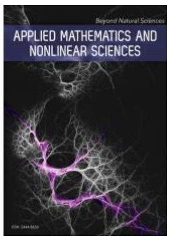


Applied Mathematics and Nonlinear Sciences

https://www.sciendo.com

Research on Vehicle Detection Algorithm Based on Embedded ARM

Yueming Deng[1,†], Dan Deng[1]

1. Guangzhou Panyu Polytechnic, Guangzhou, Guangdong, 511483, China.

Submission Info

Communicated by Z. Sabir
Received July 2, 2024
Accepted October 9, 2024
Available online November 11, 2024

Abstract

Based on the theory of machine learning in the field of artificial intelligence, this paper proposes to use the computer vision platform OpenCV to construct an embedded ARM vehicle detection model. Determine the ARM embedded software and hardware and adopt Haar features for the Adaboost algorithm to design the OpenCV vehicle classifier. Cross-compile the ARM chip using Linux to generate new firmware for OpenMV. Use the DFU tool for embedded ARM chips to upgrade and re-burn them into the embedded development board for machine vision OpenMV. By using the classifier file and OpenCV's image processing algorithm, the work of vehicle detection and recognition is completed, and the vehicle target is labeled with a candidate box in the picture and video. The results demonstrate that the algorithm in this paper maintains the leakage detection rate and false detection rate below 5% in four different working conditions: strong light, normal light, weak light, and nighttime, thereby fully demonstrating the effectiveness of the research conducted in this paper.

Keywords: OpenCV; Embedded ARM; Adaboost algorithm; Haar features; Vehicle detection.
AMS 2010 codes: 65Y20

†Corresponding author.
Email address: dym3212758@139.com

ISSN 2444-8656
https://doi.org/10.2478/amns-2024-3133

1 Introduction

Intelligent Transportation System, also known as ITS, was created with the development of economic globalization and information technology. It mainly refers to the traffic management system that combines computers with image processing technology, vision technology, electronics, communication, automation and other technologies that have emerged in recent years [1-2]. It can realize a kind of real-time, efficient, accurate larger scope and no blind spot traffic intelligent management system. It has an irreplaceable role in effectively improving the efficiency of transportation, ensuring the safety of transportation, and promoting the sustainability of development [3-4]. In recent years, researchers have made great contributions to the study of intelligent transportation in China, and the results are remarkable. The vision-based vehicle detection in vehicle identification and tracking technology has played a great role in promoting the intelligence of China's traffic management system [5-6]. Since vision-based vehicle detection and recognition technology has irreplaceable superiority over other methods, it is of great value in terms of application value and scientific research theory, and the market prospect is very broad [7-8].

In the traditional image processing system, PCs occupy a major position, but due to the large size of the PC itself, poor portability, and poor stability in the outdoor long-time continuous operation, limiting the scope of its application [9]. At present, there are also many image-processing platforms based on other systems, most of which use microcontrollers, DSPs, FPGAs, ARMs, etc., as central processors. Among these embedded platforms, ARM-based embedded platform systems have been gradually used more in image processing systems due to their relatively low development cost, good stability, fast computing speed and other characteristics [10-11]. With the continuous development of the field of artificial intelligence, the vehicle detection algorithm based on embedded ARM has a substantial improvement compared with the traditional method. With the gradual deepening of the research of embedded ARM and driverless cars, how to use embedded ARM to quickly and accurately realize the vehicle detection algorithm of driverless cars and improve the ubiquity of the detection algorithm will be the focus of this paper [12-14].

Vehicle detection is one of the research topics of target detection and an important branch of digital image processing tasks. Literature [15] improved the YOLO v5 network algorithm using the flip tessellation algorithm and applied it to the vehicle detection method in order to improve the accuracy of vehicle detection in different traffic scenarios and to reduce the false detection rate of occlusion on vehicle targets. Literature [16] proposed a two-step detection algorithm by combining Harr and HOG features and applied it to the detection and tracking performance experiments of multi-vehicle targets, and the results verified the validity of the method, which can improve the target detection accuracy and time efficiency. Literature [17] mainly uses test images closer to actual road conditions to compare and evaluate the detection accuracy of five mainstream deep learning target detection algorithms, namely RCNN, R-FCN, SSD, RetinaNet and YOLOv3. Literature [18] focuses on examining the robustness and efficiency of vehicle detection in intelligent transportation systems and discusses the effectiveness of vehicle detection algorithms in different environments through a literature review.

In addition, literature [19] constructed a real-time vehicle recognition system based on a low-cost embedded ARM platform and an image processing algorithm for mobile vehicle detection and verified the system's superior performance through testing, which can realize real-time vehicle detection, speed measurement and license plate recognition. Literature [20] proposed an embedded lightweight vehicle detection algorithm by combining improved YOLOv3-tiny, spatial pyramid pooling structure, kmean++ clustering algorithm and Generalized IoU loss function, and experimental tests found that the proposed algorithm embodies real-time performance and accuracy in multi-vehicle target detection. Literature [21] based on the YOLOv5 model proposed a real-time effective and can be deployed on the satellite platform remote sensing image vehicle detection method. The test results show that the method can provide technical support for satellite detection in orbit. Literature [22] proposed a real-time vehicle detection scheme using a low-power embedded graphics processor (GPU), which contains three schemes, namely loose unfolding, tight unfolding and hybrid unfolding, and the feasibility of the proposed scheme is verified through experimental comparisons on the dataset, which improves the efficiency of vehicle detection.

This paper combines Haar features and the Adaboost algorithm to build a vehicle detection classifier, which is then rewritten in a Linux/Ubuntu environment to generate new firmware for OpenMV. The DFU tool for embedded ARM chips is utilized for upgrading and re-burning the embedded development board of OpenMV for machine vision. In this paper, the code editing of software test cases is completed using the VisualStudio platform and the OpenCV algorithm function library. By using the classifier file and OpenCV's image processing algorithm, the work of vehicle detection and recognition is completed, and the vehicle target is labeled with a candidate box in the picture and video. Create a dataset and evaluation indexes to evaluate the algorithm presented in this paper both experimentally and application-wise.

2 Embedded software, hardware-related development platforms

2.1 Embedded Software Platform

2.1.1 Visual Studio development environment

Visual Studio is composed of a set of component-based development tools developed by Microsoft, which also includes several other technologies for generating powerful, high-performance applications [23]. In addition, Visual Studio is optimized for team-based design, development, and deployment of enterprise solutions.

2.1.2 Open CV, a computational vision platform

OpenCV, the "Open" and "Computer Vision" in its name, represent open source and computer vision, respectively, which together constitute this powerful open-source computer vision library [24]. OpenCV has become an important tool in the field of computer vision by virtue of its open source, unified programming style, optimized code, convenient API interface, fast image processing capability, and multi-platform support, which provides strong support for video image processing tasks such as motion detection and tracking.

2.1.3 Visual Integrated Development Environment IDE

Applying the MVC pattern, the IDE can be divided into several sub-modules, i.e., Glyph Model, Glyph Editor, Glyph Portrayer, Property Grid, Glyph Browser, Glyph model, Project Browser, Code Editor and IDEUI.

2.1.4 Compilation platforms and tools

The command line is used to install the Linux Binscope Security Options Detection Tool, which is compatible with all versions of Linux. In the current mainstream Linux distributions, there are SUSE Linux 10 32-bit version, SUSE Linux 11 64-bit version, Wind River Linux 32-bit, Wind River Linux

64-bit version, etc., while in this paper, Wind River Linux 64-bit version is selected to study the embedded ARM-based Vehicle Detection.

2.1.5 Programming Language Micro Python

Python is an open-source, free, easy-to-learn, and easy-to-use programming language with a powerful standard library and a wide range of application areas, including web development, artificial intelligence, data processing, and more. Its simple, straightforward syntax rules and expressions have made it one of the programming languages preferred by developers, and it is also widely used in the development of well-known applications and services, such as Dropbox, YouTube, Instagram, etc. Python is a language that can be developed much faster and can be iterated quickly to meet the needs of rapid development.

2.1.6 DFU Program Burning

Embedded program burning modes are JTAG, SWD, and DFU. The DFU program encompasses the microcontroller DFU Demo code, PC-side upgrade program, PC-side Demo code, and related information manuals, and so forth. The use of the DFU program not only makes it possible to quickly upgrade the functions of the developed product, but also to update the corresponding upgrade program.

2.2 Embedded Hardware Platform

2.2.1 Machine vision module

The machine vision module uses OpenMV, a new low-power smart camera designed to be the “Arduino of machine vision”, to create scalable machine vision modules. The OpenMV module can be very well used in machine vision and is based on the STM32H743VI ARM Cortex M7 and OV7725 camera as the basis for the machine vision module. For the OpenMV underlying driver and image processing library part, some OpenCV libraries are mainly referenced in this system design and encapsulated in C language.

2.2.2 Image sensing unit

In order to realize the image acquisition unit for sample image acquisition and grading after the acquisition, there is a sensing unit installed near the detection position and grading position to realize the judgment of sample position and signal presence or absence. On the front of the CCD camera is where the detection position sensing unit is mounted among others. A sensing unit for measuring the grade is installed in the grading unit. The sensor that is utilized is the E18-F10NK red color sensor. In the case of the sample color and the background color, a slight difference in the situation can also be detected and the sample color in the device is green or yellow, and the background color is white.

2.2.3 ARM chips

The native 64-bit instruction set is already supported by the latest ARM chips. In addition to the processor architecture upgrade, the ARM processor process is also becoming more advanced with the progress of industrial technology. The latest ARM processor has been used with a 7nm process [25]. Now the ARM chip has developed a number of series of many models. Common ARM processor performances from low to high are ARM7 series, ARM9 series, ARM9E series, ARM10E series, etc., which are widely used in mobile devices, vehicle testing program development, home appliances and factory production and other fields because it has a high performance, low cost and low energy consumption and other characteristics, is widely used in cell phones Because of its high performance, low cost and low power consumption. It is widely used in a variety of terminals and application scenarios, such as mobile phones, laptops, and servers.

2.2.4 Data Interaction Module Unit

An SDDataStruct class is designed and created by the data interaction module, which inherits the ISDDataStruct class and is primarily used to open the TCP server function. The data interaction module is designed for remote communication with the onboard data processing unit and designs and creates a SmartDevice class, which inherits the QObject class and mainly realizes the data interaction function with the onboard data processing unit. The module is designed to create a SmartDevice class, which inherits from the QObject class and mainly enables data interaction with the on-board data processing unit. This module is also responsible for processing and forwarding command and data messages sent from the ground server to the in-vehicle gateway, and ensuring that they are correctly parsed and delivered to the target device.

3 Offline training and generation of OpenCV-based vehicle classifiers

3.1 Training of weak vehicle classifiers based on Haar features

3.1.1 Haar Characterization

Haar feature involves inputting rectangular features of an image to obtain image features of a specific target. The principle of the Haar feature is to change the distribution of pixels by intercepting the gray level of a specific region in an image. For example, in grayscaled images, the areas on both sides of the eyes and the bridge of the nose of a person are darker than the other areas of the face. In a real-time road condition monitoring system based on an ARM chip, using the positive samples of the training model, for example, vehicle pictures at different angles, there are many differences in the shadows of different models. The body and tire size ratio is also different for different models; the tangent area of the windshield and the ratio of the entire front end are also different for different models at different angles, which are considered different Haar features for different targets. Various rectangular features as shown in Figure 1, there are three main common rectangular features, which are two-rectangular, three-rectangular features and four-rectangular features.

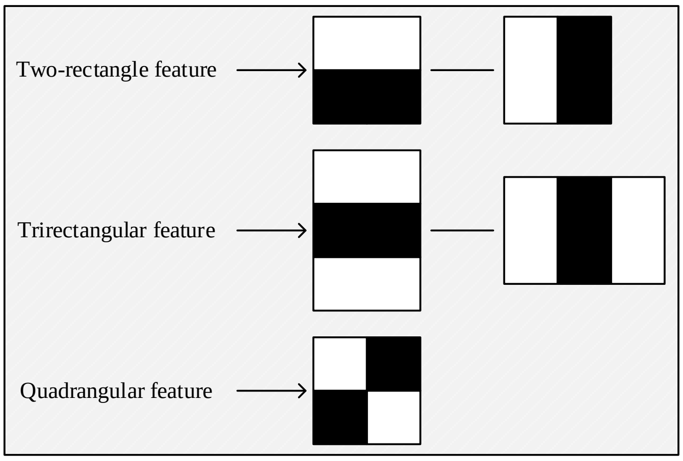


Figure 1. Various rectangular features

The detection window is shown in Fig. 2 and the number of internal rectangular features is too large for a common detector, and the arithmetic of the rectangular features needs to be calculated. The calculation method is described below. In a window of size $m*m$, only the upper left vertex $A(x_l, y_l)$ and the lower right vertex $B(x_2, y_2)$ of the rectangle need to be determined, i.e., a rectangle can be determined if this rectangle must also satisfy the following two conditions (known as the (s,t) condition and a rectangle that satisfies the (s,t) condition is known as a conditional rectangle).

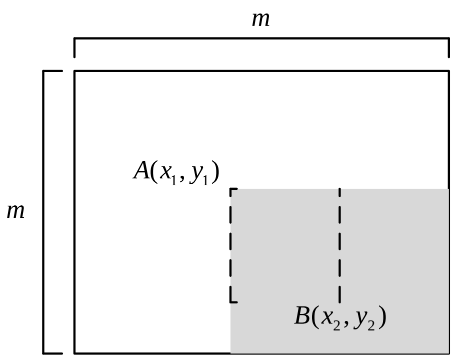


Figure 2. Detection window

1) The length of the side in direction x must be divisible by the natural number s (equally divided into s segments).
2) The length of the side in direction y must be divisible by the natural number t (equally into t segments). The minimum size of the rectangle is $s\times t$ or $t\times s$, and the maximum size is $[m/s]\cdot s\times[m/t]\cdot t$ or $[m/t]\cdot t\times[m/s]\cdot s$; where [] is the rounding operator.

Determine $A(x_i, y_i): x_i \in \{1, 2, \cdots, m-s, m-s+1\}, y_i \in \{1, 2, \cdots, m-t, m-t+1\}$

After obtaining point A, point B can only be taken in the shade, so there:

$$\begin{aligned}
&x_2 \in X = \{x+s\,|-1, x+2\cdot s-1, \cdots, x+(p-1)\cdot s-1, x+p\cdot s-1\} \\
&y \in Y = \{y+t-1, y+2\cdot t-1, \cdots, y+(q-1)\cdot t-1, y+q\cdot t-1\} \\
&p = \frac{m-x+1}{s}, q = \frac{m-y+1}{t}
\end{aligned} \tag{1}$$

This yields the number of all rectangles in the $m\times m$ sub-window that satisfy the (s,t) condition:

$$\begin{aligned}\Omega_{(s,t)}^{m} &= \sum_{x=1}^{m+s+1}\sum_{y=1}^{m-t+1} p\cdot q \\ &= \sum_{x=1}^{m-s+1}\sum_{y=1}^{m-t+1}\left[\frac{m-x_i+1}{s}\right]\cdot\left[\frac{m-y_i+1}{t}\right] \\ &= \sum_{x=1}^{m-s+1}\left[\frac{m-x_1+1}{s}\right]\cdot\sum_{y=1}^{m-t+1}\left[\frac{m-y_1+1}{t}\right] \\ &= \left(\left[\frac{m}{s}\right]+\left[\frac{m-1}{s}\right]+\cdots+\left[\frac{s+1}{s}\right]+1\right)\cdot\left(\left[\frac{m}{t}\right]+\left[\frac{m-1}{t}\right]+\cdots\left[\frac{t+1}{t}\right]+1\right)\end{aligned} \tag{2}$$

(Ω is the number of features, s and t are the types of features)

The total number of all features in the sub-window where Ω^{n}, is the sum of the number of various features. I.e:

$$\Omega^{m} = \Omega_{(1,2)}^{m} + \Omega_{(2,1)}^{m} + \Omega_{(1,3)}^{m} + \Omega_{(3,1)}^{m} + \cdots + \Omega_{(1,n)}^{m} + \Omega_{(n,1)}^{m} \tag{3}$$

Since features 1 and 2, features 3 and 4, etc. have rotational symmetry, this can be simplified to:

$$\Omega^{m} = 2\cdot\Omega_{(1,2)}^{m} + 2\cdot\Omega_{(1,3)}^{m} + \cdots 2\cdot\Omega_{(1,n)}^{m} (\text{n is infinite}) \tag{4}$$

The pixel values of a region can be calculated using the integral of the endpoints of the region, and rectangular features can be calculated as rectangular eigenvalues. To wit:

$$ii(x,y) = \sum_{x'\le x, y'\le y} i(x',y') \tag{5}$$

The integral map of coordinate $A(x,y)$ is the sum of all pixels in its upper left corner, where $ii(x,y)$ denotes the integral map and $i(x,y)$ denotes the original image, which is the color value of this point in the case of a color image, and its gray value in the range 0-255 in the case of a grayscale image. The characteristic integral is shown in Fig. 3, where $A(x,y)$ denotes the integral map of point (x,y); and $s(x,y)$ denotes the sum of all the original images in the y direction of point (x,y). The integral map can also be derived using Eq. (6) and Eq. (7):

$$s(x,y) = s(x,y-1) + i(x,y) \tag{6}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{7}$$

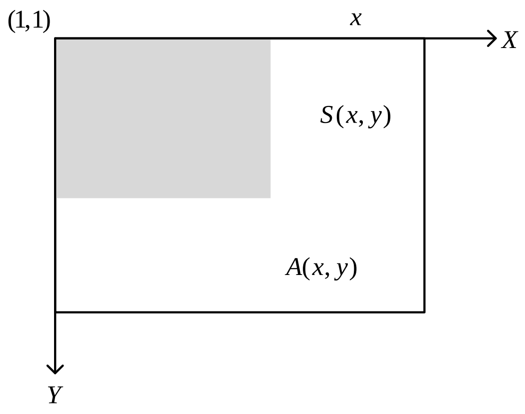


Figure 3. Feature integral

3.1.2 Constructing the training sample dataset

The Haar features and their algorithmic principles have been described in detail in the above sections. In this section, we will target the construction of a database of forward vehicles suitable for China's road traffic situation, which mainly contains positive and negative samples. A suitable number of sample datasets is not only conducive to improving the recognition rate of the classifier, but also will effectively shorten the training time of the classifier. A high-quality training sample dataset requires that the included samples satisfy typical usage scenarios, have low data redundancy, and contain sample images with appropriate resolution and consistency. In the process of constructing the sample dataset, a total of 300 positive samples were collected in this paper. Among them, some of the positive samples are typical vehicle datasets that are publicly available on the web and are screened to select images that meet the characteristics of having a vehicle in front of them as an alternative to the sample dataset.

3.1.3 Sample Image Preprocessing

To improve the extraction of Haar features from the sample dataset constructed above, it is also necessary to perform image preprocessing on the sample dataset, which mainly involves size normalization and grayscaling processing.

1) Size normalization

   The size normalization processing is to exclude the interference of pixel factors on Haar feature extraction and computation, because the positive and negative samples with different pixel sizes have different feature values and number of features when they are subjected to Haar feature extraction and computation. The recognition effect of the resulting classifier will be seriously impacted if samples with different pixels are used during classifier training. Therefore, it is necessary to normalize the size of all positive and negative samples in the sample dataset and batch process them into images of 24×24 pixels for classifier training.

2) Grayscale processing

   It is to convert color images into grayscale images. The intensity values for color images are between 0 and 255, and the same holds true for grayscale images. Black is 0, white is 255, and the color gradually fades from black, fading with different gray values, and finally transitioning to white.

The use of grayscale images instead of color images in the image preprocessing process is to reduce the amount of information to be processed and to speed up the detection of subsequent vehicle features. Color images have a lot of information, which makes the processing process complex and increases the amount of computation. In contrast, grayscale images have 2 bits less image depth compared to color images, so the amount of computation required is relatively less. In addition, from the point of view of local color, length, width, and other distribution characteristics, the two are identical, so the image is grayscale.

In accordance with the importance and other indicators, a weighted average algorithm is adopted for the three RGB components. The weights of each component are distinct, and the OpenCV library's grayscale weights can be utilized. In addition, a weight value can be proposed based on human biology (the human eye is most sensitive to green and least sensitive to blue). Namely:

$$Gray = 0.07216B + 0.71516G + 0.21267R \tag{8}$$

$$Gray = 0.11B + 0.59G + 0.3R \tag{9}$$

3.1.4 Constructing weak classifiers

In the previous section, the construction of the stop-negative sample dataset, the sample image processing and the extraction of Haar features are completed before and after, and in this section, the construction of a weak classifier will be carried out based on the above Haar features and feature values. Classifiers that have a correct rate of object classification above 50% are called weak classifiers. Although its correct rate is not as high as those of the strong classifiers and cascade classifiers discussed in the next section, the weak classifiers are characterized by simple mathematical structure, small computation, and high real-time performance.

Each Haar feature has a corresponding weak classifier; it may be assumed that $f_n(x)$ is the eigenvalue of the n nd Haar feature, and its weak classifier can be expressed by Eq. (10) as:

$$h_n(x) = \begin{cases} 1, p_n f_n(x) < p_n\theta_n \\ 0, otherwise \end{cases} \tag{10}$$

Where the classification result of the weak classifier is $h_n(x)$, and its value 1 means for the presence of vehicle target objects and 0 means the absence of vehicle target objects. p_n is the direction parameter of the inequality, which takes the value range of $\{1, -1\}$, with 1 indicating that the inequality has the sign of less than sign, and -1 the opposite. θ_n is the feature value of the weak classifier $h_n(x)$, i.e., the classification threshold, and this threshold is the key to construct the Haar feature weak classifier. An appropriate threshold reduces the sample classification error rate of the weak classifier for Haar features over the entire sample set. The most appropriate threshold value should result in minimizing its classification error rate.

3.2 Cascade Classifier Based on Adaboost Algorithm

3.2.1 Adaboost algorithm

Adaboost's goal is to enhance the accuracy of the classifier judgment through multiple iterations. The weak classifier (which is the vehicle weak classifier built on Haar features in the previous section) is trained with the same weights as all the training samples at the start. In the Nth iteration, the weights of the samples are determined by the results of the N-1 iteration. In other words, the results of the last iteration determine the weights of the current samples: at the end of each iteration, the weights of the samples are adjusted once, and the weights of the incorrectly categorized samples will be raised. After each iteration, the weights of the samples will be adjusted, the weights of the misclassified samples will be raised, and the misclassified samples will become visible, leading to a new sampling distribution. So on and so forth, under the new sample distribution, the weak classifier is trained again to obtain a new weak classifier. A strong classifier can be formed by combining N weak classifiers with specific weights after N iterations.

3.2.2 Candidate boxes

The specific settings of region candidate frames are shown in Table 1. The region candidate frames of each feature map layer of the Adaboost algorithm have different aspect ratios, $a_r \in \{1,2,3,1/2,1/3\}$, in order to enhance the robustness of the detection of different shapes of the target in the Adaboost algorithm $conv4\_3$, $fc7$, $conv8\_2$ are treated as the prediction of the feature maps, and each feature map generates a certain number of region candidate frames with different aspect ratios. If the feature map size is m × m and the number of region candidate frames with aspect ratio is k, then this prediction layer produces m × m × k region candidate frames.

Table 1. Regional candidate box specific Settings

Feature map	$conv4\_3$	$fc7$	$conv8\_2$
m×m	45×45	28×28	15×15
K	4	6	6
Number	5638	2033	575
1:1	35×35	55×55	95×95

3.2.3 Adaboost Cascade Classifier

The traditional boosting algorithm is to randomly select weak classifier samples to improve their value, but this method cannot accurately classify the weak classifiers. Adaboost cascade classifier is shown in Fig. 4. The Adaboost algorithm utilizes the error data of previous weak classifiers to train the next weak classifiers, which reduces the weights of the accurate classifiers in subsequent training. Training; at the same time, it also improves the weight of the unclassified weak classifiers to realize adaptive classification; finally, it modifies the weight of each classifier through this decisive generation to integrate into a strong classifier. The algorithm's specific steps are as follows.

1) Given n training sample $(x_1,y_1),(x_2,y_2)\cdots(x_n,y_n)$, where $y_i=1$ denotes a positive sample and $y_i=-1$ denotes a negative sample.

2) Initialize the weights of the positive and negative samples, m, l being the number of positive and negative samples, respectively:

$$\omega_{I,i} = \begin{cases} \frac{1}{2m}, \text{Positive sample} \\ \frac{1}{2l}, \text{Negative sample} \end{cases} \tag{11}$$

3) The weak classifier built for each extended Haar-like matrix feature j is:

$$g_j(x, f, p, \theta) = \begin{matrix} 1, & pf(x) < p\theta \\ 0, & \text{other} \end{matrix} \tag{12}$$

Where x is the target feature window, f is the feature value, p is used to adjust the direction of the inequality sign, and θ is the threshold of the feature value. Take all the weak classifiers for T iterations, for $t = 1, 2 \cdots T$.

1) Normalize the weights:

$$\omega_{t,i} = \frac{\omega_{t,i}}{\sum_{j=1}^{n} \omega_{t,j}} \tag{13}$$

2) For each feature weak classifier j, find its weighted error:

$$\varepsilon_j = \sum_{i=1}^{n} \omega_{i,j} \,|\, g_j - y_i \,| \tag{14}$$

3) Select the classifier g_i with the smallest weighting error ε_i and update its weights:

$$\omega_{t+1,i} = \omega_{t,j} \beta_t^{1-e} \tag{15}$$

Where, $\beta_i = \frac{\varepsilon_i}{1 - \varepsilon_i}$, if the feature window can be classified correctly, $e = 0$, otherwise $e = 1$

4) After T rounds of iterative training, the final strong classifier is obtained:

$$G(x) = \begin{cases} 1, \sum_{t=1}^{T} \alpha_t g_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, \text{other} \end{cases} \tag{16}$$

$\alpha_t = \log \frac{1}{\beta_t}$ of them.

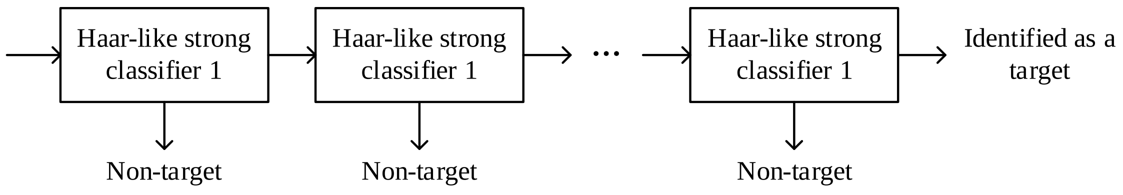

Haar-like strong classifier 1
Haar-like strong classifier 1
…
Haar-like strong classifier 1
Identified as a target
Non-target
Non-target
Non-target

Figure 4. Classification principle of cascade classifier

3.3 Classifier Loading and Testing

3.3.1 Linux-based firmware secondary development

The underlying algorithms of the OpenMV module are written in C/C++ language in modules, and the commonly used image processing algorithms and embedded system underlying related software frameworks have been encapsulated during the development of the software architecture, and one by one submodule are invoked in the application layer using the interpreted language MicroPython.

The so-called secondary development of firmware is to rewrite or rewrite the required image processing functions or recognition algorithm modules in the Linux/Ubuntu environment to generate new OpenMV firmware and then use the embedded ARM chip’s DFU tool for upgrading and re-burning it into the embedded development board of OpenMV for machine vision.

3.3.2 Generating Classifier Files Based on OpenCV

In the above section, the training principle of the cascade classifier has been described and analyzed in detail, and the classifier file needed for the forward vehicle online detection algorithm is made by calling opencv_traincascade in OpenCV. The exe file was generated by performing cascade classifier training. The process of creating the classifier file can be broken down into three basic steps:

Step 1: Import the training sample dataset

The training sample dataset’s construction has been explained in detail in 3.1.2 above. The classifier will be trained by importing the training sample dataset and giving it to opencv_traincascade.exe.

Step 2: Generate the sample description file

The sample description file is available through openCV_createsamples. The exe is carried out to generate the description file of the positive samples, i.e., the file with the extension .vec, which holds the description file of the positive samples for data storage in binary form. Negative samples require the generation of a description file with the extension.dat, which is also necessary.

Step 3: Perform classifier training and .xml file generation

The opencv_traincascade will be called throughout the training process. Exe executable to train the Adaboost cascade classifier based on Haar features. Due to opencv_traincascade. The exe file already encapsulates the whole process of Haar feature extraction and Adaboost classifier training. Only the positive sample description file (pos. vec) and the negative sample description file (neg. dat) from the second step are taken as inputs, and the desired classifier configuration file (cardetection.xml) can be obtained by training. This training process is time-consuming on PCs, and it is recommended to use a remote server to invoke a high-performance multi-core processor for classifier training, which can effectively shorten the training time and improve the training efficiency.

3.3.3 Visual Studio-based platform loading and testing

After completing the training of the classifier, the configuration file (car_detection.xml) of the classifier is obtained, in order to better test the performance of the classifier.

In this paper, we use the Visual Studio platform and the OpenCV algorithm function library to code-edit the software test cases and load pictures and videos that contain vehicle targets to be tested. By calling the classifier file and the image processing algorithm of OpenCV, the work of detecting and recognizing the vehicle in front is completed, and the vehicle target is marked with a candidate box in the picture and video.

4 Example analysis of vehicle inspection

4.1 Algorithm Validation Analysis

4.1.1 Evaluation indicators

Both object classification and localization in the target detection task need to be evaluated, and the measure of target detection accuracy is mAP, which consists of two components: accuracy and recall. The accuracy and recall of each target detected by the network are corresponding. Since each input image in target detection corresponds to multiple targets with multiple labeled inputs, and the network has to predict the category score and location coordinates of each target, its mAP computation step is more cumbersome than that of a classification network.

This subsection detects cars and other targets, and is a binary classification problem where the ultimate goal is to correctly detect all cars without treating other targets as cars. There are four cases of detection result: TP means that the target is correctly detected, FP means that the other objects are treated as cars, FN means that the cars are detected as other targets, and TN means that the other targets are not detected as cars, and the interrelationship of these four cases is shown in Fig. 5. Accuracy is measured by the ratio of positive targets to all targets detected. Recall refers to the ratio of detected positive targets to all positive targets in the whole dataset, i.e., whether the network can find as many cars as possible in the picture. The level of accuracy represents the network’s ability to recognize the target category, while the recall represents the size of the network’s ability to recognize the target. The accuracy and recall are calculated in the following equation:

$$precision = \frac{TP}{TP + FP} \tag{17}$$

$$recall = \frac{TP}{TP + FN} \tag{18}$$

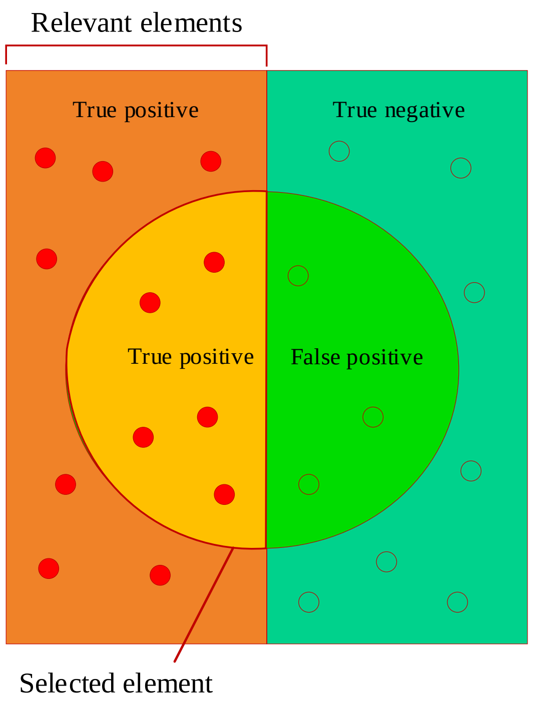


Figure 5. Four kinds of relationships

For a particular class of tests, the category scores output by the network are sorted from high to low. The corresponding accuracy and recall are calculated from top-1 to top-n. As the test sample increases, the accuracy decreases, and the recall increases. Recall is x-axis and accuracy is y-axis, and the accuracy-recall curve is plotted as shown in Fig. 6. A well-performing detector can increase recall while ensuring higher accuracy, and a poorly performing detector may need to lose a significant amount of accuracy in exchange for the trade-off with recall.

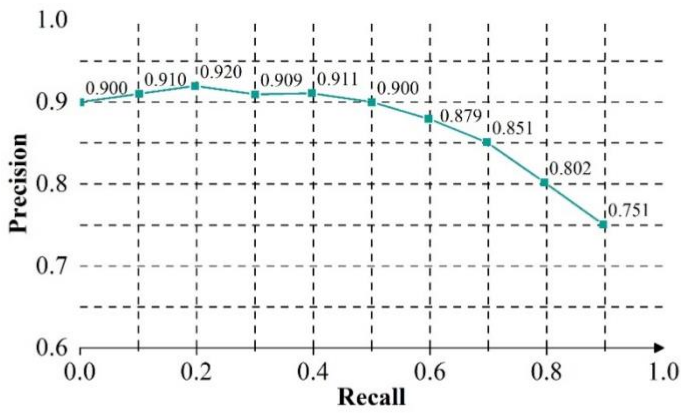


Figure 6. Precision - recall curve

Let there be N sample of a class with M positive examples, then M recalls can be obtained: $\{1/M, 2/M, \cdots, M/M\}$. For each recall r, the maximum accuracy is calculated as follows:

$$P(r) = \max_{r^{'} \geq r}\left(p\left(r^{'}\right)\right) \tag{19}$$

The average precision (AP) for this category is calculated as follows:

$$AP = \frac{1}{M} \sum_{r \in \{1/M, 2/M, \cdots, M/M\}} P(r) \tag{20}$$

Eq. AP measures how good the algorithm is on each category, and mAP measures how good the algorithm is at detecting performance on all categories, calculated as follows:

$$mAP = \frac{\sum_{i=0}^{n} AP_i}{n} \tag{21}$$

In the above equation, n denotes the number of categories and AP_i denotes the average precision of the i rd category.

4.1.2 Data analysis

For the specific data of vehicles, it is known from the previous analysis that it is necessary to set the candidate box matching the data according to its distribution characteristics, in order to improve the network detection accuracy and reduce the network complexity of the purpose. In order to set the appropriate region candidate box for the vehicle dataset, this paper runs K-Means clustering on the training data based on OpenCV's vehicle detection algorithm, which mainly uses the Adaboost algorithm as Haar's feature extraction classifier, the region candidate box and the aspect ratio are set around the center of the clustering, so that the region candidate width and the real box is more compatible with the revised width and height distribution of the region candidate box. The training data in Figure 7 is represented by the cyan points, and the colored straight lines indicate the different aspect ratios of the region candidate frames. The original frame has been modified as follows:

1) Delete the frames with an aspect ratio of 1/3.

2) Keep only the prediction layers conv4_3, fc7, and conv8_2, and delete all the convolutional layers behind them.

3) Set 4 region candidate boxes for conv4_3, and 5 region candidate boxes for fc7 and conv8_2 respectively, and note the changes as OpenCV_change.

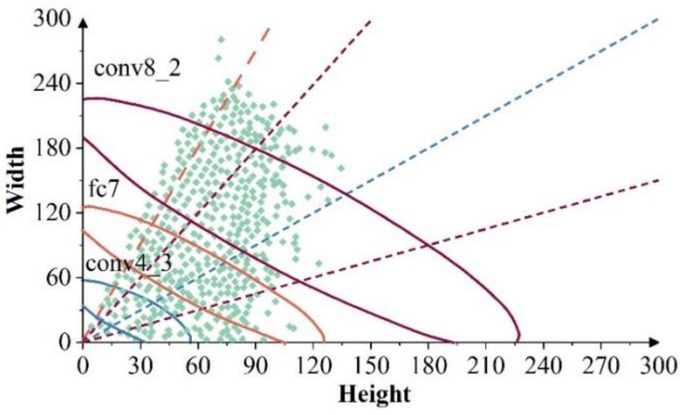
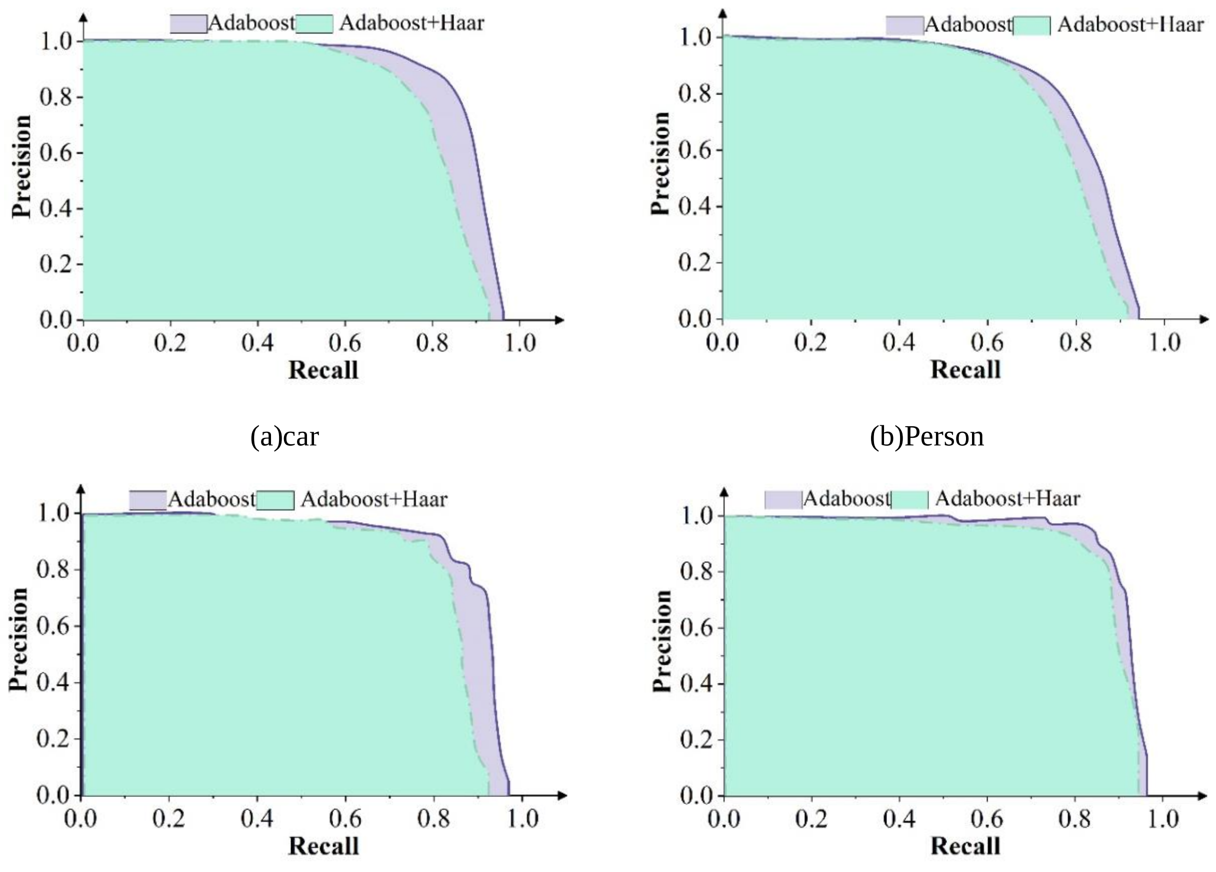


Figure 7. Modified area candidate frame wide high score cloth

The mAP comparisons of the detection effects of the Adaboost algorithm and the OpenCV (Haar+Adaboost) algorithm on the dataset are shown in Table 2 (the table contains only some kinds of AP data). Figure 8 shows the Precision-Recall curves of several kinds of objects in target detection in figure (a)~(d) are Car, Person, Bus, and Cat, respectively. A purple line indicates the detection results of the Adaboost algorithm. Combining Table 1 and Fig. 8, it can be seen that the OpenCV (Haar+Adaboost) algorithm has a map that is 4.51% higher than that of the Adaboost algorithm, and the convergence speed is also faster than that of the Adaboost algorithm.

Table 2. mAP contrast results

Algorithm	Map	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dog
Adaboost+Haar	79.39	86.65	77.63	70.25	49.12	88.84	88.64	88.31	91.65	60.43	83.35	88.44
Adaboost	74.88	74.25	82.33	70.22	61.08	42.44	83.15	81.35	88.45	86.25	72.05	82.15

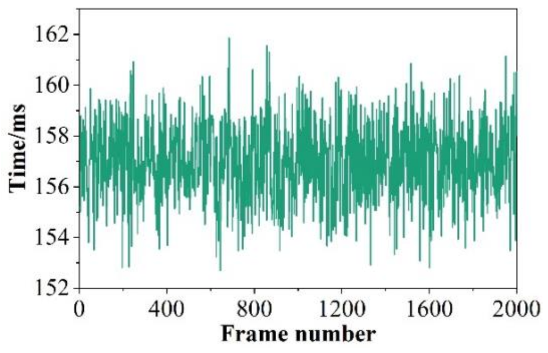


(a)car (b)Person

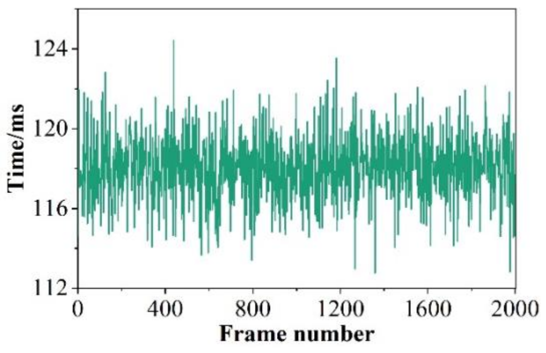
(c)Bus (d)Cat

Figure 8. The precision of several objects in the target detection curve

After completing the testing and comparison of the detection accuracy of the two models on the computer using the test data set, it is also necessary to test the running speed of the two models on the embedded ARM development board. Image data is read directly into the OpenCV run detection on the ARM development board through the USB camera. In the detection thread, first get the system time before the image input network, when the image input network detection is complete, and then get the current system time, the difference between the two for the model to detect 2000 frames of images consumed by milliseconds. In the actual test, the OpenCV (Haar+Adaboost) algorithm detects one frame of image in an average time of 111ms, and the overall operation can be obtained at 9.01fps. The comparison of the detection speed of Adaboost and OpenCV (Haar+Adaboost) algorithms running on ARM development boards is shown in Fig. 9, in which (a)~(b) are respectively Adaboost, OpenCV (Haar+Adaboost). Considering the ARM development board is not the best mobile processor on the market today, and it is the test result in a GPU-less environment, it is necessary to burn the DFU program to improve the detection speed of the OpenCV (Haar+Adaboost) algorithm.



(a)Adaboost



(b)Haar+Adaboost

Figure 9. Different algorithms run on the arm development board

4.2 Algorithm Application Analysis

4.2.1 Different working conditions

In order to test the real-time and accuracy of the vehicle detection algorithm proposed in this paper, this paper carries out a long-distance road test according to four working conditions: strong light, normal light, weak light and nighttime, and counts the leakage rate and false detection rate of vehicle detection under each working condition, and the time of execution of the statistical algorithm. The accuracy of the algorithm is determined by analyzing the leakage detection rate and false detection rate, and the real-time performance of the algorithm is determined by the average value of the execution time of the statistical algorithm.

Under daytime conditions, the test vehicle is driven on a multi-lane highway at a variable speed of 80 to 120 km/h, and the recognition test is conducted on vehicles traveling on the highway. For each cross-combination condition, the recognition system is activated and the test data is recorded.

Under nighttime conditions, the test vehicle is driven on a multi-lane highway at a variable speed of 80~120 km/h, and the recognition test is conducted on vehicles traveling on a single lane. The recognition system is activated and the detection result data is recorded in different lighting situations.

In order to analyze the environmental adaptability of the vehicle detection algorithm proposed in this paper, the vehicle detection results in continuous towel chastity are counted for different working conditions. Statistics of vehicle detection in strong, normal, weak, and night scenes for 2000 consecutive frames of images respectively. It should be noted that because the nighttime lamp can only illuminate this lane, so only the detection of vehicles in this lane at night daytime can be detected at the same time adjacent to the left and right lanes of the vehicle.

4.2.2 False Detection Rate and Missed Detection Rate

To verify the accuracy and reliability of the proposed algorithm, the leakage detection rate and the false detection rate are defined: the leakage detection rate is the probability that a vehicle leakage detection occurs for the current frame during the whole algorithm execution; the false detection rate is the probability that a vehicle false detection occurs for the current frame during the whole algorithm execution. Equation (22) displays the formula for the missed detection rate and equation (23) displays the formula for the false detection rate:

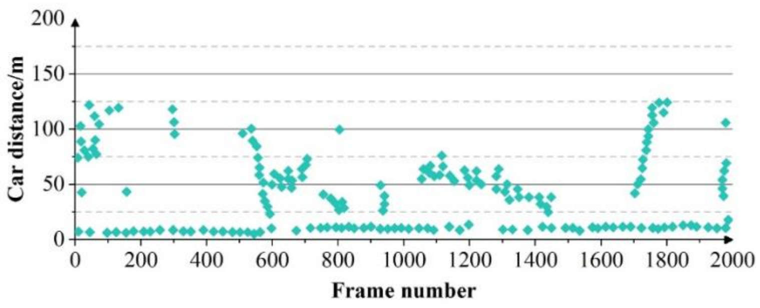
$$\left(1-\frac{N_{get}}{N_{real}}\right)\times 100\% \tag{22}$$

$$\left(1-\frac{N_{getreal}}{N_{real}}\right)\times 100\% \tag{23}$$

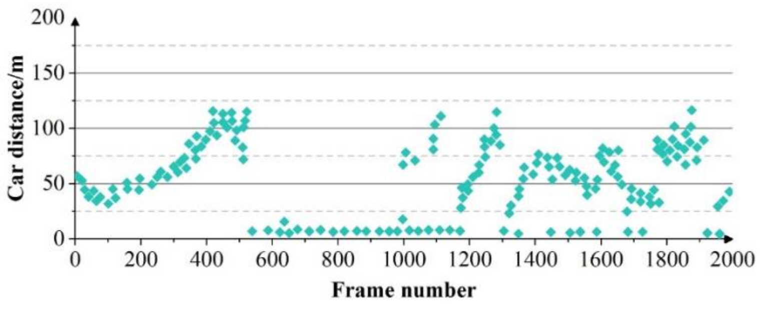
Wherein, the number of true targets N_{real}, i.e., the number of targets in the image sample that should be recognized as vehicles, the number of algorithm-recognized targets N_{get}, i.e., the number of targets in the image sample that the vehicle detection algorithm determines to be considered as style vehicles, and the number of algorithm-recognized true targets $N_{getreal}$, i.e., the number of correct targets in the image sample that the vehicle detection recognition algorithm determines to be style targets.

4.2.3 Intense light analysis

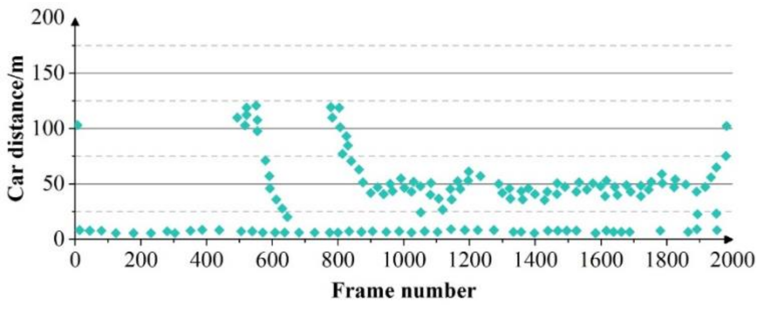
Under the condition of strong illumination, Figure 10 gives the ranging data of the target vehicles in the three lanes in the consecutive 2000 frames of images, Figure (a) ~ (c) for the left lane, the current lane, the right lane, where the horizontal coordinate is the consecutively processed image tilt counts, and the vertical coordinate is the longitudinal distance of the front vehicle from the current vehicle calculated by the recognition software, and if the value is 0, then it indicates that the algorithm considers that there is no vehicle target in the current lane. According to the vehicle ranging result chart under strong illumination, the number of algorithm recognitions is counted as 1416, and by observing the test video, the number of real targets is counted as 1466, which leads to the number of missed detection by the algorithm as 50 times. In addition, by observing the video, it is found that there are 8 times that the ranging error is too large due to the influence of the bridge shadow, and the algorithm recognizes the number of real targets as 1408. The time range of the video has a leakage rate of 3.41% and a false detection rate of 0.56%, which can be calculated from this.



(a)Left lane



(b)Current lane

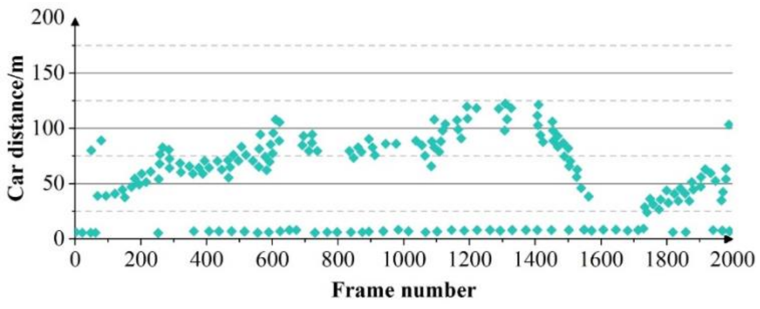


(c)Right lane

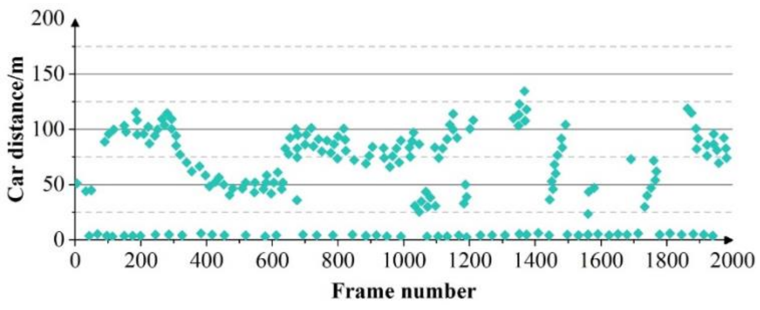
Figure 10. Target vehicle ranging data

4.2.4 Normal light analysis

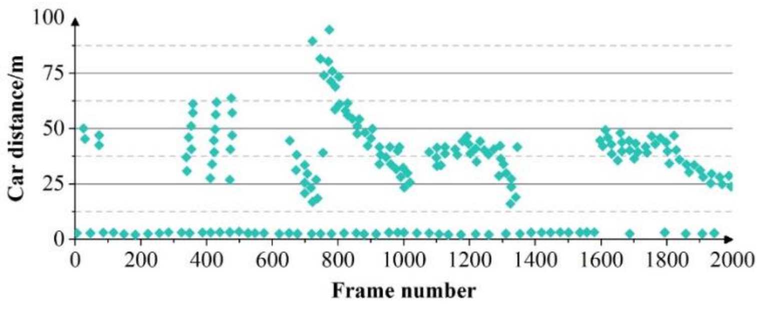
Under clear light conditions, Figure 11 gives the ranging data of the target vehicles in three lanes in 2000 consecutive frames of images, where the horizontal coordinate is the number of consecutively processed image frames counted, and the vertical coordinate is the longitudinal distance of the front vehicle from the current vehicle calculated by the recognition software, and if the value is 0, it indicates that the algorithm considers that there is no vehicle target in the current lane. According to the vehicle ranging results under normal lighting map statistics of the algorithm to identify the number of 1557, by observing the test video, statistics of the number of real targets for 1622, which leads to the number of algorithms missed 65 times; through the observation of the test video can be seen when the road surface of other objects with too large a shadow (such as the shadow of the tree or the shadow of the bridge), it is easy to produce a misdiagnosis and found that there are five misdiagnosis, that is, the bridge shadow containing a The shadow of the bridge containing a vehicle is misdetected as a vehicle, i.e., the algorithm recognizes the true number of targets 1552. The leakage rate and misdetection rate in the video time range are 4.01% and 0.32%, respectively, according to this information.



(a)Left lane



(b)Current lane

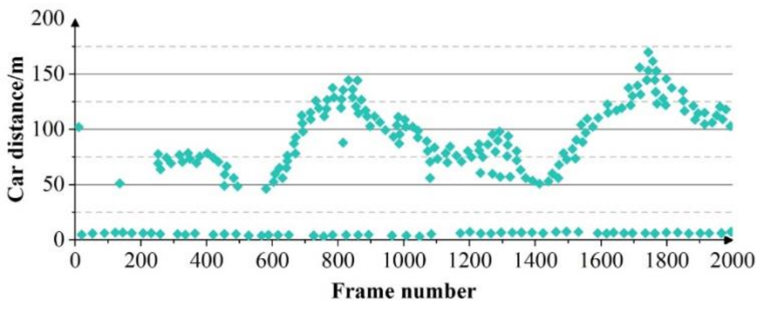


(c)Right lane

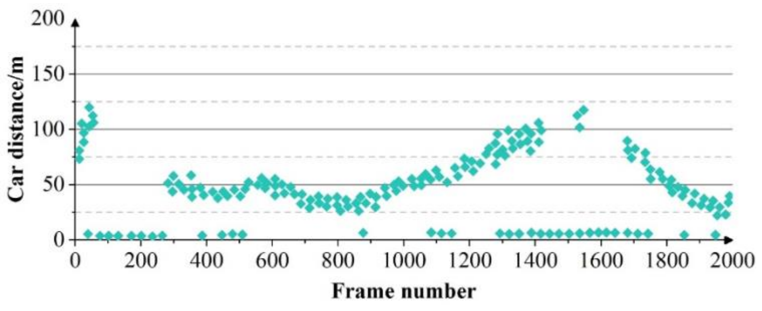
Figure 11. Target vehicle ranging data under normal light

4.2.5 Low light analysis

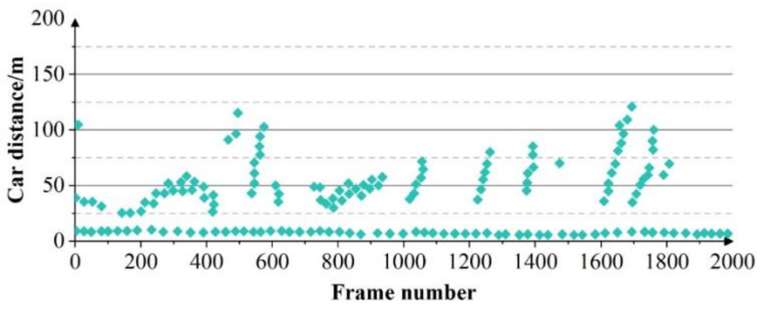
Under the conditions of weak illumination, Figure 12 gives the ranging data of the target vehicles in three lanes in a continuous 2000-frame image, where the horizontal coordinate is the continuous processing of the image shouting counts, and the vertical coordinate is the longitudinal distance of the front vehicle from the car calculated by the recognition software if the value is 0, it indicates that the algorithm believes that there is no vehicle target in the current lane. According to the vehicle ranging results in low light map statistics of the algorithm recognition number of 1622, by observing the test video, statistics of the real target number of 1698, which leads to the algorithm missing the detection of the number of 76 times; In addition, by observing the test video, it was found that the apparent ranging errors caused by road bumps were 8 times and the algorithm recognized the number of real targets 1614. This information allows us to calculate the leakage and false detection rates of 4.47% and 0.49% for the video time range.



(a)Left lane



(b)Current lane



(c)Right lane

Figure 12. The target vehicle is ranging from weak light

4.2.6 Night analysis

Under nighttime lighting conditions, Figure 13 gives the ranging data of the target vehicle in the current lane in 2000 consecutive frames of images, where the horizontal coordinate is the continuous processing of the image shouting counts, the vertical coordinate is the longitudinal distance of the front vehicle from the vehicle calculated by the recognition software, and if the value is 0, it indicates that the algorithm believes that there is no vehicle target in the current lane. According to the nighttime vehicle ranging results chart statistics of the algorithm recognition number of 948, the actual test has been the existence of the target, the real target number of 991; in the detection process, the resulting algorithm misses the number of 43 times; In addition, by observing the test video, it was found that road markings caused 7 false detections, and the algorithm recognized the number of real targets 941. Based on this information, it can be determined that the miss and false detection rates for the video time range are 4.53% and 0.74%, respectively.

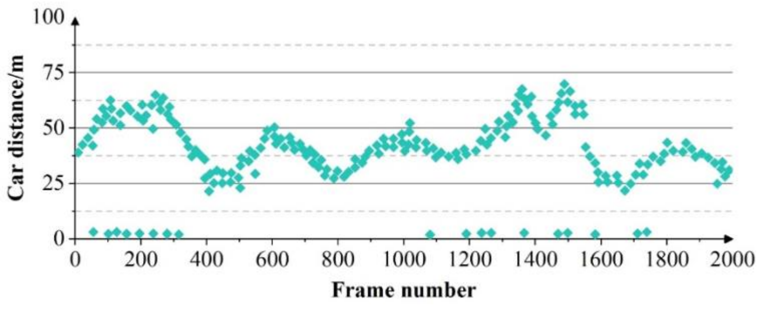
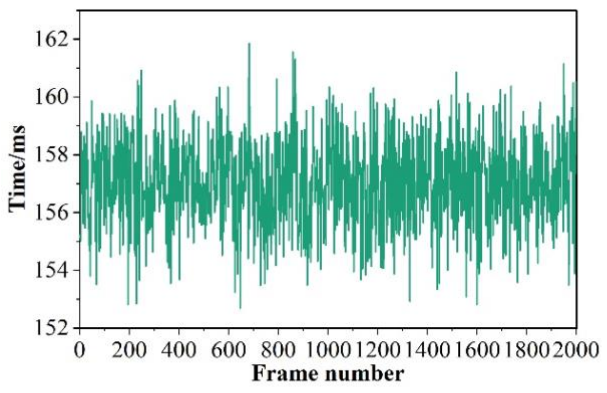


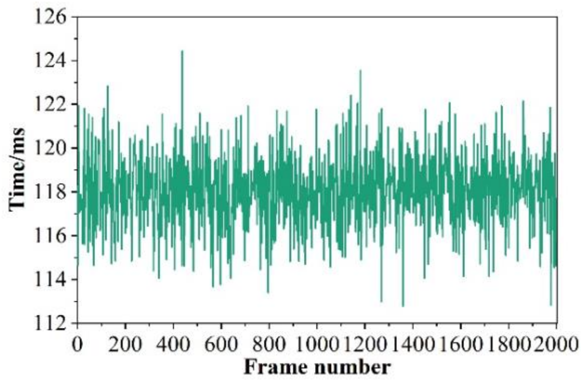
Figure 13. Night target vehicle ranging data

4.2.7 Detection time analysis

In terms of algorithm time, the daytime vehicle detection algorithm consumes more time than the nighttime vehicle detection algorithm because it has to process images within three lanes. The daytime vehicle detection algorithm's time consumption for 2000 consecutive frames is depicted in Fig. 14(a), which has an average processing time of about 157 milliseconds, and the time consumption of the nighttime vehicle detection algorithm for 2000 consecutive frames is given in Fig. 14(b), which has an average processing time of about 118 milliseconds.



(a)Day time



(b)Nigth

Figure 14. Algorithm time analysis

5 Conclusion

Due to poor weather conditions, low illumination, or overexposure to the external environment, the captured vehicle images are often excessively gray and have detail loss, which leads to numerous vehicle violations. This paper proposes a vehicle detection algorithm that utilizes embedded OpenCV to address these problems. The algorithm is validated and analyzed for applications when combined with the dataset.

1) On the dataset, the map of the OpenCV (Haar+Adaboost) algorithm is 4.51% higher than that of the Adaboost algorithm, and at the same time, the convergence speed is more excellent than that of the Adaboost algorithm.

2) The algorithm in this paper has excellent performance in strong light, normal light, weak light, night time and detection time, which confirms the effectiveness of the vehicle detection algorithm based on embedded ARM.

References

[1] Lin, Y., Wang, P., & Ma, M. (2017, May). Intelligent transportation system (ITS): Concept, challenge and opportunity. In 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids) (pp. 167-172). IEEE.

[2] Telang, S., Chel, A., Nemade, A., & Kaushik, G. (2021). Intelligent transport system for a smart city. Security and privacy applications for smart city development, 171-187.

[3] Suryadithia, R., Faisal, M., Putra, A. S., & Aisyah, N. (2021). Technological developments in the intelligent transportation system (ITS). International Journal of Science, Technology & Management, 2(3), 837-843.

[4] Zhu, L., Yu, F. R., Wang, Y., Ning, B., & Tang, T. (2018). Big data analytics in intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems, 20(1), 383-398.

[5] Zhao, J., Hao, S., Dai, C., Zhang, H., Zhao, L., Ji, Z., & Ganchev, I. (2022). Improved vision-based vehicle detection and classification by optimized YOLOv4. IEEE Access, 10, 8590-8603.

[6] Song, H., Liang, H., Li, H., Dai, Z., & Yun, X. (2019). Vision-based vehicle detection and counting system using deep learning in highway scenes. European Transport Research Review, 11(1), 1-16.

[7] Chetouane, A., Mabrouk, S., Jemili, I., & Mosbah, M. (2022). Vision-based vehicle detection for road traffic congestion classification. Concurrency and Computation: Practice and Experience, 34(7), e5983.

[8] Velazquez-Pupo, R., Sierra-Romero, A., Torres-Roman, D., Shkvarko, Y. V., Santiago-Paz, J., Gómez-Gutiérrez, D., ... & Romero-Delgado, M. (2018). Vehicle detection with occlusion handling, tracking, and OC-SVM classification: A high performance vision-based system. Sensors, 18(2), 374.

[9] Nixon, M., & Aguado, A. (2019). Feature extraction and image processing for computer vision. Academic press.

[10] Lentaris, G., Maragos, K., Stratakos, I., Papadopoulos, L., Papanikolaou, O., Soudris, D., ... & Furano, G. (2018). High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. Journal of Aerospace Information Systems, 15(4), 178-192.

[11] Bailey, D. G. (2023). Design for embedded image processing on FPGAs. John Wiley & Sons.

[12] Katare, D., & El-Sharkawy, M. (2019, January). Embedded system enabled vehicle collision detection: an ANN classifier. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0284-0289). IEEE.

[13] Mhalla, A., Chateau, T., Gazzah, S., & Amara, N. E. B. (2018). An embedded computer-vision system for multi-object detection in traffic surveillance. IEEE Transactions on Intelligent Transportation Systems, 20(11), 4006-4018.

[14] Arabi, S., Haghighat, A., & Sharma, A. (2020). A deep-learning-based computer vision solution for construction vehicle detection. Computer-Aided Civil and Infrastructure Engineering, 35(7), 753-767.

[15] Zhang, Y., Guo, Z., Wu, J., Tian, Y., Tang, H., & Guo, X. (2022). Real-time vehicle detection based on improved yolo v5. Sustainability, 14(19), 12274.

[16] Wei, Y., Tian, Q., Guo, J., Huang, W., & Cao, J. (2019). Multi-vehicle detection algorithm through combining Harr and HOG features. Mathematics and Computers in Simulation, 155, 130-145.

[17] Wang, H., Yu, Y., Cai, Y., Chen, X., Chen, L., & Liu, Q. (2019). A comparative study of state-of-the-art deep learning algorithms for vehicle detection. IEEE Intelligent Transportation Systems Magazine, 11(2), 82-95.

[18] Yang, Z., & Pun-Cheng, L. S. (2018). Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. Image and Vision Computing, 69, 143-154.

[19] Sotomayor, D., Rosero, M. F., Benítez, D. S., & León, P. (2017, October). A real-time vehicle identification system implemented on an embedded ARM platform. In 2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON) (pp. 1-7). IEEE.

[20] Wu, H., Hua, Y., Zou, H., & Ke, G. (2022). A lightweight network for vehicle detection based on embedded system. The Journal of Supercomputing, 78(16), 18209-18224.

[21] Su, H., Dong, Z., Yang, F., & Lin, Y. (2021, January). Remote sensing vehicle detection based on embedded system. In Twelfth International Conference on Signal Processing Systems (Vol. 11719, pp. 8-15). SPIE.

[22] Hussain, B., Nawaz, S., & Yousaf, M. H. (2019). Visual vehicle detection scheme on low-powered embedded GPU. Journal of Intelligent & Fuzzy Systems, 36(2), 1867-1877.

[23] Krismadinata,Firstia Bevi Aulia,Ricky Maulana,Muldi Yuhendri,Maaspaliza Azri & Kannabiran Kanimozhi. (2023). Development of graphical user interface for boost converter employing visual studio. IOP Conference Series: Earth and Environmental Science(1).

[24] Wei Jiaxin,Yang Jin & Liu Xinyang. (2024). A text extraction framework of financial report in traditional format with OpenCV. Journal of Intelligent & Fuzzy Systems(4),8089-8108.

[25] Yue Qianqian,Hu Rui & Zhang Xiaoling. (2021). Analysis and Technology Realization of Power Grid Harmonic Detection System Based on ARM Embedded System. Journal of Physics: Conference Series(1).