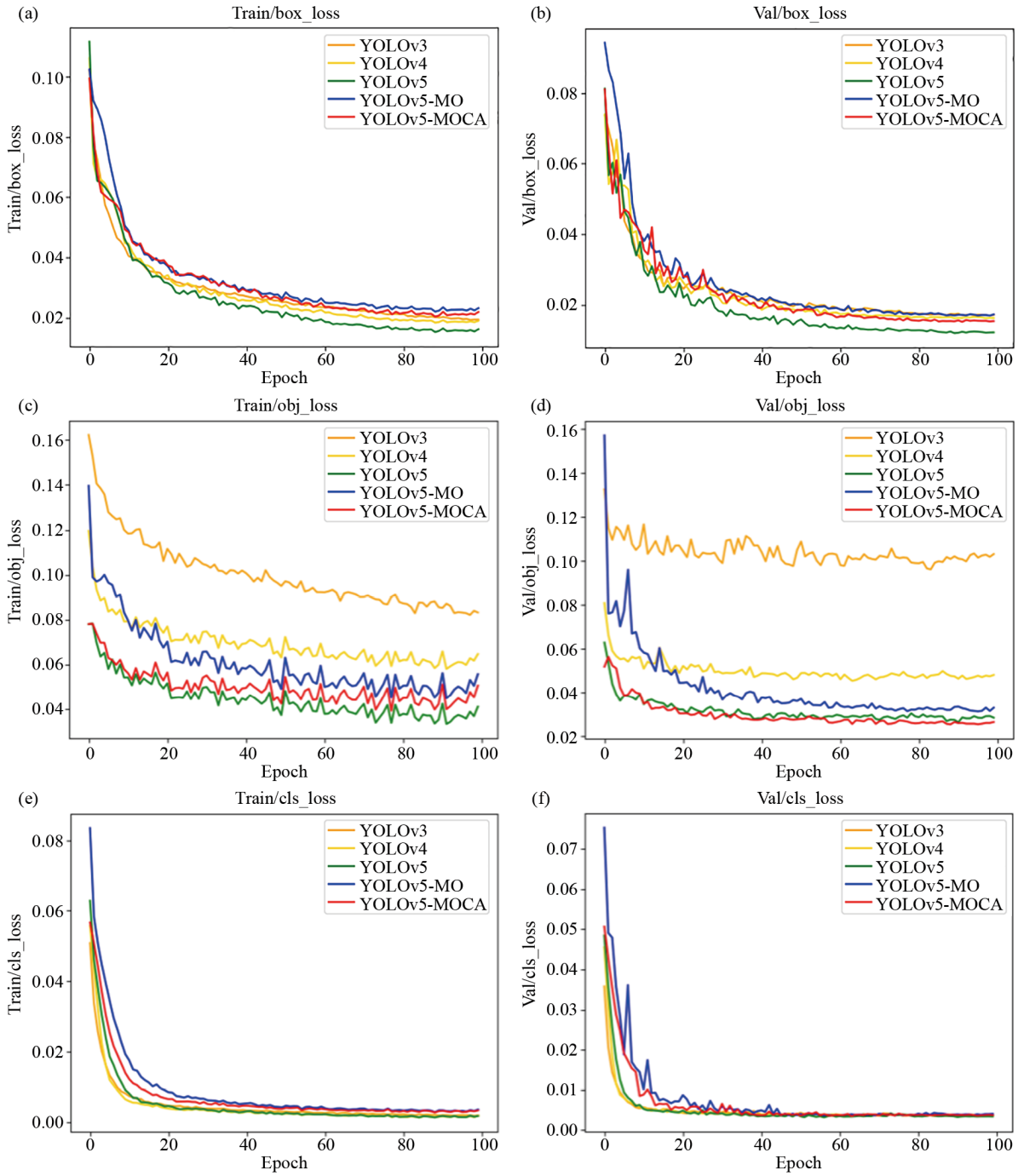


Figure 11. Loss function variation curve

Figure 11 shows the loss function plots after training 100 epochs in the homemade dataset, where the left column (a), (c), (e) plots are the loss function plots during the training process, and the right column (b), (d), and (f) plots the change of the loss function during the testing process.

The two plots in Figure 11a and Figure 11b are the loss function plots of the bounding box (box), the green curve YOLOv5 model box_loss has the best performance, the starting values of the training and testing are about 0.112 and 0.081 respectively, and eventually decreases to the neighborhood of 0.015 and 0.012, respectively; the blue color is the

obj_loss curve of the YOLO-MO model, with a combined performance is poor, with the loss decreasing from 0.102 to near 0.022 during training, and from 0.094 to near 0.017 during testing; the red curve is the final improved model, with the loss decreasing from 0.102 to near 0.024 during training, and from 0.084 to near 0.018 during testing, and it can be seen that under the obj_loss metrics, the new model is not as good as the original YOLOv5 model, but performs better than YOLO-MO.

The two figures in Figure 11c and Figure 11d show the curves of the object (obj) loss function change, the green curve is the YOLOv5 model, whose obj_loss decreases from 0.078 to near 0.036 during training and from 0.062 to near 0.028 during testing; the blue curve is the YOLO-MO curve, which decreases from 0.139 and 0.157 to near 0.028 during training and testing, respectively. 0.157 to near 0.053 and 0.032 in training and testing respectively, which can be seen that the preformance is not as good as that of the initial model; the red curve is for the YOLO-MOCA model, which decreases from 0.078 to near 0.047 in the training set and from 0.051 to near 0.025 in testing, which occurs probably due to the introduction of the attentional mechanism that makes the model recognize the objects more easily.

Figure 11e and Figure 11f show the classification (cls) loss function curves of the model, and the individual curves have similar trends in the graphs, the green curve is for the YOLOv5 model, whose cls_loss decreases from 0.063 and 0.048 to the neighborhoods of 0.002 and 0.003 in the training and testing, respectively; the blue curve is for the YOLO-MO model's cls_loss curve, where the loss decreases from 0.083 to near 0.003 at the very beginning in the training set, and from 0.075 to near 0.004 in the test; while the red curve YOLO-MOCA is similar to YOLO-MO, where the loss decreases from 0.057 to near 0.003 in the training set, and from 0.051 to near 0.004 in the test, which is a slightly worse performance compared to the initial model.

In terms of the performance of speed optimization, the results of a single experiment are given in the previous experiments, and Table 4 shown next records the output times of the three YOLOv5 models for 10 consecutive detections of the same image. In the table delay optimization refers to the YOLO-MO model and performance optimization refers to the YOLO-MOCA model. There are three values in the output of each model, pre-process, interference, and NMS, all of which are speed-related metrics, all of which have an impact on the total time and the values fluctuate, so the values in the table are the results of adding up the three. The average value of the original model comes to 23.6 ms, while the average value of the YOLO-MO model is 21.2 ms, which is better than the original model overall. The CBAM model is slower than the YOLO-MO model, with an average of 21.7 ms, but it is mostly better than the initial model, although occasionally it takes longer than the original model, mainly because the addition of modules to the model will naturally increase the time, and the original reduction of the time will not be too much, so it is easy to fluctuate and occasionally longer than the original model.

Table 4. Comparison of model speeds (ms)

Model	1	2	3	4	5	6	7	8	9	10	Mean	Std
YOLOv5	23	21.4	22.5	21	21	22	22	21	23.1	23.6	22.1	1.0
Latency optimization	22.5	21	21.2	20.8	20.6	22.3	20.8	20.8	21.1	21.2	21.2	0.6
Performance optimization	23.1	21.2	20.9	21.5	20.3	22.7	23.3	20	21.4	21.7	21.6	1.1

To comprehensively evaluate the effectiveness of YOLO-MOCA, this study selected six cutting-edge models optimized for robot tasks as benchmarks:

DenseNet-121 [38]: This model uses a dense connection structure to enhance the transmission of multi-scale obstacle features, making it suitable for real-time obstacle avoidance decisions in complex environments.

EfficientNetB0 [39]: This model optimizes computational efficiency through composite scaling, providing balanced perception capabilities for resource-constrained mobile robots.

GoogLeNet [40]: This model uses a multi-parallel convolution structure with Inception modules to simultaneously detect obstacles of different sizes, improving dynamic obstacle avoidance adaptability.

MNasNet-05 [41]: This model achieves a lightweight design through search optimization using a neural network architecture to meet the low-latency obstacle avoidance requirements of robots.

MobileNetV2 [42]: This model uses an inverted residual block structure, which significantly reduces computational energy consumption while maintaining precision.

Extended MobileNetV2 [43]: This model enhances the identification of key obstacles through an improved attention mechanism, enabling efficient path planning.

Table 5. Comparison of model performance

Model	Precision	Recall	mAP@0.5	Time/ms
DenseNet-121	0.88	0.88	0.85	48.2
EfficientNetB0	0.86	0.86	0.83	32.5
GoogLeNet	0.86	0.87	0.84	36.8
MNasNet-05	0.61	0.56	0.58	18.6
MobileNetV2	0.89	0.88	0.86	22.4
Extended MobileNetV2	0.92	0.89	0.89	24.1
YOLOv5	0.92	0.88	0.93	23.6
Latency optimization	0.88	0.87	0.91	21.2
Performance optimization	0.90	0.90	0.91	21.7

The experimental results demonstrate that the proposed YOLO-MOCA model achieves a significant breakthrough in the trade-off between precision and speed. As shown in Table 5, in terms of precision, YOLO-MOCA attains an mAP@0.5 of 0.91, surpassing most baseline models and trailing only the original YOLOv5 (0.93). Compared to MobileNetV2 (0.86) and Extended MobileNetV2 (0.89), it exhibits precision improvements of 5.8% and 2.2%, respectively. Regarding real-time performance, its inference speed reaches 21.7 ms, marking an 8.1% enhancement over the original YOLOv5 (23.6 ms), while significantly outperforming models such as DenseNet-121 (48.2 ms) and EfficientNetB0 (32.5 ms). Notably, while maintaining high precision, YOLO-MOCA's speed approaches that of the lightweight-designed MNasNet (18.6 ms), yet delivers a 56.9% improvement in mAP@0.5. These findings validate the effectiveness of synergistic optimization between the Mobile One and CBAM modules, providing a highly suitable solution for application scenarios with stringent real-time requirements.

4. Obstacle avoidance decision-making program

4.1 *Distance measurement algorithm design*

After the NPC detects an obstacle, it also needs to know the distance between itself and the obstacle in order to make a decision, this subsection mainly discusses the idea and implementation of the distance measurement algorithm, while the idea can refer to the related algorithms of automatic driving [24].

4.1.1 *Basic idea*

The core idea of this algorithm is based on the perspective projection geometry principle in computer vision, which is essentially an adaptive improvement of the similar triangle theory in specific scenarios. Specifically, in the traditional similar triangle model shown in formula (1), the actual height x of the object satisfies the proportional relationship with the image height b, focal length c, and object distance a:

$$\frac{x}{b} = \frac{a}{c}, \tag{6}$$

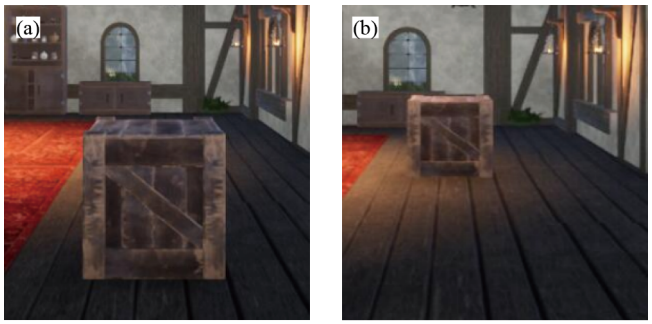
where x is positively correlated with a. Specifically, the algorithm first obtains the pixel coordinates of the obstacle bounding box through object detection, and then calculates its pixel area s. To establish a distance measurement model, we predefine a baseline scenario: when the physical distance between the obstacle and the NPC is a known value l, the corresponding bounding box pixel area is s_1. During real-time operation, the system obtains the pixel area s_2 of the obstacle's bounding box in the current image through object detection. However, in the perspective projection model of computer vision, the actual distance d of an object is inversely proportional to its pixel area s on the imaging plane-the farther the distance, the smaller the pixel area. Therefore, we derive the distance measurement formula by calibrating the baseline (known distance l corresponds to area s_1) and real-time detection (distance x corresponds to area s_2):

$$\frac{s_1}{s_2} = \left(\frac{x}{l}\right)^2. \tag{7}$$

This formula establishes a quantitative relationship between pixel area change and actual distance, satisfying the universal applicability of perspective projection geometry: when the distance of an object from the lens changes from l to x, its image area changes in inverse proportion to the square of the distance.

4.1.2 *Agorithm validation*

Next to verify the feasibility of this method, a wooden box is placed in the scene in Figure 12, the distance between the wooden box and the NPC is unknown in Figure 12b, and the distance in Figure 12a is known to be 500 according to the default units of the Unreal Engine, respectively, after the target detection, obtain the coordinates of the points of the two wooden box frames and calculate the area of the two frames according to the coordinates, and get the area of the box in close proximity to be 23,855 and the area of the box of distant objects be 5,744, according to the above proportionality relationship, the estimated distance is 1,018.58, while the actual dWistance in the game is 1,000. 23,855, the area size of the distant object box is 5,744, according to the above proportionality, the estimated distance is 1,018.58, and the actual distance in the game is 1,000, it can be seen that although the results of the calculations and the actual distance is different, but due to the experimental environment is not too strict, and after a number of times of the above calculations, the average error is at about 2%, so we believe that this proportionality can be used to estimate the distance.



Nearby Far away

Figure 12. Formula validation

4.1.3 *Special case solutions*

In the scene, there are some objects whose angles in the image do not have much effect on the area they occupy in the image, such as columns, barrels and characters, and a screenshot of the characters from multiple angles can show that the difference in their area is not large. While there are some objects such as the table, from different angles to see the past its frame area gap is large, such as the example in Figure 13, the distance between the object and the NPC in the two images

is the same, but the angle of placement is different, after the calculation to get the Figure 13a area of 41,866, the Figure 13b area of 60,242, it can be found that the area gap is very obvious, so that if it does not carry out a special treatment will affect the final The difference in area is very obvious, which will affect the final result of distance estimation if it is not treated specially.

Original angle New angle Verification

Figure 13. Special case

In order to solve this problem, we can consider setting multiple benchmark images for this kind of object under the circumstance of allowing errors, take Figure 13 as an example, we can regard Figure 13a as the original benchmark image, and Figure 13b is the benchmark image after the object is rotated by 45 degrees, and then according to the length-to-width ratio of the frames in the benchmark image, we can divide the ratio into an approximate range, and use different benchmark images for the distance estimation within the different ranges. distance estimation using different datum images in different ranges can solve this problem to some extent. For example, in the case of Figure 13c, the object is presented in the image at an angle that is not its front face, but a certain angle on the side. The distance of the object in the new base image is known, and the distance of the object in the left image is 675.8, while the actual distance is 700, with an error of 3.46%, which is larger than the usual error, but since the requirements of the experiment are not too strict, the error is still in an acceptable range.

4.2 *Distance measurement algorithm design*

After the NPC detects the obstacle target through the vision model and calculates the distance through the ranging algorithm, the next step is to avoid the obstacle. This section describes the design of the avoidance decision mechanism, describes its process, and examines the results.

4.2.1 *Decision-making process*

Two special cases that were not discussed in the above discussion are the partial objects that appear at the left and right edges of the image, as shown by the objects labeled yellow in Figure 14a. This case does not actually have much effect on the movement of the NPC. Because the size of the NPC's model also needs to be taken into account when the NPC moves forward, and in this experiment the NPC's model is small, and if it continues to move forward, the edge objects actually have no effect on the NPC's advancement, so a range can be set in the decision-making, roughly as shown in Figure 14b, and the objects within the range of the two blue lines will be prioritized. Thus, objects on the left and right sides can be automatically ignored at first in the decision-making process, and only objects directly in front of the NPC are prioritized, and then whether to include them in the calculation range can be considered according to various situations. And according to the estimation, in order to take care of different resolutions, a range of 7/24 can be taken in the left and right of the screen for exclusion.

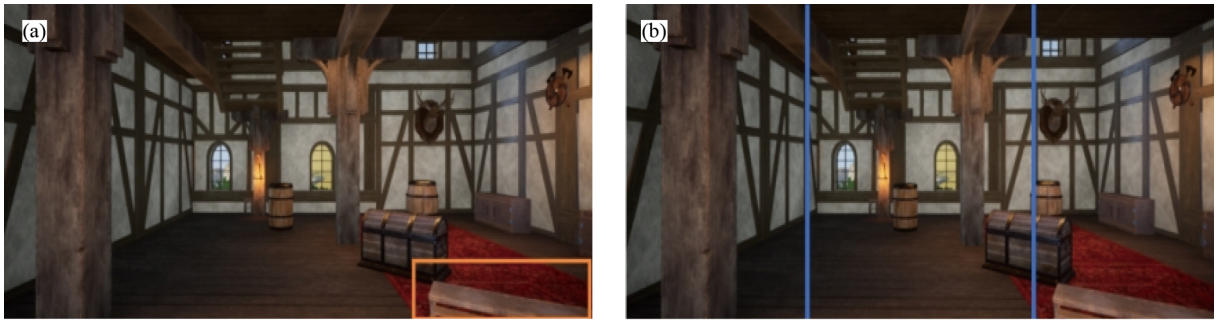


Figure 14. Edge object handling

Another situation is the part of the object appearing below the image directly in front of the NPC, for example, such as the one shown in Figure 15, in this case, it can generally be directly determined that this object is the closest to the NPC, and the decision can be made directly disregarding the distance information, and the judgment is made by relying on the corners of the box and the center of the box directly below it, with the points on the lower side of the longitudinal coordinates approaching 0, and with a gap in the aspect ratio of the box and all the baseline images. In the calculation, the model will estimate the distance of each object according to the recognition box while recognizing the object, and screen out the objects on both sides according to the coordinate information to ensure the completeness of the image information of the nearest object. The first step is to determine whether there is an object in the middle, without which you can go straight ahead. Then see if there is an object directly below the image, if there is, then you can directly let the NPC perform a random left-right rotation of 90°. If there is no object, then we compare the distance of each object, and in our experiments we found that in general the two closest objects have the greatest influence on the results.

Figure 15. Object directly underneath

In the next calculation, objects on both sides of the image that were screened out at the beginning may be used. The distance of the nearest object in front is calculated, if the distance is greater than the threshold, the object will not affect the NPC for the time being and maintain the straight ahead movement. Here the threshold is also the distance, after comprehensive consideration, the threshold is tentatively set to 600, on the contrary, the object may hinder the NPC to move forward, in accordance with the orientation of the object to initially confirm the direction of movement, in this case the object is on the right side of the NPC as an example, at this time initially determined that the NPC needs to be deflected to the left. Among all the objects located to the left of the nearest object, identify the second closest object to the NPC, if the distance of this object is greater than the threshold, it is determined that it will not affect the movement for the time being, and then the NPC will turn left by 45°; on the contrary, if the second closest object is also within the threshold, the distance between the two objects will be calculated.

$$x = \arctan\left(\frac{CE}{DE}\right). \tag{8}$$

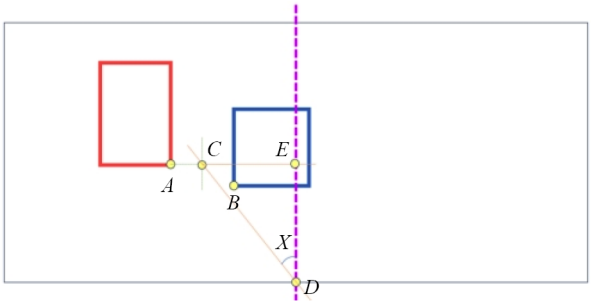


Figure 16. Object directly underneath

As shown in Figure 16, the coordinates of the four corners of the box can be obtained directly in YOLO, so it is only necessary to calculate the difference between the horizontal coordinates of the lower-right corner of the left object and the lower-left corner of the right object to obtain the spacing of the two objects, if the spacing is greater than the width of the NPC, then calculate the angle angle between the midpoint of the two to the NPC's line and the center line of the picture, so that it is used as a shift angle to help the NPC for offsetting. There are two objects in the image were marked by the red and blue boxes, can be calculated with the help of A, B two-point coordinates of the interval between the two boxes of the midpoint C, point C and the image of the center line constitutes a triangle CDE, through the D, C two points can be obtained from the length of CE and DE, through the inverse trigonometric function can be found out the angle of X, Equation (8) for this process of calculation of the formula; Conversely, if the distance is not enough, then the choice of letting the NPC randomly to the left and right shift 90°.

4.2.2 *Decision-making mechanism validation experiment*

After the decision-making mechanism has been determined, it needs to be validated to ensure that the present mechanism can be computed to obtain the correct result, i.e., it allows the NPC to achieve obstacle avoidance. In the formal experiments, during the NPC's travel, the eye camera will periodically capture images, which will then be sent to the model for recognition, after which distance measurement as well as decision making will be performed.

In this validation experiment, the above process will be simulated, first of all, the NPC will be placed in the scene, the camera captures the image and its results after target detection as shown in Figure 17a, referring to the decision-making process described above, because the wooden box is underneath, so it should be given the instruction to randomly rotate to the left or right by 90°, and after the algorithm calculates that its final output is -90, i.e., 90° to the left; Control the NPC to turn and move forward a little, and again intercept the image, the image at this time for Figure 17b, according to the logic algorithm should also be given to randomly rotate to the left and right 90° instructions, the result of the output of the same -90; Figure 17c is in accordance with the same process of the intercepted image, according to the logic output should be -10 or so, from the treasure chests and crates between the walk through the output of the results of the -9.7, a little deviation, but in the simulation, the final output is -90, i.e., turn left 90°. Some deviation, but the simulation can indeed be successfully passed; finally came to the scene in Figure 17d, according to the logic of the obvious should be passed from the columns and chairs, the algorithm output is -22.3, can be passed normally. To summarize, the output can ensure that the NPC can take the correct moving route most of the time.

In addition to this, it is necessary to verify that normal decision making is still possible with the interference of the player character. Move the player to a situation similar to Figure 18, where the player is located in front of the NPC. According to the decision-making process, since the player's position is to the right, it will be involved in the calculation together with the objects on the left, and the distance between the player and the crate on the left is not enough, so it should be randomly rotated to the left and right by 90°, and the output result will be + 90, that is, rotated to the right by 90°. To summarize, the output can ensure that the NPC can take the correct moving route most of the time.

Figure 17. Status of target detection

Figure 18. Object directly underneath

4.3 *Interaction mechanism design*

In the previous section, the target detection and decision-making scheme have been determined, and the next step is to determine how to let NPCs get the results of decision-making. According to the analysis, it is only necessary to build a bridge between Python and Unreal Engine, and the characteristics of this bridge is that both language environments can be easily accessed and called, based on this feature can be initially determined to let the local file system become the bridge.

We just need to let the output of the model be saved in a local text file, so that the Unreal Engine side can read the result and make corresponding actions. The specific process is to calculate the results on one side of the model, that is, the information of the offset angle and direction, record it in the local text file, and then the Unreal Engine reads the file information according to the cycle and reacts according to the information, which is shown in the operation logic schematic in Figure 19.

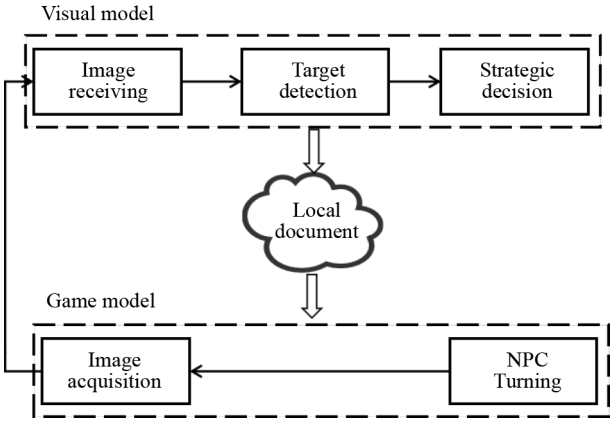


Figure 19. Interaction diagram

4.4 *NPC obstacle avoidance experiment*

After designing the interaction mechanism, all the basic components of the whole demonstration system game objects, characters, target detection models, decision mechanisms, etc. are in place, and in this section the system experiments will be formally carried out after making preparations and the results of the experiments will be elaborated.

4.4.1 *Experimental process observation*

This subsection will observe and record the movement of NPCs in the scene. Firstly, after running the game the NPCs will move in the scene, while at the same time the loop calculation is turned on at the visual model. The observer can then manipulate the player character to impede the NPC's progress.

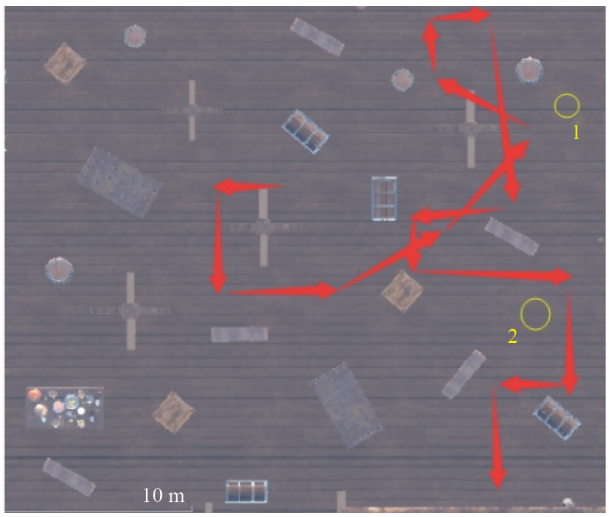


Figure 20. Interaction diagram

The results were recorded in the form of observing the movement route while drawing the NPC's advancement route in the top view of the scene. The line formed by the red arrows in Figure 20 is the NPC's action route in one of the experiments, in which the circled portion is the player's position, and the sequence of the player's movement between different positions is pointed out by the red arrows. After a few more experiments, the NPCs were able to move smoothly

in most cases. However, in some cases, either due to the construction problem of the scene or the matching problem between the calculation cycle and the NPC's moving speed, the NPC could not move forward normally due to the wrong route. Therefore, the decision-making algorithm was added to deal with the relevant situations, and after the NPC was stuck for a certain period of time, it turned around, and finally made the NPC keep moving in the scene.

4.4.2 *Conclusion of the experiment*

When the experiment starts, the NPC moves automatically in the scene, and the camera located at the eye of the NPC periodically collects images and delivers them to the computer vision model for target recognition. In this experiment, the computer vision model can recognize all the objects in the image that have a large impact on the NPC's advancement normally, and can make decisions based on the recognition results, thus giving the direction that the NPC will go to next. The results of the model processing can be normally received by the game, and the NPC in the game can turn according to the received information, and can move more smoothly in the scene, while avoiding the obstacles in front of them, the initial concept is basically completed.

5. Conclusion

In this study, we conducted a series of design and implementation work around the use of a game NPC obstacle avoidance simulation technique to simulate the obstacle avoidance behavior of a general-purpose unmanned delivery service robot. First, we optimized the adopted computer vision model in terms of both latency and performance to ensure that it better adapts to the experimental requirements. Then, we designed the NPC's decision logic in detail, including obtaining the decision basis through distance measurement, utilizing the decision algorithm to derive the action result, and realizing the result transmission through the interaction mechanism, which finally enables the NPC to move and avoid obstacles smoothly in the game scene. Looking ahead, we will further focus on the reliability and ubiquity of the solution. Specifically, we will endeavor to improve the stability and applicability of the scheme in practical applications. At the same time, we will also carry out in-depth optimization of the fineness of the decision-making results to improve the NPC's responsiveness and intelligence level in complex environments, so as to provide more mature technical support for the actual deployment of unmanned delivery service robots.

Acknowledgement

This study was supported by the National Natural Science Foundation of China (U21A20474) and the General Program of Guangxi Natural Science Foundation (2024JJA170142).

Conflict of interest

The authors declare no competing financial interest.

References

[1] Ribeiro G, Monge J, Postolache O, Pereira JMD. A novel AI approach for assessing stress levels in patients with type 2 diabetes mellitus based on the acquisition of physiological parameters acquired during daily life. *Sensors*. 2024; 24(13): 4175. Available from: https://doi.org/10.3390/s24134175.

[2] Przegalinska A, Triantoro T, Kovbasiuk A, Ciechanowski L, Freeman RB, Sowa K. Collaborative AI in the workplace: Enhancing organizational performance through resource-based and task-technology fit perspectives. *International Journal of Information Management*. 2025; 81: 102853. Available from: https://doi.org/10.1016/j.ijinfomgt.2024.102853.

[3] Chang CY, Santra AS, Chang IH, Wu SJ, Roy DS, Zhang Q. Design and implementation of a real-time face recognition system based on artificial intelligence techniques. *Multimedia Systems*. 2024; 30(2): 114. Available from: https://doi.org/10.1007/s00530-024-01306-y.

[4] Wang B, Gan L. Exploring factors influencing the acceptance of digitalized smart campus cafeteria models: A utaut model-based analysis. In: *2023 International Symposium on Computational Intelligence and Design (ISCID)*. Hangzhou, China: IEEE; 2023. p.38-42.

[5] Liu S, Tang Y, Tian Y, Su H. Visual driving assistance system based on few-shot learning. *Multimedia Systems*. 2023; 29(5): 2853-2863. Available from: https://doi.org/10.1007/s00530-021-00830-5.

[6] Leong PY, Ahmad NS. LiDAR-based obstacle avoidance with autonomous vehicles: A comprehensive review. *IEEE Access*. 2024; 12: 164248-164261. Available from: https://doi.org/10.1109/ACCESS.2024.3493238.

[7] Armanto H, Rosyid HA, Muladi M, Gunawan. Improved non-player character (NPC) behavior using evolutionary algorithm-A systematic review. *Entertainment Computing*. 2025; 52: 100875. Available from: https://doi.org/10.1016/j.entcom.2024.100875.

[8] Banerjee C, Nguyen K, Fookes C, Karniadakis GE. Physics-informed computer vision: A review and perspectives. *ACM Computing Surveys*. 2025; 57(1): 1-38. Available from: https://doi.org/10.1145/3689037.

[9] Jiang H, Lu Y, Zhang D, Shi Y, Wang J. Deep learning-based fusion networks with high-order attention mechanism for 3D object detection in autonomous driving scenarios. *Applied Soft Computing*. 2024; 152: 111253. Available from: https://doi.org/10.1016/j.asoc.2024.111253.

[10] Day S, Smallwood WK, Kuhn J. Simulating industrial control systems using Node-RED and Unreal Engine 4. In: *National Cyber Summit (NCS) Research Track 2021*. Cham: Springer International Publishing; 2022. p.13-21.

[11] Zhu X, Lyu S, Wang X, Zhao Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. IEEE: Canada; 2021. p.2778-2788.

[12] Kopel M, Hajas T. Implementing AI for non-player characters in 3D video games. In: *Intelligent Information and Database Systems*. Cham: Springer; 2018. p.610-619.

[13] Bombara G, Vasile CI, Penedo F, Yasuoka H, Belta C. A decision tree approach to data classification using signal temporal logic. In: *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. New York: Association for Computing Machinery; 2016. p.1-10.

[14] Leike J, Krueger D, Everitt T, Martic M, Maini V, Legg S. Scalable agent alignment via reward modeling: A research direction. *arXiv:181107871*. 2018. Available from: https://doi.org/10.48550/arXiv.1811.07871.

[15] Gouda KC, Thakur R. Energy-efficient clustering and path planning for UAV-assisted D2D cellular networks. *Ad Hoc Networks*. 2025; 170: 103757. Available from: https://doi.org/10.1016/j.adhoc.2025.103757.

[16] Dobrev S, Narayanan L, Opatrny J, Pankratov D. Exploration of High-dimensional grids by finite state machines. *Algorithmica*. 2024; 86(5): 1700-1729. Available from: https://doi.org/10.1007/s00453-024-01207-6.

[17] Edwards G, Subianto N, Englund D, Goh JW, Coughran N, Milton Z, et al. The role of machine learning in game development domain-A review of current trends and future directions. In: *2021 Digital Image Computing: Techniques and Applications (DICTA)*. Gold Coast, Australia: IEEE; 2021. p.1-7.

[18] Shao K, Tang Z, Zhu Y, Li N, Zhao D. A survey of deep reinforcement learning in video games. *arXiv:191210944*. 2019. Available from: https://doi.org/10.48550/arXiv.1912.10944.

[19] Olawade DB, Omeni D, Gore MN, Hadi M. Enhancing qualitative research through virtual focus groups and artificial intelligence: A review. *International Journal of Medical Informatics*. 2025; 203: 106004. Available from: https://doi.org/10.1016/j.ijmedinf.2025.106004.

[20] He L, Wang B, Peng Y, Zhang X. An unmanned sweeper path planning algorithm for structured roads. *IEEE Access*. 2024; 12: 26242-26250. Available from: https://doi.org/10.1109/ACCESS.2024.3359644.

[21] Mirowski P, Pascanu R, Viola F, Soyer H, Ballard A, Banino A, et al. Learning to navigate in complex environments. In: *International Conference on Learning Representations*. London: OpenReview; 2017. p.1059-1067.

[22] Jiang P, Ergu D, Liu F, Cai Y, Ma B. A review of yolo algorithm developments. *Procedia Computer Science*. 2021; 199: 1066-1073. Available from: https://doi.org/10.1016/j.procs.2022.01.135.

[23] Redmon J, Divvala SK, Girshick RB, Farhadi A. You only look once: Unified, real-time object detection. In: *IEEE Conference on Computer Vision and Pattern Recognition*. USA: IEEE; 2016. p.779-788.

[24] Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. In: *IEEE Conference on Computer Vision and Pattern Recognition*. USA: IEEE; 2017. p.6517-6525.

[25] Redmon J, Farhadi A. YOLOv3: An incremental improvement. *arXiv:180402767*. 2018. Available from: https://doi.org/10.48550/arXiv.1804.02767.

[26] Bochkovskiy A, Wang CY, Liao HYM. YOLOv4: Optimal speed and accuracy of object detection. *arXiv:200410934*. 2020. Available from: https://doi.org/10.48550/arXiv.2004.10934.

[27] Zheng Z, Yu W. RG-YOLO: Multi-scale feature learning for underwater target detection. *Multimedia Systems*. 2025; 31(1): 26. Available from: https://doi.org/10.1007/s00530-024-01617-0.

[28] Wang Y, Yang Z, Liu R, Li D, Lai Y, Ouyang L, et al. Multi-attribute object detection benchmark for smart city. *Multimedia Systems*. 2022; 28(6): 2423-2435. Available from: https://doi.org/10.1007/s00530-022-00971-1.

[29] Li C, Li L, Jiang H, Weng K, Geng Y, Li L, et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv:220902976*. 2022. Available from: https://doi.org/10.48550/arXiv.2209.02976.

[30] Wang CY, Bochkovskiy A, Liao HYM. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Canada: IEEE; 2023. p.7464-7475.

[31] Hidayatullah P, Syakrani N, Sholahuddin MR, Gelar T, Tubagus R. YOLOv8 to YOLO11: A comprehensive architecture in-depth comparative review. *arXiv:250113400*. 2025. Available from: https://doi.org/10.48550/arXiv.2501.13400.

[32] Lin TY, Maire M, Belongie SJ, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: Common objects in context. In: *European Conference on Computer Vision*. Cham: Springer; 2014. p.740-755.

[33] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. *arXiv:170603762*. 2017. Available from: https://doi.org/10.48550/arXiv.1706.03762.

[34] Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. USA: IEEE; 2018. p.7132-7141.

[35] Xing J, Lu J, Zhang K, Chen X. ADT: Person re-identification based on efficient attention mechanism and single-channel dual-channel fusion with transformer features aggregation. *Expert Systems with Applications*. 2025; 261: 125489. Available from: https://doi.org/10.1016/j.eswa.2024.125489.

[36] Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K. Spatial transformer networks. *arXiv:150602025*. 2015. Available from: https://doi.org/10.48550/arXiv.1506.02025.

[37] Wang Y, Pu J, Miao D, Zhang L, Zhang L, Du X. SCGRFuse: An infrared and visible image fusion network based on spatial/channel attention mechanism and gradient aggregation residual dense blocks. *Engineering Applications of Artificial Intelligence*. 2024; 132: 107898. Available from: https://doi.org/10.1016/j.engappai.2024.107898.

[38] Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. USA: IEEE; 2017. p.2261-2269.

[39] Tan M, Le QV. EfficientNet: Rethinking model scaling for convolutional neural networks. In: *International Conference on Machine Learning*. USA: PMLR; 2019. p.6105-6114.

[40] Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, et al. Going deeper with convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. USA: IEEE; 2015. p.1-9.

[41] Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, et al. MnasNet: Platform-aware neural architecture search for mobile. In: *IEEE Conference on Computer Vision and Pattern Recognition*. USA: IEEE; 2019. p.2820-2828.

[42] Sandler M, Howard AG, Zhu M, Zhmoginov A, Chen LC. MobileNetV2: Inverted residuals and linear bottlenecks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. USA: IEEE; 2018. p.4510-4520.

[43] Misir O, Celik M. Visual-based obstacle avoidance method using advanced CNN for mobile robots. *Internet of Things*. 2025; 31: 101538. Available from: https://doi.org/10.1016/j.iot.2025.101538.