Applying the Particle Filter and Classifier to Front Vehicle Detection and Tracking

Wen-Chang Cheng¹, Chih-Chuan Chen² and Shan-Xiong Wei²

¹Dept. of Computer Science and Information Engineering, Chaoyang University of Technology, 168, Jifong E. Rd., Wufong Dist., Taichung, 41349, Taiwan

²International Integrated Systems, Inc, IISI, 102, 5F Hengyang Rd., Zhongzheng Dist., Taipei, 10043, Taiwan

Abstract: In this paper, we propose a computer vision-based method of front vehicle detection and tracking. The method firstly completes the detection and tracking of the candidate vehicle’s position in the input image by using the particle filter, and then extracts the HOG features in the candidate region, finally inputs these features into the beforehand trained AdaBoost classifier to complete vehicle identification. Since the method only uses the particle filter, no matter whether there is vehicle in the front, a candidate vehicle region will be determined, which results in high false positive rate. Therefore, in the method proposed herein, we will carry out further identification on the candidate regions by using beforehand trained classifier, so as to effectively reduce the false positive rate. The experimental results show that our method realizes an accuracy rate of up to 93.8%, and the execution speed of images under the resolution of 320×240 pixels is up to 15 frames per second.

Keywords: Driver assistance system; Particle filter; AdaBoost classifier; HOG features; Symmetry properties.

1.INTRODUCTION

Keeping a safe distance between the driving vehicle and the front vehicle is the most effective way to reduce crash or collision [1, 2]. Through the front vehicle detection system, the traveling distance to the front vehicle can be controlled at any time to remind the driver to pay attention to drive, so as to effectively reduce accidents. So far, the methods of detecting the front vehicle can be roughly divided into three types based on different sensors: wireless sensor-based method [2], computer vision-based method [3-6] and hybrid-based methods [7, 8]. The wireless sensor-based method mainly uses laser, radar, infrared, thermal sensing or microwave sensors to detect vehicles, its advantages include high accuracy and weatherproof feature (it can be used day or night, shiny or rainy), but its cost is high and cannot recognize objects. The computer vision-based method uses a camera equipped in the vehicle to shoot images of the front vehicle for detection, its advantage is lower cost and it can be used together with the driving logger as well as provides more different applications such as vehicle lane detection, pedestrian detection or object recognition [2]. Its disadvantage is that it is easily affected by environment and light source, thus, different methods might be required for day and night. The last method combines with above two methods. It has the advantages of above two methods, but its cost is high. Without considering the cost, it is the best selection. However, the driving logger is widely used in recent years. Thus, the computer vision-based method is easy to use, which not only reduces the cost but also increases the value of the driving logger. In addition, the smart phones are prevalent in recent years, thus their cameras can be used to perform the analysis on the front vehicle. Based on this, we mainly discuss the computer vision-based method in this paper.

Many vehicle detection methods based on computer vision have been proposed [3-8] in recent years, among which some adopt particle filters to complete the vehicle detection and tracking, while the others are classifier-based methods directly using beforehand trained classifiers to classify the candidate regions of inputted images. The former detects the most possible position of vehicle according to the particle position and features in the input image, obviously, it has one disadvantage, i.e. when there is vehicle in the inputted image, and the particle filter can correctly detect vehicle position. However, when there is no vehicle in the inputted image, the particle filter cannot know whether there is or not, so it still detect the most possible position of vehicle in the inputted image, which results in occurrence of high false alarm; as for the classifier-based method, the candidate regions with the same size will be segmented based on the inputted images for feature extraction, which will then be inputted to the classifier for classification and identification. Therefore, its accuracy is not high, but the false positive rate is high. In addition, in order to detect vehicles with different distances, this method needs to segment the inputted image for several times at different sizes and then performs the same classification and identification, which causes a slow processing speed, thus it cannot be applied in real time.

In this paper, we propose a simple improvement method which combines with advantages of the above two
methods and improves their disadvantages. It firstly uses the particle filter to detect the position of the candidate vehicle in the inputted image, and then extracts features of candidate regions and finally inputs the features into the classifier for vehicle identification. In this way, when the input image contains vehicles, it can correctly detect vehicle positions; however, when the input image does not contain vehicles, it will be further confirmed by the classifier so as to eliminate correctly as to reduce the false positive rate. As shown in the experimental results, the overall accuracy of the proposed method has been significantly improved.

The structure of following part will be as below: section 2 briefly describes the entire flowchart of the method; section 3 introduces the method of vehicle detection and tracking by using the particle filter; section 4 describes the features extraction and classification of candidate regions of the particle filter. In this paper, we use HOG features [9] and adopt AdaBoost classifiers [10, 11] to complete the classification. Section 5 shows the experimental results and the final section gets the conclusion.

![Figure 1 Method flow chart](image)

**2. METHOD FLOW CHART**

Figure 1 illustrates the flow chart of our proposed method. After the image is inputted to the system, the particle filter will determine the bounding box based on the location and size of each particle, then calculate the weight value of each particle by the symmetry of the grayscale, the vertical edge and the horizontal edge in bounding box, and then determine the most possible candidate regions by the weights sum of all particles (part a in Figure 1), and calculate the candidate region’s HOG features (part b in Figure 1) which will be inputted to the beforehand trained AdaBoost vehicle classifiers to complete vehicle identification (part c in Figure 1). Since we focus on the distance between the driving vehicle and the front one, therefore, when the front vehicle has left the region, it will not be detected and tracked any more (part d in Figure 1). At this time, the particle state is reinitialized (part e in Figure 1) so as to get ready to perform the vehicle detection and tracking for an inputted image. Similarly, when the particle filter has determined the most possible position of vehicle which is judged as no-vehicle by the AdaBoost, it will stop detection and tracking. At this time, the particle state is also reinitialized so as to get ready to perform the vehicle detection and tracking for the new inputted image. Otherwise, the vehicle position will be determined by the states and weights of all particles and the distance between the vehicle and the front one is calculated through internal and external parameters of the camera (part f in Figure 1).

**3. DETECTION AND TRACKING OF THE FRONT VEHICLE**

This section explains how to use the particle filter to complete vehicle detection and tracking. Firstly, define each particle’s state that indicates the bounding box of a vehicle. Then, indicates the procedures of predicting, estimating and re-sampling the particle filter.

**3.1 Bounding box particle**

Figure 2 illustrates a example of the bounding box of the vehicle [3]. Therefore, each bounding box particle vector $\mathbf{x}$ is expressed as follows:

$$\mathbf{x}_i = [x, y, w, h, dx, dy, dw, dh], \quad i = 1,...,N,$$

(1)

wherein, $x$, $y$, $w$ and $h$ respectively represents the coordinate positions of the lower left corner at the bottom of the bounding box, height and width. $dx$, $dy$, $dw$ and $dh$ respectively represents the variations of $x$, $y$, $w$ and $h$. The positions corresponding to relevant parameters are shown in Figure 2.

![Figure 2 Schematic diagram of boundary box particles of front vehicle](image)
3.2 Prediction of particles

The purpose of the particle prediction step is to predict next state $x_{i,t}$ based on the current state $x_t$. The prediction process will introduce the predicted noise $w_t$. The particle prediction can be expressed by a simple kinetic motion equation:

$$x_{i,t+1} = Ax_{i,t} + Bw_{i,t}, i = 1, ..., N,$$

wherein,

$$A = \begin{bmatrix} \text{diag}(I_{k \times k}) & \text{diag}(dI_{k \times k}) & 0_{k \times k} & \text{diag}(I_{k \times k}) \end{bmatrix},$$

and

$$B = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_N).$$

Predicted noise $w_t$ can be expressed as a noise of normalized Gaussian distribution. Due to the difference of each parameter scale in the particle vector, the predicted noise $w_t$ introduced for prediction can be adjusted by $B$ matrix parameters so that the predict noise $w_t$ vector is suitable for each parameter scale of particles. Through the calculation in equation (2), the next state of each particle is easily predicted so as to estimate the prediction results.

3.3 Particle estimation

The purpose of the particle estimation is to measure the matching degree of the bounding box of each particle and the vehicle. After observation, it is found that the vehicle has obvious symmetry properties and bottom shadow feature, since the later is susceptible to weather and road surface material. Therefore, we only use the characteristic of the left and right symmetry. To strengthen this characteristic, we not only consider the left and right symmetry of gray scale, but also consider that of horizontal and vertical edges [3]. We define the formula for the left and right symmetry as below:

$$d_k(y, x) = \sum_{j=1}^{y/2} \sum_{i=1}^{x/2} |I_k(y-j,x+i) - I_k(y-j,x-i)|,$$

wherein, $d_k(y, x)$ presents the symmetric value of the bounding box formed by the particle at the position $(y, x), k \in \{1, 2, 3\}$ respectively presents gray scale, horizontal edge and vertical edge image. $I(u, v)$ presents the pixel values of the image. Figures 3 and 4 are two examples. For the convenience of description, the extracted image widths of the above two bounding boxes are wider than those of the actual bounding boxes. Figure 3 and Figure 4 show the bounding boxes with or without vehicle respectively. In the figure, the first row shows grayscale, horizontal edge and vertical edge images, the third row shows symmetry values corresponding to the image of the first row. The figure shows that the smaller symmetrical value is, the higher the symmetry is. If different threshold values are set for feature values, the possible candidate regions can be obtained, as shown in the second row in the figure. It can be found that even multiple symmetrical values are lower threshold value, such as the vertical edge of the image shown in the figure. Even so, the estimation result of combined multiple symmetrical values still can find out the optimal candidate region of corresponding vehicles. In addition, figure 4 shows the bounding box without vehicles. As shown in the figure, the optimal candidate regions of corresponding vehicles can still be found out.

Because the weight value of each particle is in proportion to the measurement results, the weight value of each particle can be expressed as following:

$$p(z_t | x_t) = \frac{\sum_{j=1}^{N} d_k(y, x)^2}{\sigma^2}, i = 1, ..., N,$$

wherein, $d_k(y, x)$ is the symmetrical value of particles at position $(y, x)$. $\sigma$ represents the standard deviation to the measured Gaussian function. In the last step, the weight value of each particle is used to calculate $cdf$ function [12]. The new position of particles can be re-sampled through $cdf$ function correspondence. Therefore, as for particles with large weigh values, there are more new particles corresponded after re-sampling. Conversely, as for particles with small weigh values, there are less new particles corresponded after re-sampling.
4. CLASSIFICATION OF CANDIDATE REGIONS

In this paper, we adopt HOG feature [9] and use AdaBoost classifiers [10, 11] to complete the classification. The following will introduce them respectively.

4.1 HOG features

Histogram of oriented gradients (HOG) feature extraction algorithm is a feature commonly used in object detection, and it is a feature extraction method performing intensity statistics based on oriented gradients of each pixel in the candidate region. It firstly segments the pixels of the candidate region into \( m \times m \) blocks, then calculates the gradient magnitude and orientation of the pixel gray scale value in each block, and then make a statistics of all gradient magnitudes of all pixels in each block based on the orientation to form a histograms, at last combines histograms of all blocks into a vector as the feature of object detection. The calculation of general gradients often adopts Sobel method [13]. Therefore, the horizontal component \( G_x(x, y) \) and the vertical component \( G_y(x, y) \) of the pixel gradients are calculated first, and the gradient magnitude of pixels can be calculated as below:

\[
\|\nabla f(x, y)\| = \sqrt{G_x^2 + G_y^2},
\]

(7)

and the gradient direction \( \theta(x, y) \) is expressed as:

\[
\theta(x, y) = \arctan \left( \frac{G_y(x, y)}{G_x(x, y)} \right)
\]

(8)

The statistics on gradient magnitude of each block is performed based on orientation features. Every 45 degree of the oriented gradient is divided as a unit. There are 8 units in total, namely \( U_k \), \( k = 1, ..., 8 \). When \( \theta(x, y) \) belongs to a unit \( U_k \), the cumulative magnitude value \( \|\nabla f(x, y)\| \) in this direction \( U_k \) can be calculated as bellows:

\[
q_k = \varphi_k + \frac{\|\nabla f(x, y)\|}{\min_{i=1,2,...,8} \|\nabla f(x, y)\|} \theta(x, y) \in U_k, \ k = 1, 2, ..., 8
\]

wherein, \( \varphi_k \) is the cumulative value of magnitudes in 8 directions. Finally, the block’s statistical histogram functions as its feature vector. In this application, block overlaps are available so as to increase the feature expression.

4.2 AdaBoost classifier

Since HOG features contain many dimensions and some features are not helpful to classification, therefore, features must be selected and then classified by classifiers. AdaBoost classifier can select features and has features of classifiers. AdaBoost classifier, firstly proposed by Freund et al [10, 11], is a strong classifier linearly combined by multiple weak classifiers, as shown in Figure 5. Each weak classifier only performs classification on one dimension of the input feature vector. In the whole training process, the algorithm can be adaptive to increase the number of weak classifiers so as to improve the overall accuracy of the classification and focus on key features. Once a weak classifier is added, the algorithm uses the minimum error to calculate the weight value of corresponding weak classifiers. At the same time, the weight value of each training sample is re-adjusted and sent to next newly added weak classifier for use. Based on the method of adding new weak classifiers, the overall performance of classifiers can be gradually improved.

**Figure 5** AdaBoost classifier architecture.

**Step1.** Give example images \((x_1, y_1), ..., (x_n, y_n)\), where \(x_i \in R^n\), \(y_i \in \{-1, 1\}\), for negative and positive example respectively.

**Step2.** Initialization distribution \(\omega_i(1) = 1/2p, 1/2q\), for \(y_i = 1, -1\), respectively where \(p \) and \(q \) are the number of positives and negatives respectively.

**Step3.** For \(t = 1, ..., T\):

- Find classifier \(h_t(x)\rightarrow\{1, -1\}\), \(h_t = \text{argmin} (\xi), j = 1, ..., n\), where \(\xi_j = \sum_{i=1}^{n} \omega_i(1) [y_i \neq h_t(x_i)]\).
- Weight classifier:
  \[
  \beta_t = 0.5 \ln \left( \frac{1 - \xi_t}{\xi_t} \right)
  \]
- Update distribution:
  \[
  \omega_i(t+1) = \frac{\omega_i(t) \exp(-\beta_t y_i h_t(x_i))}{Z_t}, \forall i,
  \]
  \(Z_t\) is for normalization.

**Step4.** Output final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \beta_t h_t(x) \right).
\]

**Figure 6** AdaBoost Algorithm.

Figure 6 shows the algorithm of the AdaBoost classifier. Assume that a training sample set \(\{x_i, y_i\}, i = 1, ..., m\) is given, in which \(x_i \in R^n\), \(y_i \in \{1, -1\}\). Firstly, the origin weight values of all the training samples are initialized. If the training sample set contains \(p\) positive samples and \(q\) negative samples. In other words \(m = p + q\), therefore, the weights of the positive sample and the negative sample are set as \(1/2p\) and \(1/2q\) respectively. Then, the selection cycle of \(T\) weak classifiers is performed, when the \(t\)-th cycle is performed,
weak classifiers \( h(x), j = 1, \ldots, n \) with minimum errors for each dimension will be found out. Then weak classifiers with minimum errors will be selected out from these weak classifiers as the weak classifier \( h(x) \) of the \( t \)-th cycle, and then the weight value \( \beta_i \) of this weak classifier \( h(x) \) is calculated. The weight value \( \beta_i(t+1), i = 1, \ldots, m \) of each training sample is re-adjusted, so that when the \((t+1)\)-th cycle is performed, the samples wrongly classified at the \((t)\)th cycle will be classified preferentially. Finally, the product sums of all \( T \) weak classifiers and their corresponding weight values \( \beta \) are calculated and the \textit{sign} function is taken, then we can get strong classifiers \( H(x) \).

5. EXPERIMENTAL RESULTS
This section will introduce the execution of the experiment and experimental results. Figure 7 shows that the test camera is installed in the vehicle below the rear view mirror on the vehicle’s front windshield to capture images, and manually cut out 200 training templates with vehicles. Figure 8 shows part of vehicle examples, through which we can find that the example size is different because the distances of vehicles captured are different. Since the calculation of HOG features does not require normalizing the example size, the vehicle example will directly maintain originally segmented size. Furthermore, 400 frames of training examples without vehicles are randomly extracted from the driving logger. Figure 9 shows part of training examples without vehicles. Since they are randomly segmented, the sizes of training examples without vehicles are the same. To verify the performance of the AdaBoost vehicle classifier, we divided the above two samples into two groups, one for training and the other for testing, and the interchangeable verification is performed.

![Figure 7 Camera Installation and ROI Range Setting](image.png)
The performance discussion explored the problem of time consuming. However, it is time-consuming. Therefore, we use 20 weak classifiers as the compromise choice to form the final AdaBoost vehicle classifier. In addition, to make the AdaBoost vehicle classifier more effective and practical, we select 200 examples with vehicle and 400 samples without vehicle to perform samples re-training by the AdaBoost classifier.

![Figure 8](image8.png) Part of training samples with vehicles.

![Figure 9](image9.png) Part of training samples without vehicles.

Figure 10 shows the training and testing results of the AdaBoost vehicle classifier [14], the left figure shows the error result of training, and the right one shows the testing results. The horizontal axis presents the number of weak classifiers and the vertical axis presents the error rate. From Figure 10, it can be found that, when the number of weak classifiers reaches 9 among AdaBoost classifiers during the training, the error rate of training samples has reached 0%, and the testing error rate is about 14%. If the number of weak classifiers is continuously increased, the error rate of training examples still remains at 0%. However, the error rate of testing samples decreases and it will maintain at about 10% once the number of weak classifiers of the AdaBoost classifier reaches 30. Although the increase of weak classifiers can reduce the error rate of the testing samples, the calculation time will be increased too. Therefore, we use 20 weak classifiers as the compromise choice to form the final AdaBoost vehicle classifier. In addition, to make the AdaBoost vehicle classifier more effective and practical, we select 200 examples with vehicle and 400 samples without vehicle to perform samples re-training by the AdaBoost classifier.

![Figure 10](image10.png) Classifiers’ training and testing results.

After the training of the AdaBoost vehicle classifier is completed, we extract additional $P = 1000$ images with vehicles in front from the driving logger as well as $N = 1000$ images without vehicles, and test these images by the proposed method. The experimental results are recorded in Table 1. In the experiment, the particle filter with 100 particles, the AdaBoost classifier and our proposed method are used. The performance discussion includes detection rate ($DR$), false positive rate ($FPR$), and accuracy rate ($AR$). $DR$ is defined as $TP/TP$, in which $TP$ is the number of vehicles in the detected image with vehicles. $FPR$ is defined as $FP/N$, in which $FP$ is the number of vehicles wrongly detected from the images without vehicles. $AR$ is defined as $(TP+TN)/(P+N)$, in which $TN$ is the number of non-vehicles in the detected image without vehicles. Table 1 shows that, if the vehicle detection and tracking only uses particle filters, the $DR$ is 100% and the $FPR$ is also 100%, so the $AR$ is only 50%. The method only using the AdaBoost vehicle classifier has 91.1% of $DR$, 11.8% of $FPR$ and $AR$ up to 89.7%, which is about 40% higher than that of the method only using the particle filter. However, it is time-consuming. Finally, our proposed method has 97.3% of $DR$, 9.8% of $FPR$ and $AR$ up to 93.8%. Although some inputted images contain vehicles, they are excluded through the classification of classifiers. Thus the $DR$ cannot reach 100% which can be realized by only using particle filter. However, our proposed method shows significant improvement in overall $AR$. Since the AdaBoost classifier only focus on the classification of candidate regions determined by the particle filter, it solve the problem of time-consuming. In a word, our proposed method can effectively improve the accuracy and improve the false prediction of the particle filter while there is no vehicle in front. It also improves the problem of time consuming caused by the blind detection of the AdaBoost classifier.

<table>
<thead>
<tr>
<th>Method</th>
<th>$DR$ ($TP/P$)</th>
<th>$FPR$ ($FP/N$)</th>
<th>$AR$ $(TP+TF)/(P+N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle filter</td>
<td>100%</td>
<td>100%</td>
<td>50.0%</td>
</tr>
<tr>
<td>AdaBoost classifier</td>
<td>91.1%</td>
<td>11.8%</td>
<td>89.7%</td>
</tr>
<tr>
<td>Particle filter + AdaBoost classifier</td>
<td>97.3%</td>
<td>9.8%</td>
<td>93.8%</td>
</tr>
</tbody>
</table>

6. CONCLUSION

In this paper, we have completed a computer vision-based method for detecting and tracking the front vehicle. The method applies particle filter to complete vehicle detecting and tracking and then uses the AdaBoost vehicle classifier to further finish vehicle identification in candidate regions. The experimental results show that the method only using particle filter for vehicle detecting and tracking has 100% of $DR$, 100% of $FPR$ and 50% of $AR$.
because it cannot know whether there is actual vehicles or not. However, the method only using AdaBoost vehicle classifier has 89.7% of AR, which is about 40% higher than that of the method only using the particle filter, but it is time-consuming. Our proposed method has 97.3% of DR, 9.8% of FPR and AR up to 93.8%, which improves the problems of vehicle detecting and tracking method only using particle filter, and improves the problem of time consuming caused by blind detection of the AdaBoost classifier. The execution speed of our proposed method can be up to 15 frames per second under the resolution of 320×240 pixels.

References


