Face recognition and real-time tracking system based on convolutional neural network and parallel-cascade PID controller

Teh-Lu Liao\textsuperscript{1}, Hsin-Chieh Chen\textsuperscript{2}, Qing-Huang Song\textsuperscript{1} and Yi-You Hou\textsuperscript{3}\textsuperscript{\Letter}

Abstract
The purpose of this research is to develop a high-efficiency, low-cost, and easy-to-use tracking system for vehicles, and it is expected that the system can be extended to areas such as service robots, autonomous driving, and manufacturing. In this paper, we introduced an object detection algorithm based on convolutional neural networks to realize face recognition, which has better efficiency and robustness than traditional machine learning methods. With the concept of edge computing, we deployed the model on the local embedded system to improve the information transmission and security issues of cloud computing. In order to realize the tracking system, this paper builds a mecanum-wheel vehicle with omnidirectional mobility, and proposes a parallel-cascade PID controller architecture based on the mecanum-wheel vehicle. The fixed distance linear tracking control can be realized through the dual-loop feedback control of distance and yaw angle; moreover, the vehicle slipping which is caused by difference rotation speed can be improved. Finally, through algorithm optimization, controller parameter adjustment, and system integration, an omnidirectional mobile vehicle with recognition and tracking functions is realized. The experiment results indicate that the system is stable and robust during actual operation.

Keywords
Face recognition, convolutional neural networks (CNNs), Mecanum wheel robot, proportional-integral-derivative (PID)

Date received: 24 December 2020; accepted: 7 June 2021

Introduction
In recent years, the development of deep learning technology has made the field of service robots and self-driving cars more diversified and popular. The recognizing and tracking of dynamic objects are important for robots and vehicles. With these technologies, we can perform some complex tasks such as home care,\textsuperscript{1} rescue,\textsuperscript{2} transport,\textsuperscript{3} bio-medical,\textsuperscript{4} and information,\textsuperscript{5,6} etc. Moreover, we can reduce the use of specialized sensors with the aid of images, and even let the machine autonomous.\textsuperscript{7}

In the early face recognition stage, it is often necessary to perform face detection before face recognition. Most of the face detection algorithms are used with manual Haar-like features\textsuperscript{8} or HOG (Histogram of Oriented Gradient)\textsuperscript{9} to perform features extraction, and then trained the classifier such as Adaboost (Adaptive Boosting)\textsuperscript{10} or SVM (Support Vector Machine)\textsuperscript{11} to realize face detection. After obtaining the face area, the image needs to be pre-processed, such as cropping, face alignment, noise removing etc., and finally face recognition is realized through matching similarity such as PCA (Principal Component Analysis)\textsuperscript{12} or LBPH (Local Binary Patterns Histograms).\textsuperscript{13} Although these traditional machine learning methods have good performance, the steps are complicated and the anti-interference ability of external factors is poor, so that they are not suitable for dynamic detection. Compared with traditional machine learning methods, the object detection methods based on deep learning are simple and perform better.

\textsuperscript{1}Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan
\textsuperscript{2}Department of Biomedical Engineering, Hungkuang University, Taichung, Taiwan
\textsuperscript{3}Department of Intelligent Commerce, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan

Corresponding author:
Yi-You Hou, Department of Intelligent Commerce, National Kaohsiung University of Science and Technology, No. 58, Shenzhong Road, Yanchao District, Kaohsiung 82444, Taiwan.
Email: yyhou@nkust.edu.tw

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
The requirements of certain applications and advances in technology have led to the development of object detection algorithms that are widely used in computer vision tasks such as face detection, face recognition, autonomous driving, and image labeling. Object detection is usually performed using a two-stage detector or one-stage detector. In two-stage detection, a model first proposes candidate object bounding boxes through a region proposal network and extracts features through region of interest pooling to classification and bounding-box regression tasks; an example of a model that employs two-stage detection is Faster R-CNN.\(^1\,4\,11\) In one-stage detection, a model proposes predicted boxes from input images directly without the region proposal step; examples of models that employ one-stage detection include SSD\(^1\,\text{6}\) and YOLO.\(^1\,\text{7}\) Two-stage detectors have high localization and classification accuracy but low inference speed, whereas one-stage detectors have high inference speed but a lower accuracy than two-stage detectors.

In order to achieve real-time detection performance on embedded systems with limited computing power, this paper uses one stage detector YOLOv3\(^1\,\text{9}\) algorithm to achieve face recognition, and some experimental optimizations are made for the YOLOv3 algorithm depend on the task, as shown in Table 1. Therefore, these optimizations eventually increase mAP by 8.5% on the custom data set. In addition, we deploy the trained model on the embedded systems and use deep learning accelerator to increase the operation speed by five times.

In this paper, we build a mecanum wheel vehicle with omnidirectional mobility and proposes a parallel-cascade PID architecture as the control system of the vehicle to achieve tracking function. The difference from the general PID architecture, the parallel-cascade PID architecture is allowed to input multiple control signals and has multi-loop, these characteristics make the system more effective in reducing the effects of disturbances and more stable; however, it is not easy to adjust parameters due to cascade architecture. The most important thing for a good tracking system is the immediacy and stability of operation. Therefore, the following experiments will focus on how to improve the accuracy of the model and good controller design and parameter adjustment.

### Methodology

#### YOLOv3

YOLOv3 was proposed by Redmon et al.\(^1\,\text{9}\). YOLOv3 has a fully convolutional network architecture, Darknet-53, inspired by ResNet. Using residual skip connections, we can solve the vanishing gradient problem and increase the depth of the network. For object detection, YOLOv3 uses a multiscale prediction method similar to that of a feature pyramid network (FPN),\(^\text{20}\) as shown in Figure 1. Shallower feature maps have higher resolution, which is conducive to localization; however, deeper feature maps have richer semantic information, which is conducive to classification. Therefore, an FPN combines these advantages and detects objects on three different scales, thereby mitigating the issues that make it difficult to detect small objects.

To allow a network to learn easily and achieve high detection accuracy, YOLOv3 inherits the YOLO9000\(^2\,\text{1}\) method of determining the anchor box and uses k-means clustering on the training set to automatically obtain good priors. The author selected nine clusters and evenly assigned these clusters to three scales for prediction using the YOLOv3 algorithm. The loss function of YOLOv3 consists of coordinate loss, confidence loss, and classification loss.

**Coordinate loss** is defined as follows:

\[
- \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{obj}(2 - \hat{w}_i \cdot \hat{h}_j) \left[ \hat{x}_i \ln x_i + (1 - \hat{x}_i) \ln(1 - x_i) \right] \\
- \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{obj}(2 - \hat{w}_i \cdot \hat{h}_j) \left[ \hat{y}_j \ln y_j + (1 - \hat{y}_j) \ln(1 - y_j) \right] \\
+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{obj}(2 - \hat{w}_i \cdot \hat{h}_j) \left[ (w_j - \hat{w}_i)^2 + (h_j - \hat{h}_j)^2 \right]
\]

where , \(B\), \(\hat{w}_i\), \(\hat{h}_j\), \(w\), and \(h\) are respectively the number of grids, number of bounding boxes predicted by each

| DB | KB | LB | MP | MC | GloU | DIoU | CloU | MB | HR | mAP@5:95 |
|----|----|----|----|----|------|------|------|----|----|---------|
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 72.62   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 72.76   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 74.02   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 75.14   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 73.42   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 79.20   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 80.66   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 79.86   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 81.12   |
| ✓  | ✓  | ✓  | ✓  | ✓  | ✓    | ✓    | ✓    | ✓  | ✓  | 82.24   |

DB: Default anchor box; KB: k-means anchor box; LB: Letterbox; MP: Mixup; MC: Mosaic; GloU: GloU loss; DIoU: DIoU loss; CloU: CloU loss; MB: Multi-anchor boxes; HR: High resolution.
grid, ground truth width, ground truth height, predicted width, and predicted height. In general, YOLOv3 treats object detection as a regression problem; it divides the image into an $S \times S$ grid and predicts $B$ bounding boxes for each grid cell. If the center of an object falls into a grid cell and its bounding box has the highest Intersection of Union (IoU) with the ground truth, that bounding box is responsible for detecting that object. $\text{obj}_{ij}$ denotes that the $j$th bounding box in cell $i$ is responsible for prediction, which means that coordinate loss will only penalty the bounding box responsible for detecting the object.

Confidence loss is defined as follows:

$$
\text{Confidence loss} = \frac{1}{C_0} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \left[ C_i \ln C_i + (1 - C_i) \ln(1 - C_i) \right]
$$

where $\hat{C}$ and $C$ are respectively the ground truth confidence score and predicted confidence score, respectively. Generally, the proportion of the background in an image is usually more than that of objects. Therefore, we set $\lambda_{\text{noobj}} = 0.5$ to decrease the influence of grids without objects on confidence loss to prevent model instability during training.

Classification loss is defined as follows:

$$
\text{Classification loss} = \frac{1}{C_0} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{\text{obj}} \sum_{c=\text{classes}} \left[ \hat{p}_c(c) \ln p(c) + (1 - \hat{p}_c(c)) \ln(1 - p(c)) \right]
$$

where $\hat{p}$ and $p$ are respectively the category probability of the ground truth bounding box and the category probability of the predicted bounding box, and the value of $c$ depends on the number of categories in the training dataset.

Through the above loss function, the gradient is calculated using stochastic gradient descent to update the network parameters and achieve end-to-end training. In summary, YOLOv3 achieves a good balance between speed and accuracy; however, the experiment results on the MS COCO dataset indicate that YOLOv3 performs poorly with medium and large objects, and the performance of mAP@0.75 is slightly inferior to other models.

**Improvement and training process**

Because the accuracy of the bounding box containing the object is the main focus, which means whether IoU is good enough. If the IoU is used as a coordinate loss function, it is modified to $1 - \text{IoU}$. However, the IoU has the advantage of scale invariance, which means that the similarity between two arbitrary shapes is independent of their size; the IoU though has the following drawback. First, if there is no overlap between the prediction and ground truth bounding boxes, the IoU is 0, which cannot reflect if two boxes are near each other or away from each other and does not provide any gradient for backpropagation, as shown in Figure 2(a). Second, in the case of the same IoU, the IoU does not reflect the manner in which two objects overlap, as shown in Figure 2(b).

Owing to the above shortcomings, a Generalized Intersection over Union (GIoU) was proposed by Rezatofighi et al.; the GIoU loss is defined as follows:
where $A$ and $B$ are two arbitrary convex shapes, which in this case are the predicted bounding box and the ground truth bounding box, respectively, and $C$ is the smallest box covering $A$ and $B$. The GIoU not only overcomes the shortcomings of the IoU but also retains the strengths of the IoU, as shown in Figure 3(a).

Based on the concept of the GIoU, Zheng et al.\textsuperscript{23} proposed the Distance Intersection over Union (DIoU) and showed that the GIoU has some shortcomings. When $C - (A \cup B) = 0$, the GIoU loss becomes an IoU loss and cannot converge well, as shown in Figure 3(b). The DIoU loss is defined as follows:

$$L_{\text{DIoU}} = 1 - \left( \frac{\text{IoU} - \frac{C - (A \cup B)}{C}}{C} \right)$$

where $a$ and $b$ denote the central points of $A$ and $B$, $\rho(\cdot)$ is the Euclidean distance, and $C$ is the diagonal length of the smallest enclosing box covering the two boxes. Because the DIoU loss simultaneously considers the central point distance and the overlapping area of the bounding boxes, it overcomes the problems associated with the GIOU, as shown in Figure 3(c). Zheng et al.\textsuperscript{23} also proposed the Complete Intersection over Union (CIoU) based on the DIoU. The CIoU loss is defined as follows:

$$L_{\text{CIoU}} = 1 - \left( \frac{\text{IoU} - \frac{\rho^2(a, b)}{C^2} - \alpha v}{v} \right)$$

where $\alpha$ is a positive trade-off parameter, and $v$ is the similarity between the aspect ratios of the bounding boxes. The CIoU considers not only the distance from the center point but also the aspect ratios of the bounding boxes, resulting in faster convergence and better performance, as shown in Figure 3(d).

To make the model more robust, we employed two data augmentation methods during training. The first one is Mixup,$\textsuperscript{24}$ which multiplies two images and superimposes them with different coefficient ratios to increase image semantics and prevent overfitting, as shown in Figure 4(b). The other method is Mosaic,$\textsuperscript{25}$ which randomly crops an area of four images and stitches them into one image, as shown in Figure 4(c). This method mixes four training images, whereas Mixup only mixes two input images. The image semantics achieved with Mosaic are richer than those achieved with Mixup. However, using the four-image mosaic instead of a single image during training reduces the need for large batches.

Faces are long, thin, and rectangular. To maintain the original aspect ratio during the image resizing

---

*Figure 2.* (a) Drawback of the IoU. (b) representation of $\text{IOU}=0.7$.

*Figure 3.* (a) Representation of the GIoU, (b) drawback of the GIoU, (c) representation of the DIoU, and (d) drawback of the CIoU.
process, the letterbox resize method is adopted; this method prevents the deformation of an object, as shown in Figure 5(c).

In YOLOv3, k-means clustering is used to obtain good priors, which allows the network to learn easily and achieve high detection accuracy. In this study, we also chose nine anchor boxes just like in a previous study; we applied k-means clustering to our custom dataset. On our custom dataset, the nine clusters were (14,19), (19,29), (29,41), (47,46), (39,67), (63,66), (60,109), (93,87), and (118,109). As seen in Figure 6, most of the boxes were tall and thin, just like human faces. Furthermore, to increase accuracy, we used multiple anchor boxes for a single ground truth instead of a single anchor box for a single ground truth during training.

The dataset used in this study consisted of 700 face images collected by us; 750 bounding boxes were manually labeled in three categories. To ensure even distribution of data, we ensured that the number of bounding boxes in each category was equal. Because the amount of training data was less, we used random scaling, cropping, and flipping to prevent overfitting during the training process. In addition, we used the Darknet-53 pretraining model weights that were trained on ImageNet as the initial weights for training to ensure stability during the training process and achieve fast convergence.

**System realization and integration**

**Hardware architecture**

The development kit used in this study was Nvidia Jetson Nano. Jetson Nano is a small, powerful
computer that is based on a Maxwell architecture with 128 NVIDIA CUDA cores and delivers a computing performance of 472 GFLOPS; moreover, the development boards contain a 40-pin GPIO header. All these features rendered the Jetson Nano suitable for our task.

The mobile vehicle used in this study was a Mecanum wheel. Each wheel consisted of many sub-wheels arranged at a 45° angle around the wheel axis, as shown in Figure 7(a). The direction of rotation and the speed of each wheel allow the vehicle to move omnidirectionally and attain a higher number of degrees of freedom during operation, as shown in Figure 7(b).

Each Mecanum wheel was driven by a JGB37-520 brushed DC motor with a Hall sensor and had a TB6612FNG dual motor driver to control the motor. During movement, the current motor speed was calculated using the signal from the Hall sensor, and the speed information was used for movement control.

For frame capturing, a Logitech C310 webcam was used. This webcam captures images with 1280 × 720 pixels and records videos in 720 p. A gimbal, which consisted of an MG90S servomotor installed to the webcam, and a PCA9850 servomotor driver, which was used to control the servomotor, were used for face tracking.

Because the Mecanum wheel robot is sensitive to the torque of the wheel, it may slip owing to the difference in the motor speed during movement. To prevent slipping, we installed an MPU6050 gyroscope on the vehicle and used Arduino Due to receive yaw angle information through an I2C bus to monitor the vehicle when it moves.

The power system consisted of a 5in1 V3 power hub and a 12 V four-cell LiPo battery. The power hub had a linear regulator function to keep the output power stable and provided output voltages of 12 and 5 V to the motor drivers, Jetson Nano, Arduino Due, and other hardware devices. The system is equipped with a low-voltage alarm function to remind users when the battery is about to die to prevent the sudden shut down of the system because of a dead battery.

The Mecanum wheel robot along with the hardware components used in this study is shown in Figure 8; the dimensions of the structure shown in the figure are 26 × 21 × 31 (cm).
Controller design and simulation

To make the system stable and efficient, we need to mathematically model the system before control.26–28

The derivation of the dynamic equations of motion is presented next. The kinematics model of a Mecanum wheel robot is shown in Figure 9.

Let \( f_1, 2, 3, 4, u_1, 2, 3, 4, n_k, n_b, R, \) and \( R_a \) be the force of each motor, the input voltage to each motor, the angular velocity of each wheel, wheel radius, and armature resistance, respectively. The force vector is written as follows:

\[
\begin{bmatrix}
    f_1 \\
    f_2 \\
    f_3 \\
    f_4 \\
\end{bmatrix} = \frac{n_k}{R} \begin{bmatrix}
    u_1 \\
    u_2 \\
    u_3 \\
    u_4 \\
\end{bmatrix} - \frac{n^2 K_b^2}{R R_a} \begin{bmatrix}
    \omega_1 \\
    \omega_2 \\
    \omega_3 \\
    \omega_4 \\
\end{bmatrix}
\]  

(7)

The dynamic equation of motion is \( \ddot{X} = A \dot{X} + BU \), and \( V_1, 2, 3, 4 \) is the velocity of the wheels; when the vehicle moves, each wheel outputs the velocity at an angle of 45°. The dynamic equations of motion are derived using Newton’s second law of motion and the relationship between torque and angular acceleration, as shown below:

\[
\begin{bmatrix}
    \dot{X}_G \\
    \dot{Y}_G \\
    \dot{\theta} \\
\end{bmatrix} = HT_{M}^G K \begin{bmatrix}
    f_1 \\
    f_2 \\
    f_3 \\
    f_4 \\
\end{bmatrix} = M \begin{bmatrix}
    f_1 \\
    f_2 \\
    f_3 \\
    f_4 \\
\end{bmatrix}
\]  

(8)

where

\[
H = \begin{bmatrix}
1/m & 0 & 0 \\
0 & 1/m & 0 \\
0 & 0 & 1/L_z \\
\end{bmatrix},
\]

\[
T_{M}^G = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
K = \begin{bmatrix}
\cos(45^\circ) & \cos(45^\circ) & \cos(45^\circ) \\
-\sin(45^\circ) & \sin(45^\circ) & -\sin(45^\circ) \\
-L \sin(45^\circ - \varphi) & L \sin(45^\circ - \varphi) & L \sin(45^\circ - \varphi) \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\omega_3 \\
\omega_4 \\
\end{bmatrix} = \frac{1}{R} \begin{bmatrix}
V_1 \\
V_2 \\
V_3 \\
V_4 \\
\end{bmatrix}
\]

(9)

Using equations (7)–(9), the dynamic equations are expressed as follows:

\[
\begin{bmatrix}
\dot{X}_G \\
\dot{Y}_G \\
\dot{\theta} \\
\end{bmatrix} = \frac{n_k}{R R_a} M \begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
\end{bmatrix} - \frac{n^2 K_b^2}{R R_a} MN \begin{bmatrix}
\dot{X}_G \\
\dot{Y}_G \\
\dot{\theta} \\
\end{bmatrix}
\]

(10)

Therefore, \( X = [X_G, Y_G, \theta]^T \), \( U = [u_1 \ u_2 \ u_3 \ u_4]^T \), \( A = -\frac{n^2 K_b^2}{R R_a} M \), and \( B = \frac{n_k}{R R_a} \).

\[
\theta \text{ is the rotation angle of the vehicle, } L \text{ is the distance between the vehicle centroid and the wheel centroid, and } \varphi \text{ is the angle between the vehicle centroid and the wheel centroid.}
\]

The relationship between the velocity of the vehicle and the velocity of wheels is given by:

\[
u(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k - 1)}{\Delta t}
\]

(11)

In this study, we use the PID control algorithm to design the controller; A general discrete-time PID controller is represented by:

\[
u(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k - 1)}{\Delta t}
\]
where \( u(k) \) is the control input signal, \( e(k) \) is the error at step \( k \), \( \Delta t \) is the sampling, and \( K_p, K_i \), and \( K_d \) are the parameters of the discrete-time PID controller. To realize the functions of tracking and slip correction, we proposed a parallel-cascade PID controller, whose architecture is shown in Figure 10. This architecture is mainly divided into three parts: motor velocity control, distance control, and yaw angle control. The adjustment of PID parameters is reference to the Ziegler-Nichols\(^{29}\) method. After observing the experimental results and making suitable fine-tuning according to the characteristics of each parameter. In the process of parameter adjustment, we need to adjust the parameters of the secondary loop to make the system stable before you further adjust the primary loop; therefore, make sure the response time of the secondary loop is faster than the primary loop, so that cascade control can be performed.

First, the secondary loop of the parallel-cascade PID controller is responsible for controlling the motor velocity. The inner loop is executed by Arduino Due, and its sampling time is 50 ms. Each motor has its own PID parameters, that is, there are four sets of PID controllers in the inner loop of the parallel-cascade PID controller. After many experiments and adjustments, the four sets of parameters of the PID controller are as follows:

\[
\begin{align*}
K_p^1 &= 1.75, & K_i^1 &= 0.1, & K_d^1 &= 2.7; \\
K_p^2 &= 0.12, & K_i^2 &= 3.1 \\
K_p^3 &= 1.78, & K_i^3 &= 0.15, & K_d^3 &= 3.1; \\
K_p^4 &= 0.1, & K_i^4 &= 3
\end{align*}
\]

Because the parameters of each motor are slightly different, the parameters of the PID controller for each motor are different. The results of velocity control are shown in Figure 11. The results show that the system has a fast and stable response without any overshoot.

The primary loop 1 loop of the parallel-cascade PID controller is responsible for yaw angle control of the vehicle, which prevents the vehicle from slipping due to the difference in motor speed during movement. The outer loop is also executed by Arduino Due, and its sampling time is 50 ms. The parameters of the PID controller are as follows:

\[ K_p = 3.1, \quad K_i = 0.12, \quad K_d = 1.35 \]

The results of yaw angle control are shown in Figure 12. Figure 12(a) shows the results without the controller. As the vehicle moves, the yaw angle gradually increases, which indicates that the vehicle is slipping, as shown in Figure 12(b). The result after adding the controller is shown in Figure 12(c), the yaw angle is continuously corrected within a range of \( \pm 0.5^\circ \), as shown in Figure 12(d).

In addition, we provided external disturbances during vehicle movement to test the robustness of the system, as shown in Figure 13(a); and recorded the change of the yaw angle and each motor speed of the vehicle during the process, as shown in Figure 13(c) and (d).
According to the result of Figure 13(c), the controller can correct external disturbances immediately, as shown in Figure 13(b). The above test results can prove that the system has good robustness.

The primary loop 2 loop of the parallel-cascade PID controller is responsible for distance control. The outer loop is executed by Jetson Nano, and its sampling time is 160 ms. The distance between an object and the vehicle is calculated using the triangular geometric distance measurement method. The parameters of the PID controller are as follows:

\[ k_p = 2.45, \quad k_i = 0.155, \quad k_d = 0.15 \]

The results of distance control are shown in Figure 14. 150 cm must be maintained between the object and the vehicle. When the object is moving, the vehicle tracks the object through feedback control and maintains a fixed distance from the object. During the movement of the vehicle, the distance between the object and the vehicle is proportional to the vehicle velocity, thereby overcoming the problem of the vehicle being sensitive to the boundary of the target distance.

**Experiment results and discussion**

In this study, we used an Nvidia GTX 1080 Ti GPU to train the model through the Darknet framework in an Ubuntu 18.04 environment. During training, we use the SGD with a batch size of 16, the momentum and weight decay are respectively set as 0.95 and 0.005, and adopt batch normalization. The learning rate is set as 0.001 and divide it by 10 at 16k and 18k iterations, and terminate training at 20k iterations. The experimental results are shown in Table 1, all the network input resolution is 288 * 288 except for items with HR, which is 416 * 416. Obviously, a series of optimization methods were used to improve mAP by 8.5% on the custom data set.

In the Table 1, the most significant improvement is the optimization of the coordinate loss function, which uses DIOU loss to replace the mean square error and cross-entropy loss, it confirms that the model cares more about the performance on IoU rather than the scale of the bounding box. The use of K-means anchor Box and Letter-Box resize scheme has increased the mAP by 1.6%, which is optimized for the task, so this effect is predictable. The data argumentation methods
of Mixup and Mosaic can make the training data have rich image semantics and avoid overfitting. However, the Mosaic method decreased mAP by 0.6%, we think the reason is that mosaic mixes four input images to make the objects smaller. In addition, human faces are usually small in the images, so that the model cannot handle them. Increasing the data resolution in the training process can make the image information richer, but the relative execution efficiency will be affected, so this is a trade-off that must depend on the task.

Finally, considering execution speed, we selected the result with the highest mAP@.5:.95 at a resolution of 288 × 288 as the face recognition model as shown marked in red in Table 1. The experiment results obtained with the face recognition model are shown in Figure 15(a). The model does not get confused when there are several faces; it accurately determines the location of the object and its category. Figure 15(b) shows the result of real-time detection and distance calculation obtained using the Logitech C310 camera.

If the face recognition model is inferred on Jetson Nano using OpenCV, its computational efficiency would not be suitable for real-time detection tasks; therefore, we optimized the computational efficiency using TensorRT. TensorRT is a C++ library from NVIDIA; it is used for high-performance inference on NVIDIA GPUs and deep learning accelerators. The inference speed after acceleration is shown in Figure 16. As seen in the figure, the inference speed with TensorRT was more than five times that with OpenCV. Therefore, TensorRT was used to optimize the model. Jetson Nano was used to deploy the model in this study. The use of a more powerful embedded system such as Jetson AGX Xavier can certainly increase computational efficiency but the high cost is contrary to our low-cost purpose.

To avoid losing the target during the tracking process, we used the PID controller to control the horizontal and vertical rotation of the two-axis servo gimbal, which is shown in Figure 17(a). The pink and blue lines in the frame indicate the horizontal and vertical changes in the target and camera, respectively. When the target moves, the camera follows the target to ensure that the target is in the center of the frame to achieve face tracking. Because three categories are present in our custom dataset, the same effect is achieved by changing the tracking targets, as shown in Figure 17(b).

The system flowchart is shown in Figure 18. First, the camera inputs the image into the model to determine whether the object exists, after which it calculates the distance between the vehicle and the target as well as the yaw angle of the vehicle. The velocity of each

![Figure 12. Result of yaw angle control. (a) The result without yaw angle controller, (b) the schematic diagram of vehicle slipping, (c) the result with yaw angle controller, and (d) the schematic diagram of correcting vehicle slipping.](image-url)
motor is calculated through the outer loop of the parallel-cascade PID controller, and the UART communication protocol is used for data transmission. Finally, the inner loop of the parallel-cascade PID controller controls the velocity of each motor to realize object tracking and slip correction. The other process is to calculate the center coordinates of the target, and then use the PID controller and servomotor to realize face tracking.

In summary, the result of the system is shown in Figure 19. As the target moves, the vehicle will adjust the moving speed according to the distance and maintain a fixed distance from the target, and correct the vehicle slip during the tracking process.

**Conclusion and discussion**

In this study, we developed an object detection algorithm based on convolutional neural networks. We trained a face recognition model with both accuracy and efficiency using our custom dataset and improved the recognition accuracy through algorithm optimization. We used the concept of edge computing to deploy models on local embedded systems and were able to increase the model inference speed using deep learning accelerators. To increase the stability and robustness of the system, we developed a parallel-cascade PID controller architecture for a Mecanum wheel vehicle. The controller used the distance between the vehicle and the object, the yaw angle, motor speed, and good parameter adjustment to ensure that the vehicle tracked the object from a fixed distance tracking and corrected for
vehicle slip during vehicle movement. On the camera, we built a two-dimensional servo gimbal and used feedback control of the PID controller, which allowed the camera to rotate horizontally and vertically, to realize face tracking. Finally, through the integration of software, firmware, and hardware, an omnidirectional unmanned mobile vehicle tracking system with recognition and tracking functions was achieved.

Figure 15. Results of face recognition model. (a) Test result of model and (b) result of real-time detection and distance calculation.

Figure 16. Inference speed comparison of OpenCV and TensorRT on Nvidia Jetson Nano.

Figure 17. Results of face tracking. (a) Result of face tracking with PID controller and (b) result of face tracking using PID controller for different targets.
Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was financially supported by the Ministry of Science and Technology, Taiwan, under grant MOST 107-2221-E-006-222, MOST 110-2218-E-006-014-MBK and MOST 111-2218-E-006-009-MBK.

ORCID iD

Yi-You Hou https://orcid.org/0000-0002-8694-0313

References

1. Portugal D, Santos L, Alvito P, et al. SocialRobot: an interactive mobile robot for elderly home care. In: 2015 IEEE/SICE International Symposium on System Integration (SII), 11-13 December 2015, Nagoya, Japan pp. 811–816.
2. Al-Kaff A, Moreno FM, de la Escalera A, et al. Intelligent vehicle for search, rescue and transportation

Figure 18. System flowchart.

Figure 19. Execution results of system. (a) The actual result of keeping a fixed distance from the object during tracking and (b) tracking demonstration of moving target.
purposes. In: 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), 11-13 October 2017, Shanghai, China, pp.110–115.
3. Alonso-Mora J, Baker S and Run D. Multi-robot formation control and object transport in dynamic environments via constrained optimization. Int J Rob Res 2017; 36(9): 1000–1021.
4. Altan A and Parlak A. Adaptive Control of a 3D Printer using Whale Optimization Algorithm for Bio-Printing of Artificial Tissues and Organs. 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), November 2020.
5. Altan A and Karasu S. Recognition of COVID-19 disease from X-ray images by hybrid model consisting of 2D curvelet transform, chaotic SALP swarm algorithm and deep learning technique. Chaos Solitons Fractals 2020; 140(110071): 110071–110110.
6. Karasu S, Altan A, Bekiros S, et al. A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. Energy 2020; 212(118750): 1–13.
7. Altan A. Performance of Metaheuristic Optimization Algorithms based on Swarm Intelligence in Attitude and Altitude Control of Unmanned Aerial Vehicle for Path Following. In: 4th International Symposium on Multidisciplinary Studies and Innovative Technologies, 22-24 October 2020, Istanbul, Turkey, pp. 1–7.
8. Viola P and Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, 8-14 December 2001, Kauai, HI, USA, pp. 511–518.
9. Dalal N and Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 20-25 June 2005, San Diego, CA, USA, pp. 886–893.
10. Sharifara A, Mohd Rahim MS and Anisi Y. A general review of human face detection including a study of neural networks and Haar feature-based cascade classifier in face detection. In: 2014 International Symposium on Biometrics and Security Technologies (ISBAST), 26-27 August 2014, Kuala Lumpur, Malaysia, pp. 73–78.
11. Guo G, Chan SZ and Li K Face recognition by support vector machines. In: Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR003580), 28-30 March 2000, Grenoble, France, pp. 196–201.
12. Zhao W, Krishnaswamy A, Chellappa R, et al. Discriminant analysis of principal components for face recognition, Face Recognition. Vol. 163. Berlin, Heidelberg: Springer, 1998. pp.73–85.
13. Ahonen T, Hadid A and Pietikäinen M. Face description with local binary patterns: application to face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006; 28(12): 2037–2041.
14. Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017; 39: 1137–1149.
15. Altan A, Karasu S and Zio E. A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. Appl Soft Computing J 2021; 100(106996): 1–20.
16. Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector. In: Proceedings of the European Conference on Computer Vision (ECCV), 11-14 October 2016, Amsterdam, Netherlands, pp. 21–37.
17. Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27-30 June 2016, Las Vegas, NV, USA, pp. 779–788.
18. Redmon J and Farhadi A. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 8 April 2018.
19. He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27-30 June 2016, Las Vegas, NV, USA, pp. 315–323.
20. Dollár P, Appel R, Belongie S, et al. Fast feature pyramids for object detection. IEEE Trans Pattern Anal Mach Intell 2014; 36(8): 1532–1545.
21. Redmon J and Farhadi A. YOLO9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 21-26 July 2017, Honolulu, HI, USA, pp. 7633–7671.
22. Rezatofighi H, Tsoi N, Gwak JY, et al. Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 15-20 June 2019, Long Beach, CA, USA, pp. 658–666.
23. Zheng Z, Wang P, Liu W, et al. Distance-IoU loss: faster and better learning for bounding box regression. AAAI 2020; 34: 12993–13000.
24. Zhang H, Cisse M, Dauphin YN, et al. Mixup: beyond empirical risk minimization. In: 6th International Conference on Learning Representations, 30 April 2018, Vancouver, BC, Canada, pp. 1–13.
25. Bochkovskiy A, Wang CY and Liao HYM. YOLOv4: Optimal Speed and accuracy of object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 23 April 2020, Ithaca, NY, USA, pp. 1–17.
26. Shimada A, Yajima S, Viboonaachaipheep P, et al. Mecanum-wheel Vehicle Systems Based on Position Corrective Control. In: 31st Annual Conference of IEEE Industrial Electronics Society, 6-10 November 2005, Raleigh, NC, USA, pp. 1–6.
27. Alakshendra V and Chiddarwar SS. A robust adaptive control of mecanum wheel mobile robot: simulation and experimental validation. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 9-14 October 2016, Daejeon, Korea, pp.5606–5611.
28. Gu D. Development of Wiimote indoor localization technology and Omni-directional vehicle trajectory control for intelligent. Department of Mechanical Engineering National Cheng Kung University, Tainan, Taiwan, 2011.
29. Meshram P and Kanojiya RG. Tuning of PID controller using Ziegler-Nichols method for speed control of DC motor. In: Proceedings of the IEEE ICAESM, 30-31 March 2012, Nagapattinam, India, pp.117–122.
30. Huang JP. *Realization of depth estimation from monocular camera based on defocus algorithm and reverse heat equation*. Department of Engineering Science National Cheng Kung University, Tainan, Taiwan, 2017.

31. Darknet framework. https://github.com/AlexeyAB/darknet (Accessed 5 January 2019)

32. OpenCV Deep Neural Network module (dnn module). https://docs.opencv.org/4.2.0/d6/d0f/group__dnn.html (accessed 20 January 2019)

33. NVIDIA TensorRT. https://developer.nvidia.com/tensorrt (accessed 3 March 2019).