The Vehicle collision warning system detects the vehicle ahead

Ling Deng¹ and Yan Zhou
School of Economics and Management, Nanjing University of Science and Technology

¹ E-mail: 724618077@qq.com

Abstract. In the field of computer vision and deep learning, vehicle identification is an important subject, which can be applied to the traffic auxiliary system, which can greatly reduce the occurrence of traffic accidents. In this paper, a large number of positive and negative samples are collected, and the algorithm of AdaBoost cascade classifier is used to train the classifier based on Haar feature and LBP feature. Finally, the selected feature and classifier are used for testing. Experimental results show that both of the two eigenvalues have better characteristics and higher detection accuracy when detecting the target vehicle, but LBP features have a faster detection speed, while Haar features have a higher accuracy.

1. Introduction

1.1. Research background and significance of the subject
With the social progress and economic development, the number of people holding vehicles in China has rapidly expanded, and the high frequency of traffic accidents is also worrying. According to the investigation of the traffic department, a large part of traffic accidents in China are caused by longitudinal rear-end collision. Therefore, the optimization of the vehicle anti-collision early warning system to detect the vehicle in front, and as a key technology, to provide a large degree of security for people's safe travel. Moreover, with the rapid development of deep learning and intelligent learning, this class has been widely used in various fields, such as face detection, financial data processing, pedestrian detection, etc. It can also play a major role in vehicle detection.

Sun et al. proposed a Haar rectangle feature optimized by wavelet function, combined with Gabor feature, and then used SVM to form a classifier for vehicle detection [1]. Viola proposed a face recognition based on Haar eigenvalues, which made a major breakthrough in computer vision [2]. Komaropoulos et al. described a vehicle based on K-NN algorithm and stable statistical model [3]. Matthews et al. proposed a method of vehicle identification using a neural network classifier and used PCA to extract eigenvalues of the vehicle [4]. The detection method also has better results. However, there are still many problems that need to be improved in these detection methods. For example, the operation speed of the neural network algorithm is too slow, and the algorithm is too complicated. The detection method based on the SVM classifier has excellent detection success ability, but the parameters are difficult to choose.

With the attention of cascading ideas in recent years, many scholars at home and abroad have proposed vehicle detection methods based on traditional cascade classifiers. The vehicle identification method based on the traditional cascading mode, because its cascading characteristics can quickly screen samples without vehicles, the ability to resist external environmental interference is relatively strong, and the robustness is better. Therefore, this paper proposes a forward vehicle detection method
with better recognition effect. The Haar feature and LBP feature value are used, and the Adaboost cascade classification algorithm is used to train the feature classifier, which shortens the detection time and improves the detection accuracy.

2. Extraction of vehicle features

2.1. Integral image

2.1.1. Concept of integral diagram. Intelligent recognition is to train the existing samples and then detect the unknown images. In the process of training, feature value extraction is very important. At the beginning, feature extraction is mainly carried out through pixel value. This method requires a large amount of calculation. The Haar rectangular eigenvalue is a fast and simple detection feature. Due to its characteristics, its calculation speed is much faster than that of extracting only by pixel value. However, when traversing the image with Haar rectangular features, if each feature needs to calculate the pixel value of all points in the region, it will also take a long time. Hence, the concept of integral graph comes into being.

In the paper published by Paul Viola and Michael Jones in 2001, in order to simplify the calculation of the eigenvalues of rectangular features, they applied the concept of integral graph to face detection for the first time, and integral graph was later regarded as a very important representation method of image features.

2.1.2. Detection principle of integral graph. The eigenvalues of the Haar rectangle feature can only be extracted in the grayscale image, so when using the integral graph to extract the eigenvalues of the rectangular features in an image, we also first adjust the image to a grayscale image. In the gray image, the integral graph value of a certain point is equal to the sum of the gray values of all the points in the upper left area of this point, and the value of the gray value is 0 to 255. We use \( ii(x,y) \) to represent the integral graph value at point \((x,y)\), and \( i(x,y) \) is the gray value at point \((x,y)\), so we have the following expression:

\[
ii(x,y) = \sum_{x-\Delta x \leq x < \Delta x, y-\Delta y \leq y < \Delta y} i(x, y)
\]  

(1)

When we have defined the integral graph value for each point, calculate the pixel value of a region, only need to pass the integral graph values of the four endpoints of the region, as shown in Figure 1:

![Figure 1. Calculation of the integral graph.](image)

![Figure 2. Haar-like features.](image)

When we want to calculate the pixel value of this area of D, we only need to pass the integral graph values of 1, 2, 3, and 4 points, and the value of the result D area can be quickly calculated. By definition, \( ii(1) \) is the sum of the pixel values of all points in the A region, \( ii(2) \) is the sum of the pixel
values of the A+B region, and \( i(3) \) is the sum of the pixel values of the A+C region, \( i(4) \) is the pixel value of the four regions A+B+C+D. Therefore, we require the pixel value of this region of D to be 
\[
D = i(4) + i(1) - i(2) - i(3) = (A+B+C+D) + A - (A+B) - (A+C).
\]
Because the calculation method of the integral graph is used for simplification, when extracting the eigenvalues of the Haar rectangle feature, we only need to traverse the input image once and calculate the integral value of each point. After the rectangular feature calculation, each feature value can be obtained by simply integrating the integral points of several points, simplifying a large number of calculations, and speeding up the calculation speed of the algorithm and shortening the consumption time.

2.2. Haar features

2.2.1. Introduction to Haar eigenvalues. The Haar feature is a target recognition rectangle feature proposed by Paul Viola et al. in 2001. It evolves from a Haar wavelet function to a scale function and then forms a channel and evolves into a Haar eigenvalue. When the cascaded classifier detects the target image, the selected feature type and the calculated feature value have a great influence on the calculation speed of the algorithm. Due to the many excellent characteristics of the Haar feature, it can be detected in multi-scale space, and at each detection scale, the Haar feature will generate a large number of eigenvalues at different scales. After years of development, Rainer Lienhart et al. extended the Haar eigenvalues of the basic types and proposed tilting features [5]. Later, the basic types of Haar-like eigenvalues are edge, linear, center wrap, etc. As shown in Figure 2, each of which can be detected at different scales.

During the detection process, a Haar feature can be scaled up in the detection window and moved at different locations to produce many eigenvalues, but the area ratios of the black and white regions are always the same. Taking the two rectangular features as an example, the area ratio of the two parts is always 1:1 during the scaling up and then the motion detection. The X2 feature is initially two pixels in size. After traversing the entire detection window, the X2 feature with only two pixels is scaled up, and then a full graph traversal is performed until the scaled X2 feature is the same size as the detection window. This way we get all the X2 series features in the detection window.

So how many features can be obtained in a fixed detection window for different Haar eigenvalues? In Rainer Lienhart's paper on improving Haar eigenvalues, it is proposed that when the detection window size is \( W \times H \), a certain rectangular feature size is \( w \times h \), defining \( a = \frac{W}{w} \), \( b = \frac{H}{h} \) at this time, \( a \) and \( b \) are represented as rectangular features in the detection window. The multiplier that magnifies the most. So in the fixed detection window, the number of all features that a rectangular feature can get is:

\[
(H - h + 1) + (H - 2h + 1) + (H - 3h + 1) + \cdots + (H - b \times h + 1) = b \left( H + 1 \right) \frac{h(1+b)}{2}
\]

And because the vertical and horizontal expansions are independent of each other, we can get the value of the number of eigenvalues, which is calculated by the following formula:

\[
\lambda = ab(W + 1 - w \frac{a+1}{2})(H + 1 - h \frac{b+1}{2})
\]

This paper mainly adopts five characteristics of BASIC eigenvalues. The detection window is selected as 24X24. At this time, \( W = H = 24 \), \( w = 2 \), \( b = 1 \), \( X = 8 \), \( Y = 24 \). Next we can calculate 24X24. There are a total of 162,336 features in the detection window.

2.2.2. Characteristics of Haar-like features. (a) High inter-class variability: There is a big difference between each feature. Different feature detection has the same difference in the eigenvalues obtained from the same region, so that different features can be compared when performing vehicle target detection. A good description of the characteristics of the vehicle in different positions in the detection window will have better classification ability when forming the classifier.
(b) Low intraclass variability: The results detected by the same eigenvalue are approximately the same. When detecting the same block area, the detection results of the same eigenvalue are approximately the same.

c) Local intensity difference: The Haar-like eigenvalue is detected based on the local intensity difference, that is, based on the difference between the pixel values of the white area and the black area in the rectangular feature.

d) Multi-scale detection: Multi-scale Haar-like detection windows can be established, such as 2x2, 4x4, 24x24, etc. to perform multi-scale work, and maintain the invariance of scale, the invariance of rotation and the invariance of illumination.

e) High computational efficiency: Since the integral graph is used to simplify the calculation of Haar rectangular eigenvalues, after traversing the complete graph to obtain the integral value of each point, only the integral graph values of several points are needed when calculating the eigenvalues of any block. The obtained eigenvalues can be obtained, so the calculation efficiency of each eigenvalue is the same, and the calculation efficiency is not lowered due to the expansion of the window, regardless of the radius of the detection window.

2.2.3. Haar-like eigenvalue detection principle. There are many Haar rectangle features appearing during training and detection, and the eigenvalue of each Haar rectangle feature is the sum of the pixels of the white rectangle minus the sum of the pixels under the black rectangle. The eigenvalue of the Haar feature is multiplied by the sum of the pixel values in the entire Haar feature region, plus the sum of the pixel values of the black region multiplied by the weight.

\[
\text{feature value} = \text{weight}_{all} \times \sum p + \text{weight}_{black} \times \sum p_{black} \tag{4}
\]

As for the three rectangular features selected in this paper, \(\text{weight}_{black} = 1, \text{weight}_{all} = 3\) because the areas of the black and white rectangles are not equal, the weights of the other three features are \(\text{weight}_{black} = 1, \text{weight}_{all} = -2\), and the areas of the black and white rectangular areas are the same.

2.3. LBP eigenvalue

LBP is a feature that describes the locality of an image by binary. It was first proposed by T. Ojala, M. Pietikäinen, and D. Harwood in 1994. Because the detection mode of the LBP feature training classifier is relatively simple, easy to use, and the detection accuracy is high, the LBP features are widely applied to many different fields such as deep learning and computer vision. The transformation principle of LBP features can be seen from Figure 3. The original LBP window is defined as a pixel point as the center pixel, a 3x3 window is taken around it, and the pixel value of the center pixel is used as a threshold, and then 8 pixels of the center pixel are surrounded. The value is compared with the threshold value, which is greater than the intermediate pixel point as 1, less than the intermediate pixel point is recorded as 0, and the feature value is formed into a binary sequence and then converted into a decimal number. In this way, we can get the value of the intermediate pixel, and then get the eigenvalue data of each point until the whole picture is obtained.

![Figure 3. Calculation principle of LBP eigenvalues.](image-url)
3. Detection principle

3.1. Realization of cascade classifier algorithm

The data set containing N samples is denoted as S, the sample set is \( \{ x_i \}_{i=1,2,\ldots,N} \), and the sample label is \( \{ y_i \}_{i=1,2,\ldots,N} \), \( S = \{ (x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N) \} \). There were k Haar-like features in each sample, and AdaBoost learned a series of weak classifiers from the training data and integrated them into strong classifiers.

(a) Set the initial weight distribution of all samples as \( D_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{iN}) \), \( w_{ii} = 1/N, i = 1, 2, \ldots, N \) \( (5) \)

(b) Perform m-round iteration on the sample data in the weak classifier, \( M=1, 2, 3, \ldots, m \)

1) Train the weak classifier under the current weight \( D_m \) distribution:

\[ G_m(x) : x \rightarrow \{-1, 1\} \] \( (6) \)

2) Calculate the error rate and select the \( G_m(x) \) weak classifier with the lowest error rate as the optimal weak classifier:

\[ e_m = P(G_m(x) \neq y_i) = \sum_{i=1}^{N} w_m I(G_m(x) \neq y_i) \] \( (7) \)

The error rate \( e_m \) of \( G_m(x) \) on the training data set is calculated as the weight of and of the samples misclassified by \( G_m(x) \).

3) Calculate the coefficient \( \alpha_m \) of \( G_m(x) \), which represents the weight \( G_m(x) \) f in the final classifier:

\[ \alpha_m = \frac{1}{2} \ln \frac{1 - e_m}{e_m} \] \( (8) \)

As can be seen from the above formula, \( \alpha_m \) gradually increases with the continuous decrease of \( e_m \), indicating that the performance is better. The weak classifier with higher detection accuracy plays a more important role in the final strong classifier.

4) The weight distribution of training set samples is updated to increase the proportion of correctly classified samples and reduce the weight of wrongly classified samples. In this way, the algorithm can distinguish samples that are difficult to classify. Then, the weight of the new sample data set is obtained and used for the next iteration.

\[ D_{m+1} = (w_{m+1,1}, w_{m+1,2}, w_{m+1,3}, \ldots, w_{m+1,N}) \] \( (9) \)

\[ w_{m+1,i} = \frac{w_{m,i}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), i = 1, 2, 3, \ldots, N \] \( (10) \)

(c) Linear combination of multiple weak classifiers in each stage:

\[ f(x) = \sum_{m=1}^{M} \alpha_m G_m(x) \] \( (11) \)

The final strong classifier is obtained as follows:

\[ G(x) = \text{sign}(f(x)) = \text{sign}\left( \sum_{m=1}^{M} \alpha_m G_m(x) \right) \] \( (12) \)
3.2. Cascade classifier training process

Specifically, the process of training the cascade classifier through the Haar rectangle feature is divided into three steps. In the first step, the same weight is assigned to each sample at the beginning of the training, and then the weak classifier is trained. If the classification of a certain sample is correct, the corresponding weight is increased. If it is classified incorrectly, reduce its weight and then update the next training set to train the next classifier. The entire training process is such an iterative process.

In the second step, the weak classifiers trained in the previous link are combined to form a strong classifier, which is similar to the training process of a single weak classifier. After obtaining each weak classifier, we will increase the weight of the weak classifier with high detection accuracy. Similarly, we reduce the weight of the weak classifier with low detection accuracy.

In the third step, the strong classifier is constructed into a cascade classifier, and the second step training is repeated to form a plurality of strong classifiers, each of which contains multiple weak classifiers. We again connect each strong classifier as a phase. The test will run from the first phase to the last phase. When using cascaded classifiers, we hope to have higher detection accuracy and lower false alarm rate. These ways of cascading basically meet our expectations.

Different from the original Boosting algorithm, each weak classifier of AdaBoost is realized by changing the data classification. After each training, the weights of different types of samples are updated according to different results obtained by the weak classifier, thereby affecting the next training. Using the AdaBoost classifier, you can quickly eliminate some of the less important training data features and focus on the features that make a big difference.

3.3. Cascade classifier detection process

Through the previous principles and the introduction of the detection process, we should understand that OpenCV forms a strong classifier by weak classifier combination, and a strong classifier "series" constitutes a cascade classifier. Then we look at the detection principle of the cascade classifier. When the detection window is fixed, the cascade classifier needs to traverse all the images to find the features of different scales and different positions in the image. Taking the Haar feature as an example, as the detection window moves over the image, the corresponding Haar feature in the detection window also shifts with the window, so that each region in the image can be detected. In order to perform detection on multiple scales, there are generally two common detection methods. We enlarge the detection window by a fixed ratio. In this test, we select 1.1 as the expansion ratio, and when the image is traversed by 1 time window after that, the length and width of the detection window are simultaneously amplified. Another commonly used detection method is to keep the size of the detection window unchanged, scale down the detected image, and then traverse the image with the detection window. As far as the operation speed of the algorithm is concerned, the method of traversing by enlarging the detection window is faster and better than the method of reducing the image to be detected.

4. Algorithm flow

The entire algorithm flow for identifying whether the vehicle exists in the target image is composed of two processes, a training process and a detection process. The training process collects and processes a large number of positive and negative samples, then extracts the corresponding Haar and LBP feature values from the uniform size positive and negative samples, and trains the AdaBoost classifiers corresponding to the two features prepared for the recognition process. The detection process first extracts key Haar features and LBP features from the input image to be detected, and then inputs the features into the trained two classifiers to identify whether there is a vehicle. Next, the training process and the detection process are respectively introduced.
4.1. Training process

4.1.1. Cascading Classifier Training. The process of cascading classifier training is mainly composed of the following steps:

(a) Collection of positive sample and negative sample data: 700 pictures containing vehicles and 2000 pictures excluding vehicles were collected, and the positive samples were divided into training set and test set with 500 and 200 respectively. Partial positive and negative samples are shown in Figures 4 and 5. Samples containing vehicles are the main selection highway monitoring screenshots or shot on the side of the road is clear of vehicle image on the back face front or the rear of the car, and try to include a variety of models, a variety of colors, a variety of sizes, a variety of positions of the vehicle, And the background of the training set picture is as close as possible to the final application, otherwise it is difficult to ensure the validity of the final detection algorithm. But in negative selection of sample should be selected in the final testing environments prone to background, often appear around the vehicle's background, such as pedestrians, road surface, building wall window, the roadside grass, trees, traffic signs, etc, and absolutely cannot contain vehicles, otherwise the trained classifier will be completely unrecognizable.

(b) Sample preprocessing: the positive sample data is processed, and part of the window of the vehicle in the picture is intercepted and normalized into a 24X24 format with the processing software. If not processed, the selected training window has a large size, and the training speed will be very slow and take a very long time. The negative sample does not need to be normalized like the positive sample, but the image size of the negative sample should be larger than the size of the training window 24X24, because the image of the negative sample will be intercepted to the size of the training window during the training process.

(c) Generate vec file and list description file: the description file of positive and negative samples, including address, vehicle number and window. Then the vec file is generated through the description file of the positive sample to normalize the data of the positive sample.

(d) Configure opencv traincascade training tool for classifier training: select numPos and numNeg less than the actual number of positive and negative samples and the number of appropriate strong classifier numStage for training.

Figure 4. Partial sample of the positive sample set.
4.2. Identification process

4.2.1. Experimental results. 200 positive samples were taken as the test set, and the test results were shown in Figure 6 and Figure 7. The detection accuracy of the vehicle detection method based on Haar feature and adaboost cascade classifier is about 92%, and the detection accuracy of the cascade classifier trained by LBP eigenvalues only 86%. The detection speed of the cascaded classifier trained by the Haar rectangle feature is about 28 ms for the average detection speed, and the detection speed of the cascade classifier trained by the LBP is about 46 ms. The detection accuracy and detection speed of the two eigenvalues compared with the original cascade classifier detection, there is a relatively large improvement, and in the case of ensuring the accuracy of the algorithm detection, the influence of most environmental factors is effectively avoided.

Figure 5. Partial legend of the negative sample set.

Figure 6. Example of Haar feature classifier detection results.

Figure 7. Example of LBP feature classifier detection results.

5. Conclusions

(a) According to the analysis from the XML file, Haar eigenvalues mainly rely on floating-point number calculation, and the data storage format is floating-point number, while the data storage format in LBP XML is integer. When the same positive and negative samples are trained, the false recognition rate of LBP eigenvalues is high, while the false recognition rate of Haar eigenvalues is low. In other words, when the same sample base is used, the classifier trained by Haar eigenvalues is more accurate. But this can be remedied by expanding the sample.

(b) The detection speed of the classifier trained by LBP eigenvalue is several times faster than that trained by Haar eigenvalue. The detection accuracy and speed of the two kinds of eigenvalues have
been greatly improved compared with the original cascade classifier detection. In the case of ensuring the accuracy of the algorithm detection, the influence of most environmental factors has been effectively avoided. Compared with the method used in the literature, the vehicle recognition rate of this method is relatively better than that of the classifier based on SVM detection. The training time is shorter than the training time of the traditional cascade classifier algorithm by more than ten hours.

(c) Vehicle detection is an important issue in the research of intelligent transportation systems. How to accurately detect vehicle targets in different environmental disturbances has been a problem that has not been completely solved. In the future, improvements can be made in samples on sunny days and samples from rainy days, vehicle samples at night, mud roads or other samples on the road.

Reference
[1] Sun Z, Bebis G, Miller R 2002 Digital Signal Proc. P1019-1022
[2] Viola P, Jones M 2001 Proc. Int. Conf. on Computer Vision and Pattern Recognition 1 pp 511-18
[3] Komaropulose M, Tsakalides P 2009 Proc. of the 15th Workshop on Statistical Signal Proc. pp 1-4
[4] Matthews N D, An P E, Charnley D, Harris C J 1996 J. Control Engineering Practice 4 pp 473-479
[5] Lienhart R, Kuranov A, Pisarevsky V 2003 Proceedings of the 25th German Pattern Recognition Symposium. pp 297-304