This paper proposes a novel video-based vehicle detection approach with data-driven adaptive neuro-fuzzy networks. The key ideas include configuring several virtual loops as vehicle detection zones in the image, assuming moving vehicles will cause pixel intensities and local textures to change, and then identifying such changes to detect vehicles. In this work, vehicle detection is treated as a pattern classification problem. First, 14 image features (regarding foreground area, texture change, and environmental condition) are extracted to represent the distinction between vehicle and nonvehicle patterns. Then, three neuro-fuzzy networks are trained via incremental semi-supervised learning to build a data-driven adaptive classifier, which judges whether a vehicle is located in the virtual loop. The semi-supervised learning procedure is performed based on a modified tri-training approach, to automatically optimize the structures and parameters of the component neuro-fuzzy networks. Experimental results illustrate that the proposed approach is accurate and robust to detect vehicles in complex environments (e.g. adverse illumination and weather conditions), and thus can improve the performance of video-based vehicle detection.

**Keywords:** Vehicle detection; virtual loop; neuro-fuzzy network; combined classifier; semi-supervised learning.

1. **Introduction**

In the last decade, traffic congestion and accidents in metropolises have become serious issues of public concern. Developing intelligent transportation systems (ITS) is regarded as an effective means for tackling these issues, and it is widely thought that traffic information collection plays an essential role in modern transportation systems, especially as the field of ITS is evolving to be more data-driven.
than before. Recently, more and more video detection systems (VDS) have been used to collect information in the transportation field.

Generally, the existing video detection approaches fall into two major types: vehicle tracking method and virtual loop method. The vehicle tracking method employs motion trajectories of vehicles in the video sequence to extract various traffic information, whilst the virtual loop method identifies pixel changes in local zones to judge whether vehicles exist or not. Each method has its pros and cons. In principle, the vehicle tracking method is more accurate in collecting traffic information, as vehicle trajectories originate from not only the temporal domain but also the spatial domain. Nevertheless, vehicle tracking remains a difficult topic and is neither robust nor reliable under high degrees of clutters and occlusions. Hence the vehicle tracking method is used mainly to monitor sparse traffic scenes like highways. By contrast, the virtual loop method cannot make full use of spatial information in the video sequence, so that the acquirable traffic information is relatively finite. But this type of method is hardly restricted by complex external environments, and has potential to work in all-weather conditions. In brief, the virtual loop method is more suitable for real-world vehicle detection tasks, especially under adverse illumination and weather conditions.

For practical applications, this paper focuses on improving the virtual loop method and making it work better in complex dynamic environments. Most of the existing commercial VDS, such as Iteris, Peek, and Autoscope, adopt this method. Although these products have gained more and more market share, they require improvements in performance of algorithms. Recent studies indicate that although these products work well in cloudy daytime, they tend to be disturbed by a lot of adverse factors (e.g. moving shadows, rain/snow/fog weather conditions, and vehicle headlights at night). Medina et al. evaluated the influences of illumination conditions (dawn, sunny morning, cloudy noon, dusk, and night) and adverse weather conditions (rain and snow in both daytime and nighttime, and fog in daytime) on performance of three VDS (Iteris, Peek, and Autoscope). Their findings included: (1) illumination conditions significantly influenced the performance of VDS, and the best performance was found in cloudy noon conditions; (2) during sunny morning and night, false calls greatly increased due to vehicle shadows or vehicle headlight reflections, up to 36% under sunny conditions and 50% at night; (3) the performance of VDS was not greatly impacted by daytime light fog or rain conditions without wind, but significant changes were observed under dense fog and snow in daytime, and snow and rain in nighttime.

Research on video-based vehicle detection has been conducted mainly through background subtraction, but it is difficult or even impossible to accurately segment foreground from video images because the traffic environments are often complex. Particularly, moving shadows in daytime and vehicle headlight reflections at night will make background subtraction inaccurate. Hsu et al. detected foreground pixels and removed shadows, and then computed the exponential entropy
in the detection zone to determine whether a vehicle exist. The authors did not consider nighttime vehicle detection at all. Cho et al.\textsuperscript{9} segmented foreground pixels in daytime and vehicle headlights in nighttime, and then used these features to train two neuro-fuzzy networks. This approach distinguishes between daytime and nighttime vehicle detection and causes inflexibility in real-world applications. For example, at twilight when some vehicles turn on the headlights while others do not, it is impossible to detect all vehicles by segmenting either foreground pixels or vehicle headlights. To make matters worse, accurate segmentation of foreground or vehicle headlights is very difficult in practice. Therefore, a holistic approach is desired for long-term running. Yuan et al.\textsuperscript{37} proposed a unified method to detect and count vehicles during the day and night, which was based on a multiple feature background model using morphology and color difference. They claimed this method was robust to illumination and background changes.

Recently, we proposed a general-purpose approach to video-based vehicle detection, from the view of pattern classification.\textsuperscript{29} We captured lots of traffic videos (in different times, at different locations, and under different environmental conditions), extracted 14 image features from the virtual loop zone and the global image, and finally trained several types of pattern classifiers to estimate the decision boundary between vehicle and nonvehicle patterns. After collecting a large number of training samples from the traffic videos, we used the statistical learning technology to improve accuracy and robustness of vehicle detection algorithms. This approach is consistent in detecting vehicles in all times, no matter daytime or nighttime. However, it assumed that the characteristics of the data-generating process do not change over time, and adopted supervised learning techniques to construct the pattern classifiers.

Nevertheless, traffic environments are usually complex and time-varying, due to dynamically changing illumination and weather conditions. The vehicle detection approach should be endowed with the ability of incorporating new information that emerges after the training of the underlying pattern classifier has been completed. Adapting to the changing environment requires updating structures and parameters of the pattern classifier, meanwhile preserving the originally learnt but still valid knowledge. In light of these, we make an extension to our previous work.\textsuperscript{29} Specifically, the semi-supervised learning mechanism\textsuperscript{11,18,24,25,35,40,41} is integrated into the vehicle detection methodology. We construct multiple data-driven adaptive neuro-fuzzy networks (as a combined classifier) via incremental semi-supervised learning. Through an elaborately crafted learning process, the classifier can optimize its structures and parameters automatically in complex dynamic environments. As a result, the vehicle detection approach acquires higher accuracy and stronger robustness.

The remainder of this paper is organized as follows. Section 2 introduces the whole structure of the proposed approach. In Sec. 3, we describe the feature selection and extraction module. The procedure of constructing data-driven adaptive neuro-fuzzy
networks via supervised learning and incremental semi-supervised learning is de-
tailed in Secs. 4 and 5. Experimental results and discussion are presented in Sec. 6. Finally, the conclusion is drawn in Sec. 7.

2. Structure of the Proposed Approach

The proposed vehicle detection approach is composed of two interrelated stages: training and classification. Figure 1 depicts the whole structure of this approach.

At both stages, the background modeling and foreground segmentation module is used to segment foreground from the background. Admittedly, the result of foreground segmentation may be inaccurate due to various environmental disturbances. However, this module plays an essential role in defining a compact representation of the vehicle and nonvehicle patterns. Refer to Sec. 3 for more details.

At the training stage, the feature selection and extraction module generates appropriate features from the video images, and collects a set of positive/negative training samples that correspond to vehicle and nonvehicle patterns, respectively. These samples constitute a training dataset. Then, the pattern classifier is trained via supervised learning to partition the feature space into two parts: vehicle and nonvehicle. More details are presented in Secs. 3.2 and 4.

At the classification stage, the pattern classifier continuously labels the newly measured samples (i.e. new unseen data) into vehicle and nonvehicle. In order to endow our approach with the ability of adapting to complex dynamic environments, the classification results of new samples are used backward to retrain the pattern classifier in an incremental semi-supervised learning manner (as indicated by the dotted arrow in Fig. 1). Or to put it another way, the structures and parameters of the pattern classifier are optimized automatically. More details are presented in Sec. 5.

3. Feature Selection and Extraction

On the video frame, we configure several virtual loops (as vehicle detection zones) along the moving directions of vehicles. The shape of virtual loop is convex...
quadrilateral, and at least one loop is configured on each lane. The width of virtual loop is slightly less than the lane width, and its length is approximately equal to a car’s length, as shown in Fig. 2(a). We first model the background and segment the foreground from the images, and then extract 14 features to represent the vehicle and nonvehicle patterns.

3.1. Background modeling and foreground segmentation

The Gaussian mixture model (GMM) method is adopted to model the background and segment foreground pixels. The history of every pixel in the image is modeled as a mixture of Gaussian distributions based on its RGB components. In this work, we need a visible background image to extract the image features. We fetch the mean vector of the first Gaussian distribution of each pixel to constitute the background image. For example, the result of background modeling and foreground segmentation for a benchmark video “Highway I” is illustrated in Fig. 2. It should be noted that some advanced background modeling methods, such as the kernel density approximation method and the sample-based visual background extractor (ViBe), cannot be used here because they cannot produce a visible background image.

3.2. Feature selection and extraction

Once the virtual loops have been configured, four internal lines \(a_1, a_2, b_1, \) and \(b_2\) are automatically created inside each virtual loop. The line endpoints divide every edge of the virtual loop into three segments of equal length. Formally, we define that a virtual loop is occupied by vehicle, if and only if any part of the vehicle enters the middle subregion surrounded by \(a_1, a_2, b_1, \) and \(b_2\). Based on this definition, Fig. 3(a) shows a positive example that the virtual loop is occupied by vehicle, while Fig. 3(b) shows a negative one. The image features of interest are selected and extracted as follows.

![Image](image_url)
3.2.1. Features regarding foreground area

Firstly, foreground area is an important cue for vehicle detection. When a virtual loop is occupied by vehicle, it must contain a certain number of foreground pixels. If the number of foreground pixels is too small or close to zero, it is impossible that a vehicle is located in the virtual loop. Hence the foreground ratio \(fr\) inside the virtual loop is selected as a key feature. From Fig. 3, we also find that based on only the foreground ratio \(fr\) inside the whole virtual loop, it is difficult to judge whether the virtual loop is occupied by vehicle. For that we calculate the foreground ratios \(fr_{a1}, fr_{a2}, fr_{b1},\) and \(fr_{b2}\) on the four internal lines \(a_1, a_2, b_1,\) and \(b_2.\) Together with \(fr,\) these five features characterize the foreground area.

In traffic environments, however, background subtraction is often disturbed by various adverse factors, with the foreground mask contaminated by noisy nonobject pixels. From Fig. 2 we have seen that moving shadows can be identified as foreground. Figure 4 shows another instance where vehicle headlight reflections at night are identified falsely as foreground. Hence, in addition to features regarding the foreground area, it is necessary to analyze images on a pixel-by-pixel basis.

Fig. 3. There are four lines \(a_1, a_2, b_1,\) and \(b_2\) inside each virtual loop, used to determine whether the virtual loop is occupied by vehicle. (a) A positive example. (b) A negative example.

Fig. 4. At night, reflections of vehicle headlights are identified falsely as foreground. (a) Raw image. (b) Foreground mask.
foreground area, some other features are necessary to eliminate the negative effects of such noises.

3.2.2. Features regarding texture change

In the computer vision domain, it has been proven that compared with various noises, the actual objects often change the background texture more significantly.\textsuperscript{1,27} In light of that, we calculate the standard deviation of the morphological edge magnitudes of differences between the foreground and background in the virtual loop, as a characterization of the texture change. The calculation flow is depicted in Fig. 5.

By subtracting the raw image with the background image, we get a difference image, which is not a real image, but intermediate variables in the range from $-255$ to $255$. Then, for every foreground pixel in the virtual loop, we employ the Ede morphological edge operator\textsuperscript{17} to process the difference image and get the morphological edge magnitude. Finally, we calculate the standard deviation $sd$ of morphological edge magnitudes for all the foreground pixels in the virtual loop, as well as the standard deviations $sd_{a1}$, $sd_{a2}$, $sd_{b1}$, and $sd_{b2}$ of edge magnitudes for the foreground pixels on four lines $a_1$, $a_2$, $b_1$, and $b_2$. The reason for using the Ede operator is that compared with many other edge operators (such as Sobel and Laplace operators), the edge magnitude computed with Ede operator is larger, and thus benefits to enhancing the discrimination ability between the actual objects and various noises.

From the videos “Highway I” (see Fig. 2) and “Night I” (see Fig. 4), we collect hundreds of positive (i.e. the virtual loop is occupied by vehicle) and negative samples (i.e. the virtual loop is not occupied). The histogram of $sd$ is shown in Fig. 6, where green curves correspond to the vehicle pattern and blue curves correspond to the nonvehicle pattern. We can see that $sd$ is effective to characterize the texture change, and the vehicle pattern causes a larger texture change than the nonvehicle pattern. However, the optimal threshold to discriminate vehicle from nonvehicle is different for different videos. As can be seen, the optimal threshold is about 19 for “Highway I”, and about 12 for “Night I”. In fact, this threshold depends strongly on the environmental condition.

![Fig. 5. Calculation flowchart for the standard deviation of morphological edge magnitudes of differences between the foreground and background.](image-url)
Furthermore, we collect more samples from lots of traffic videos under various illumination (daytime and nighttime) and weather (sunny, cloudy, foggy, etc.) conditions. The values of the feature pair \([fr, sd]\) are visualized in Fig. 7. As can be seen: (1) for vehicle pattern (positive samples), both foreground area and texture change are relatively large; (2) for nonvehicle pattern (negative samples), the foreground area or the texture change is relatively small; (3) there exists a great extent of overlap between vehicle pattern and nonvehicle pattern. This overlap is caused
mainly by the complexity of traffic environments. Hence, based on only the features regarding foreground area and texture change, it is difficult to discriminate vehicle from nonvehicle accurately. Some other features are required, especially those regarding environmental condition.

3.2.3. Features regarding environmental condition

In this work, the environmental condition includes illumination and weather conditions. We use the image intensity to quantify the illumination condition. The feature of image intensity is represented by the mean intensity \( m_{\text{bg}} \) of the global background image. \( m_{\text{bg}} \) can characterize the illumination condition. For example, the mean intensity in daytime is usually higher than that at night. Besides, in order to evaluate the intensity in the local virtual loop, we calculate the local mean intensity \( m_{\text{lbg}} \) based on the background image within the virtual loop.

Meanwhile, we use the image contrast to quantify the weather condition. The feature of image contrast is represented by the standard deviation \( \text{sd}_{\text{bg}} \) of Ede edge magnitudes of the global background image. \( \text{sd}_{\text{bg}} \) can characterize the weather condition. For example, the image contrast in sunny day is usually larger than that in foggy day. Besides, in order to evaluate the image contrast in the local virtual loop, we calculate the local standard deviation \( \text{sd}_{\text{lbg}} \) of Ede edge magnitudes based on the background image within the virtual loop.

So far, we have extracted 14 features from the video images, forming a feature vector for each virtual loop at each time. For reading convenience, we summarize these features in Table 1. The next step is to build an effective classifier that recognizes the feature vector as vehicle or nonvehicle.

<table>
<thead>
<tr>
<th>Index</th>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fr</td>
<td>Foreground rate inside the virtual loop</td>
</tr>
<tr>
<td>2</td>
<td>fr_{a1}</td>
<td>Foreground rate on line ( a_1 )</td>
</tr>
<tr>
<td>3</td>
<td>fr_{a2}</td>
<td>Foreground rate on line ( a_2 )</td>
</tr>
<tr>
<td>4</td>
<td>fr_{b1}</td>
<td>Foreground rate on line ( b_1 )</td>
</tr>
<tr>
<td>5</td>
<td>fr_{b2}</td>
<td>Foreground rate on line ( b_2 )</td>
</tr>
<tr>
<td>6</td>
<td>sd</td>
<td>Standard deviation of edge magnitudes for all foreground pixels inside the virtual loop</td>
</tr>
<tr>
<td>7</td>
<td>sd_{a1}</td>
<td>Standard deviation of edge magnitudes for all foreground pixels on line ( a_1 )</td>
</tr>
<tr>
<td>8</td>
<td>sd_{a2}</td>
<td>Standard deviation of edge magnitudes for all foreground pixels on line ( a_2 )</td>
</tr>
<tr>
<td>9</td>
<td>sd_{b1}</td>
<td>Standard deviation of edge magnitudes for all foreground pixels on line ( b_1 )</td>
</tr>
<tr>
<td>10</td>
<td>sd_{b2}</td>
<td>Standard deviation of edge magnitudes for all foreground pixels on line ( b_2 )</td>
</tr>
<tr>
<td>11</td>
<td>( m_{\text{bg}} )</td>
<td>Mean intensity of the global background image</td>
</tr>
<tr>
<td>12</td>
<td>( m_{\text{lbg}} )</td>
<td>Mean intensity of the local background image within the virtual loop</td>
</tr>
<tr>
<td>13</td>
<td>( \text{sd}_{\text{bg}} )</td>
<td>Standard deviation of edge magnitudes of the global background image</td>
</tr>
<tr>
<td>14</td>
<td>( \text{sd}_{\text{lbg}} )</td>
<td>Standard deviation of edge magnitudes of the local background image within the virtual loop</td>
</tr>
</tbody>
</table>
4. Classifier Construction via Supervised Learning

From traffic videos captured under a range of illumination and weather conditions, we can collect a set of training samples. Specifically, for each video, we configure a virtual loop on the first image frame, and then conduct background subtraction and feature extraction. Meanwhile, we observe the virtual loop with our eyes. If any part of a vehicle enters the middle subregion of the virtual loop, the corresponding feature vector is labeled as positive; otherwise, it is labeled as negative.

Based on the set of training samples, we proceed to construct a pattern classifier based on supervised learning. In this work, we train three neuro-fuzzy networks to construct a combined classifier. Our consideration is twofold.

In the first place, neuro-fuzzy network is a powerful classifier and has flexible structure. It integrates the low-level learning ability of neural networks and the high-level reasoning ability of fuzzy systems. As a universal estimator, it solves the modeling problem using a linguistic model consisting of a set of IF-THEN rules instead of a complex mathematical model. Besides, the structure of neuro-fuzzy network can be flexibly organized through growing-and-pruning. This is crucial for the model to be adaptive, compact, and efficient in complex dynamic environments. Although some other kinds of classic pattern classifiers, such as support vector machine and random forest, also have strong classification ability, they lack the critical flexibility of online self-organization.

The second consideration is that ensemble learning is an effective strategy to increase the classification accuracy and reduce the risk of overfitting. More importantly, it can be integrated seamlessly with the semi-supervised learning mechanism in this work. This will be discussed in Sec. 5.

To encourage diversity among the component classifiers, the training dataset is randomly separated into three disjointed subsets of equal size. Alternately, one subset is used to train a component classifier and the other two are used for validation. Repeating this process, we build three neuro-fuzzy networks, which constitute the combined classifier. When the system runs, new samples are classified via majority vote amongst the three component classifiers, as shown in Fig. 8.

![Fig. 8. Workflow of sample classification using the combined classifier.](image-url)
4.1. Structure of the neuro-fuzzy network

We build a five-layered structure for the neuro-fuzzy network, as shown in Fig. 9. It consists of the input layer, condition layer (which performs the fuzzification), rule layer (where each node denotes a fuzzy rule), consequence layer, and output layer (which performs the defuzzification). According to this structure, the crisp inputs are first fuzzified into fuzzy inputs, and then transformed into fuzzy outputs through a set of IF-THEN rules. The fuzzy outputs are finally defuzzified into crisp outputs.

In this work, the crisp input is a 14-dimensional feature vector $X = [x_1, x_2, \ldots, x_{14}]$, which corresponds to the image features as described in Sec. 3. The crisp output is a one-dimensional variable $Y = [y]$, which labels the sample as positive or negative. We use the terms $n_1$, $n_2$, $n_3$, $n_4$, and $n_5$ to denote the number of neurons in the five layers. Obviously, we have $n_1 = 14$ and $n_5 = 1$. Let $n_2 = \sum_{i=1}^{14} L_i$, where $L_i$ denotes the number of fuzzy sets for each input dimension. Since vehicle detection is a binary classification problem, $n_4 = 2$ is just the number of fuzzy sets for the crisp output. In this work, the Gaussian membership function (MF) is used in the condition and consequence layers. The centers and widths of the MFs are denoted as $(c^H_{i,j}, \sigma^H_{i,j})$ (the $i$th input dimension and the $j$th MF) and $(c^IV_l, \sigma^IV_l)$ (the $l$th MF, where $c^IV_1 = 1$, $c^IV_2 = -1$, and $\sigma^IV_l = \text{eps}$) for the two layers. Here “eps” is called machine zero, which means the smallest positive number that can be represented by a computer. By denoting $iL_k(i)$ as the fuzzy set in the $i$th input dimension of the $k$th rule, and $oL_k$ as the output fuzzy set of the $k$th rule, the final output $o^V$ is computed as

$$o^V = \frac{\sum_{k=1}^{n_3} c^IV_{oL_k} \times f^III_k}{\sum_{k=1}^{n_3} f^III_k},$$

(1)

Fig. 9. Structure of the five-layered neuro-fuzzy network.
where $f_{k}^{III} = \prod_{i=1}^{14} \mu_{i,L_{k}(i)}^{II} = \prod_{i=1}^{14} \exp \left\{ - \frac{(x_{i} - c_{i,L_{k}(i)})^{2}}{(\sigma_{i,L_{k}(i)})^{2}} \right\}$ denotes the firing strength of the rule by the feature vector $X = [x_{1}, x_{2}, \ldots, x_{14}]$. $\mu_{i,L_{k}(i)}^{II} = \exp \left\{ - \frac{(x_{i} - c_{i,L_{k}(i)})^{2}}{(\sigma_{i,L_{k}(i)})^{2}} \right\}$ denotes the Gaussian MF of the $i$th input dimension.

4.2. Clustering the training data

In this work, we adopt a data-driven clustering method to partition the feature space. We do not define the MFs for each input dimension by hand, but employ the knowledge embedded in the training data to produce them automatically. Besides, we do not use the $K$-means clustering method, because it requires predetermined the number “$K$” of clusters for each input dimension. Instead, we use a fully automatic method to cluster the training data and generate the initial MFs.

For the $i$th input dimension, the number of clusters is initialized to zero at the beginning. After randomly sorting the training data, we take the first data $x_{i,1}$. At this point, we create the first fuzzy cluster $C_{i,1}$ via

$$c_{i,1} = x_{i,1},$$
$$\sigma_{i,1} = \frac{\text{max}_{i} - \text{min}_{i}}{10},$$

where $c_{i,1}$ and $\sigma_{i,1}$ are the center and width of the Gaussian MF regarding $C_{i,1}$. According to (2), a newly created MF is centered upon the present value, and has an initial width of $(\text{max}_{i} - \text{min}_{i})/10$, where $\text{max}_{i}$ and $\text{min}_{i}$ denote the maximum and minimum values of the training data in the $i$th input dimension.

We take the remaining training data one by one, and match them with the existing fuzzy clusters. For any data $x_{i,n}$, we search for the cluster $C_{i,j}$ that has the maximum MF $m(x_{i,n}, C_{i,j}) = \exp \left\{ - \frac{(x_{i,n} - c_{i,j})^{2}}{\sigma_{i,j}^{2}} \right\}$. If $m(x_{i,n}, C_{i,j})$ is greater than $\exp(-6.25)$, that is, $x_{i,n}$ falls into the range of 2.5 standard deviations of the Gaussian MF regarding $C_{i,j}$, then $x_{i,n}$ is used to update the center and width of $C_{i,j}$ via

$$c_{i,j} = c_{i,j} + \rho_{i,j} (x_{i,n} - c_{i,j}),$$
$$\sigma_{i,j}^{2} = \sigma_{i,j}^{2} + \rho_{i,j} \{(x_{i,n} - c_{i,j})^{2} - \sigma_{i,j}^{2}\},$$

where $\rho_{i,j} = \max(1/t_{i,j}, 0.01)$ is the learning rate regarding $C_{i,j}$, while $t_{i,j}$ is the hitting number of $C_{i,j}$. On the other hand, if $m(x_{i,n}, C_{i,j})$ is less than $\exp(-6.25)$, then a new fuzzy cluster is created, with $(x_{i,n}, \frac{\text{max}_{i} - \text{min}_{i}}{10})$ as its initial center and width. This process repeats until the last training data is taken.

We find that for most of the input dimensions, the final number of clusters is four or five. Occasionally, it is three or six. Figure 10 shows the fuzzy partitioning of two input dimensions regarding foreground area and texture change. The blue bars are the feature histogram with the maximum frequency scaled to 1, and the red curves denote the resulting Gaussian MFs. As can be seen, the Gaussian MFs match the feature histograms well, and present clear physical meanings. In Fig. 10(a), the feature space is partitioned into four fuzzy sets: very small, small, large, and very
large. In Fig. 10(b), the feature space is partitioned into five fuzzy sets: very small, small, medium, large, and very large.

4.3. Rule generation

After fuzzy partitioning of the input dimension, we proceed with rule generation. The rule generation method in this work is similar to that in Ref. 31. The rule base is empty at the beginning. Take every training tuple \((X, Y)\) and find the best matched fuzzy cluster for every input–output dimension. A fuzzy rule \(R/C_3\) is formulated, with the best matched fuzzy clusters in the 14 input dimensions and the single output dimension as the condition and consequence segments. If \(R/C_3\) is novel, it is inserted into the rule base; otherwise, it is ignored.

After generating the initial rule base, we perform consistency check. If two rules have the same conditions but conflicting consequences, the rule of less importance will be deleted from the rule base. In contrast to Ref. 31, this paper adopts a different definition of rule importance. The importance of a rule is used to evaluate the degree of coverage of all the training samples by that rule. The degree of coverage of the \(n\)th training tuple \((X_n, Y_n)\) by the \(k\)th rule is defined as the minimum of the condition MF values and the consequence MF value regarding that rule,

\[
I_{k,n} = \min \left\{ \min_{i=1, \ldots, 14} \left( \mu^H_{1,iL_i(i)} \right), \mu_{oL_k} \right\},
\]

Then, the importance of the \(k\)th rule is defined as the sum of \(I_{k,n}\) over all the \(N\) training samples,

\[
I_k = \sum_{n=1}^{N} I_{k,n}.
\]
The resulting fuzzy rules (of Mandeni type) have the following form:

\[ R_k : \text{IF } x_1 \text{ is } C_{k1}^I \text{ and } \ldots \text{ and } x_{14} \text{ is } C_{k14}^I, \quad \text{THEN } y \text{ is } C_{kY}^I, \]

where \( C_{ki}^I \) is the \( i \)th condition segment and \( C_{kY}^I \) is the consequence segment, respectively, in the \( k \)th rule.

### 4.4. Parameter tuning

After generation of MFs and fuzzy rules, we fine tune the parameters in the neuro-fuzzy network using the gradient descent method. The objective function is defined as the squared error on each training tuple \((X, Y)\):

\[
E = \frac{1}{2} (o^V - y)^2 = \frac{1}{2} e^2, \tag{6}
\]

where \( e = o^V - y \). This operation aims to minimize \( E \) on the training dataset and to avoid overfitting simultaneously.

In the gradient descent process, the center and width of every MF in the condition layer are iteratively updated according to the partial derivative of \( E \) to the corresponding parameter, while the MFs in the consequence layer keep unchanged. Based on the chain rule, we have

\[
\frac{\partial E}{\partial c_{i,iL_k(i)}^I} = \frac{\partial E}{\partial o^V} \frac{\partial o^V}{\partial f_k^I} \frac{\partial f_k^I}{\partial \mu_{i,iL_k(i)}^I} \frac{\partial \mu_{i,iL_k(i)}^I}{\partial c_{i,iL_k(i)}^I} = 2e f_k^I (c_{oL_k} - o^V) (x_i - c_{i,iL_k(i)}) / \left( \sum_{k'=1}^{n_3} f_k^I (\sigma_{i,iL_k(i)}^I)^2 \right),
\]

\[
\frac{\partial E}{\partial \sigma_{i,iL_k(i)}^I} = \frac{\partial E}{\partial o^V} \frac{\partial o^V}{\partial f_k^I} \frac{\partial f_k^I}{\partial \mu_{i,iL_k(i)}^I} \frac{\partial \mu_{i,iL_k(i)}^I}{\partial \sigma_{i,iL_k(i)}^I} = 2e f_k^I (c_{oL_k} - o^V) (x_i - c_{i,iL_k(i)})^2 / \left( \sum_{k'=1}^{n_3} f_k^I (\sigma_{i,iL_k(i)}^I)^3 \right). \tag{7}
\]

It is clear that the forms of partial derivatives are very simple. The center and width of MF are updated using (8),

\[
c_{i,iL_k(i)}^I = c_{i,iL_k(i)}^I - \eta \frac{\partial E}{\partial c_{i,iL_k(i)}^I},
\]

\[
\sigma_{i,iL_k(i)}^I = \sigma_{i,iL_k(i)}^I - \eta \frac{\partial E}{\partial \sigma_{i,iL_k(i)}^I}, \tag{8}
\]

where \( \eta = 0.05 \) is a learning rate.

In general, the solution evolves toward local minima. We use the validation dataset to determine the iteration stopping point, where the mean squared error (MSE) on the validation dataset arrives at a minimum and does not decrease anymore in the following \( M \) iterations (\( M = 50 \) in this work). In this way, overfitting the training data can be avoided in some degree. Figure 11 shows the changing progress of MSE on the training and validation datasets during parameter tuning for one of the neuro-fuzzy networks. After 134 iterations, the MSE on the validation dataset arrives at a minimum of 0.1493, and does not decrease in the following 50 iterations. Hence, the 134th iteration is deemed as the iteration stopping point.
5. Classifier Optimization via Incremental Semi-Supervised Learning

Due to the complexity of real-world environments, the pattern classifier constructed via supervised learning is a general-purpose one. Although it has learnt the characteristics of many typical traffic environments, it cannot perfectly fit the current scenario. For each neuro-fuzzy network, some fuzzy rules no longer match with the current traffic environment and become redundant, while other rules should be tuned and optimized. Besides, some new rules need to be created and added to the rule base. For better performance, we optimize the structures and parameters of neuro-fuzzy networks when the classifier runs in a specific environment.

5.1. Mechanism of semi-supervised learning

In this work, the vehicle detection issue is addressed from the view of semi-supervised learning. Our consideration is twofold. First, although we get a training dataset by labeling a certain number of training samples by hand, it is impossible to characterize all traffic environments with finite training data. Second, abundant unlabeled data are readily available when the vehicle detection system runs in practice. Semi-supervised learning has a potential to employ large amount of unlabeled data, together with finite labeled data, to build a better classifier. It is regarded as a good idea to reduce human labor and improve the classification accuracy.\textsuperscript{11,12,18,19,21,24,25,35,40,41}

In general, semi-supervised learning methods include EM with generative models, self-training, co-training, transductive support vector machines, and graph-based methods.\textsuperscript{41} Ideally, we should use that method whose model assumptions fit the problem structure best. Since the extracted features have a complex and unknown

![Changing progress of MSE on the training and validation datasets.](image)

Fig. 11. Changing progress of MSE on the training and validation datasets.
distribution in the 14-dimensional space, EM with generative models is not applicable to this study. Self-training can easily be misled by classification errors. Transductive support vector machines and graph-based methods cannot endow the classifier with online self-organization ability, which is crucial here. Besides, graph-based methods are transductive in nature and cannot be used to predict on unseen data.\textsuperscript{11,18,19,24,25,35,40,41} In light of these, we focus on the co-training method, which has inspired many researches in computer vision, e.g. shape retrieval\textsuperscript{2} and visual tracking.\textsuperscript{39}

Standard co-training assumes that the features of interest can be split into two conditionally independent sets and each sub-feature set is sufficient to train a good classifier.\textsuperscript{11,41} However, the independence and sufficiency assumptions do not hold in this study. Zhou and Li proposed “tri-training” to relax this assumption.\textsuperscript{40} Unlike co-training that uses two classifiers, tri-training uses three classifiers. If two of the classifiers agree on the classification of an unlabeled data, that classification is used to train the third classifier. Tri-training avoids the need of two views in the feature set. It can be applied to common classification scenarios, and can achieve better generalization ability by combining three classifiers.

The original tri-training was used to process a fixed dataset. It assumed that unlabeled data hold the same distribution as that held by labeled data, and did not consider the unseen data.\textsuperscript{40} These ideas are not reasonable for real-world applications. In this study, when the vehicle detection system runs, abundant new data will emerge and their distribution may differ from that of the previously labeled data. The criterion that two classifiers agree on the labeling of new data can produce many noisy labels and mislead the third classifier. Hence, we need to modify the original tri-training to make it applicable to this study. As shown in Fig. 12, if all the three classifiers agree on the classification of an unlabeled data, that classification is used to retrain the classifier whose output is the farthest away from the predicted label (1 or $-1$); otherwise, this data is not used for retraining the classifier. This operation is appropriate because abundant new data will emerge as the system runs. This novel mechanism of tri-training has two advantages.

(1) Since the possibility that three component classifiers all make false predictions on an unlabeled data is quite low, the novel mechanism is able to better prevent the classifiers from being misled.

(2) The classifier whose output is the farthest away from the predicted label is the one that needs optimization the most. Retraining only this classifier will benefit to reducing the computational cost and to improving efficiency.

5.2. Structure optimization

The classifier is retrained incrementally, i.e. provided with only one training sample at a time. The retraining process performs online optimization on the structures and parameters of the neuro-fuzzy networks. Structure optimization is performed on the
condition, rule, and consequence layers. This operation includes three main steps: updating the importance of existing rules, inserting a newly created rule, and deleting obsolete rules.

5.2.1. Updating the importance of existing rules

As stated in Sec. 4.3, the importance of a fuzzy rule is used to evaluate the degree of coverage of the training samples by that rule. After classifier construction via supervised learning, we get a fuzzy rule base. The importance of each rule is initialized to 10.0.

When the vehicle detection system runs, positive and negative samples will emerge in different frequencies. For example, when the traffic is sparse, the emergency frequency of negative samples (non-vehicle pattern) will be much higher than that of positive samples (vehicle pattern). As a result, the importance of rules regarding vehicle pattern and nonvehicle pattern should be updated respectively. If the current training sample is positive, the importance of rules regarding vehicle pattern is updated. Otherwise, the importance of rules regarding nonvehicle pattern is updated. Referring to (4), the importance $I_k$ of the $k$th rule is updated via

$$I_k = \lambda I_k + \min \left\{ \min_{i=1,\ldots,14} \left( \mu_{ILk(i)}^H, \mu_{OLk}^N \right) \right\},$$

where $\lambda$ is the memory rate, and $\min\{\ldots\}$ denotes the degree of coverage of training sample $\{[x_1, x_2, \ldots, x_1], y\}$ by the $k$th rule. $\lambda$ is a key coefficient that serves to regulate the balance between preserving old knowledge and learning new knowledge for the neuro-fuzzy network. On the one hand, $\lambda$ must be large enough, in order to
keep the previously learnt and still valid knowledge in the rule base. On the other hand, \( \lambda \) must not be too large; otherwise the previously learnt but already outdated knowledge would stay in the rule base for long and degrade the classifier performance. Through elaborate evaluations, we find \( \lambda = 0.999 \) is a proper assignment of value. That is to say, the rule is forgotten by 0.1% at a time. According to (9), if a rule is repeatedly activated by new training samples, its importance will increase; otherwise, its importance will decrease.

5.2.2. Inserting a newly created rule

For the \( i \)th input dimension of the current training sample \( \{[x_1, x_2, \ldots, x_I], y\} \), we find the best matched fuzzy set that corresponds to the maximum MF value. If the maximum MF value is greater than \( \exp(-6.25) \), the best matched fuzzy set is selected as one of the condition segments of a fuzzy rule \( R^* \). If the maximum MF value is less than \( \exp(-6.25) \), a new fuzzy set is created with \( (x_i, \frac{\text{max} - \text{min}}{10}) \) (see (2)) as its center and width, and this fuzzy set serves as one of the condition segments of \( R^* \). For the unique output dimension \( y \) of the current training sample, the same operation is performed. After \( R^* \) is generated, we check whether it is novel, that is, whether it has the same condition and consequence segments as an existing rule in the rule base. If \( R^* \) is novel, we insert it into the rule base; otherwise, we ignore it. The importance of a newly created rule is initialized to 1.0.

5.2.3. Deleting obsolete rules

During structure optimization, when a fuzzy rule no longer matches with the current traffic environment, its importance will decline gradually. If the importance of a rule becomes lower than 0.01, we think that rule is obsolete and should be deleted from the rule base. After the obsolete rules have been deleted from the rule base, we check whether there exist orphaned fuzzy sets in the condition layer of the neuro-fuzzy network. Orphaned fuzzy sets are those that are not assigned to the condition segment of any fuzzy rule. If a fuzzy set is orphaned, it should be removed from the condition layer.

5.3. Parameter optimization

As part of the classifier retraining process, parameter optimization is performed instantaneously after structure optimization. The gradient descent method is used to fine tune the center and width of every fuzzy MF in the condition layer. As discussed in Sec. 4, the objective function is (6), the partial derivative of \( E \) to the corresponding parameter is (7), and the parameter updating equation is (8). Here the learning rate \( \eta \) is set to 0.01. Since we have only one training sample at a time, the parameters are optimized incrementally. The gradient descent is performed only once and then stopped. The iteration number is 1. After that, we wait for the next training sample to retrain the classifier. This operation is reasonable because abundant new samples will emerge as the system runs in practice.
In short, the proposed optimization method is completely data-driven and is able to track the small changes of training samples, especially in complex dynamic environments. Details of the classifier retraining process are summarized as a flowchart in Fig. 13.

6. Experimental Results and Discussion

To verify the proposed approach, we conduct experiments on many traffic videos with various illumination and weather conditions. These videos are divided into two separate groups: training videos and testing videos. The training videos are used to collect training samples and construct the general-purpose classifier via supervised learning. All the training videos are captured by us. The testing videos are used to evaluate the accuracy and adaptability of the online-optimized classifier in specific traffic environments. Some testing videos are benchmark videos, while others are captured by us.

As stated in Sec. 3, each training sample consists of 14 features and one label, and Fig. 3 shows the distinction between positive and negative samples. To create the training dataset, we collected 1500 positive samples and 1500 negative samples from...
the training videos. Collecting more training samples is certain to improve the classification accuracy, but is more time consuming. Based on the training dataset, we constructed three neuro-fuzzy networks (as a combined classifier) via supervised learning. As the system runs, the neuro-fuzzy networks classify newly emerged data into vehicle and nonvehicle, and are optimized automatically via incremental semi-supervised learning.

6.1. Rule changes during semi-supervised learning

In the semi-supervised learning process, the rule base of each neuro-fuzzy network changes gradually and adapts to specific traffic environment. To understand this better, we conduct experiment on a testing video “AVSS 2007”, and record the changing rule numbers of three neuro-fuzzy networks. This video belongs to the sunny type, but it contains global illumination changes that occur suddenly and irregularly and cause significant challenges, as shown in Fig. 14(b). We guess the global illumination changes are due to the sun getting into or out of a cloud. Hence, the environment is complex and dynamic. In this experiment, three component classifiers make an agreement on the classification of 98.4% new unseen data. That is to say, 98.4% new samples are used for classifier retraining.

Figure 14(c) shows the plots of rule number versus frame number during semi-supervised learning. At the beginning, the three neuro-fuzzy networks contain 837, 855, and 824 rules, respectively. These rules are generated via supervised learning, as discussed in Sec. 4. As the system runs, the neuro-fuzzy networks are optimized automatically, and new rules are created and added to the rule base, so that the rule number increases gradually. After about 6000 frames (i.e. 4 min), a steep drop appears on each curve. The drops indicate that most of the previously learnt rules regarding nonvehicle pattern have become obsolete and been deleted. Later on, after about 12 000 frames (i.e. 8 min), another steep drop appears on each curve. But this time, the drops indicate that most of the previously learnt rules regarding vehicle pattern have become obsolete and been deleted. At this point, most of the previously learnt but already outdated rules have been deleted from the rule base. The neuro-fuzzy networks have learnt new knowledge in the specific traffic environment. Afterwards, the rule numbers of three neuro-fuzzy networks continue to change up and down, eventually achieving stability. The rule base would not grow infinitely, because $\lambda$ in (9) plays the role of maintaining a subtle balance for the coexistence of past knowledge and future knowledge to achieve a compact and up-to-date system. This experiment demonstrates the adaptability of the proposed classifier in specific traffic environment.

6.2. Sample classification accuracy

After the neuro-fuzzy networks have been sufficiently retrained in specific traffic environments, we record the sample classification results by superimposing the
output states of virtual loops on the raw images, as shown in Fig. 15. We use green virtual loops to denote vehicle and blue ones to denote nonvehicle. These testing videos are representative of common environmental conditions, including sunny, cloudy, night, foggy, snowy, rainy, and tunnel conditions. Note that the seven videos (namely, “Highway I”, “Highway”, “AVSS 2007”, “WinterStreet”, “Dtneu_nebel”, “Dtneu_schnee”, and “Blizzard”) are publicly available on the Internet.

With regard to sample classification, we compare five classification approaches: support vector machine,\(^6\) random forest,\(^4\) supervised neuro-fuzzy networks (Sec. 4), neuro-fuzzy networks plus original tri-training,\(^{20}\) and the proposed approach (Sec. 5). The front three are supervised learning approaches, while the last two are semi-supervised learning approaches. For a fair comparison, the same training dataset of size 3000 is used in the experiment.
To get an objective evaluation, the performance of various approaches is compared quantitatively. Sample classification accuracy $\xi$ is used for quantitative evaluation, which is defined as

$$\xi = \frac{\text{number of correctly classified samples}}{\text{number of samples}} \times 100\%.$$  \hspace{1cm} (10)

We compute $\xi$ on the testing videos and present the results for the five approaches in Table 2. We also compute the average classification accuracy for each approach. The
Table 2. Sample classification accuracy for the five approaches.

<table>
<thead>
<tr>
<th>Testing Video (see Fig. 15)</th>
<th>Environment Description</th>
<th>Number of Frames</th>
<th>Number of Samples</th>
<th>Support Vector Machine (%)</th>
<th>Random Forest (%)</th>
<th>Supervised Neuro-Fuzzy Networks (%)</th>
<th>Neuro-Fuzzy Networks Plus Original Tri-training (%)</th>
<th>Proposed Approach (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway I</td>
<td>Long shadow</td>
<td>440</td>
<td>880</td>
<td>95.34</td>
<td>98.07</td>
<td>97.16</td>
<td>96.14</td>
<td>97.39</td>
</tr>
<tr>
<td>Highway</td>
<td>Short shadow</td>
<td>1700</td>
<td>3400</td>
<td>98.56</td>
<td>99.00</td>
<td>98.68</td>
<td>99.26</td>
<td>99.47</td>
</tr>
<tr>
<td>AVSS 2007</td>
<td>Sunny</td>
<td>4000</td>
<td>8000</td>
<td>97.24</td>
<td>97.93</td>
<td>97.78</td>
<td>98.04</td>
<td>98.19</td>
</tr>
<tr>
<td>WinterStreet</td>
<td>Night</td>
<td>1785</td>
<td>3570</td>
<td>91.79</td>
<td>91.88</td>
<td>88.74</td>
<td>89.79</td>
<td>90.69</td>
</tr>
<tr>
<td>Night</td>
<td>Night</td>
<td>2000</td>
<td>6000</td>
<td>98.42</td>
<td>97.57</td>
<td>95.93</td>
<td>98.08</td>
<td>98.43</td>
</tr>
<tr>
<td>Foggy</td>
<td>Normal fog</td>
<td>2000</td>
<td>4000</td>
<td>86.63</td>
<td>88.18</td>
<td>87.33</td>
<td>87.55</td>
<td>87.93</td>
</tr>
<tr>
<td>Dtnen_nebel</td>
<td>Heavy fog</td>
<td>349</td>
<td>698</td>
<td>96.42</td>
<td>95.99</td>
<td><strong>96.85</strong></td>
<td>96.56</td>
<td>96.56</td>
</tr>
<tr>
<td>Dtnen_schnee</td>
<td>Heavy snowfall</td>
<td>299</td>
<td>598</td>
<td>96.66</td>
<td>94.98</td>
<td>95.48</td>
<td>96.66</td>
<td><strong>96.99</strong></td>
</tr>
<tr>
<td>Blizzard</td>
<td>Snow on road</td>
<td>4000</td>
<td>8000</td>
<td>98.18</td>
<td>98.84</td>
<td>98.45</td>
<td>99.84</td>
<td><strong>99.96</strong></td>
</tr>
<tr>
<td>Rainy</td>
<td>Light rainfall</td>
<td>4000</td>
<td>12000</td>
<td>98.97</td>
<td><strong>99.13</strong></td>
<td>98.53</td>
<td>98.82</td>
<td>99.04</td>
</tr>
<tr>
<td>Tunnel</td>
<td>Tunnel</td>
<td>4000</td>
<td>8000</td>
<td>99.88</td>
<td><strong>99.89</strong></td>
<td>99.64</td>
<td>99.69</td>
<td>99.79</td>
</tr>
<tr>
<td>Average</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>96.45</td>
<td>96.74</td>
<td>96.17</td>
<td>95.88</td>
<td><strong>97.01</strong></td>
</tr>
</tbody>
</table>
highest accuracy on each row is annotated in bold. From Table 2, we obtain three findings:

1. Among the three supervised learning approaches, random forest (which is composed of 500 decision trees and results in an average classification accuracy 96.74%) outperforms support vector machine (whose average classification accuracy is 96.45%) and supervised neuro-fuzzy networks (whose average classification accuracy is 96.17%). This proves that random forest has strong generalization ability and high accuracy when classifying new unseen data.

2. The proposed approach is effective to improve the pattern classifier (which is composed of three neuro-fuzzy networks) in real-world applications, causing the average classification accuracy to rise from 96.17% to 97.01%. By contrast, original tri-training\textsuperscript{40} causes the average classification accuracy to decline slightly from 96.17% to 95.88%. The most likely reason is that in Ref. 40 the criterion that two classifiers agree on the labeling of new data produces noisy labels and misleads the third classifier. This finding validates the necessity of modifying the original tri-training mechanism for online optimizing the pattern classifier in complex environments.

3. The proposed approach has the highest average classification accuracy among the five approaches. Admittedly, the average classification accuracy of supervised neuro-fuzzy networks is lower than that of random forest. However, the online optimized neuro-fuzzy networks based on the proposed approach achieve higher average classification accuracy than that achieved by random forest (97.01% versus 96.74%). These validate the significance of classifier optimization and the adaptability of the proposed approach in complex environments.

According to only the accuracy metric, the improvement may seem marginal. Hence we make qualitative comparisons on some testing videos, so as to reveal the underlying error sources of our approach and two classic ones (SVM and random forest). In Fig. 16, the red arrows indicate classification errors. In video “Highway I”, SVM and random forest often recognize moving shadows falsely as vehicle, while the proposed approach seldom does that. However, the dark vehicle wheel is recognized as nonvehicle by the proposed approach, perhaps because it has a similar color appearance with the shadows. In videos “WinterStreet” and “Foggy”, SVM and random forest often recognize vehicle headlight reflections falsely as vehicle, while the proposed approach does that less frequently. In video “Rainy”, SVM and random forest often recognize the soft shadows falsely as vehicle, while the proposed approach does that much less. However, the low-contrast vehicle bodies can be recognized as nonvehicle by the proposed approach, perhaps because they cause low texture changes, similar to those caused by moving shadows or headlight reflections. Or to put it another way, when the decision boundary is fine-tuned to classify moving shadows and headlight reflections correctly, the low-contrast vehicle bodies can be recognized falsely.
From Table 2, Figs. 15 and 16, it can be seen that the proposed approach is quite accurate and robust to distinguish between vehicle and nonvehicle patterns in complex environments. The impacts of various adverse aspects such as moving shadows, headlight reflections, and fog/snow/rain, can be mitigated effectively. In brief, the proposed approach is able to work in all-weather conditions.

6.3. Vehicle counting accuracy

When the system runs, we obtain the classifier output (1 or −1) for each virtual loop at each instant. In the time dimension, the classifier outputs corresponding to a virtual loop are concatenated into a string. Ideally, when a vehicle passes, the virtual loop’s state should be transmitted from “−1 − 1 − 1 . . .” (nonvehicle), to “111 . . .” (vehicle), and finally to “−1 − 1 − 1 . . .” (nonvehicle) again. However, there may be accidental classification errors in the string of classifier outputs. Hence we perform median filtering on the sequential classification results of every virtual loop. Median filtering is simple yet effective to eliminate the impacts of isolated classification errors. The window size of median filtering is set to 3 for “Highway I” and to 5 for the other testing videos. Suppose a vehicle passes a virtual loop and we get a string of classifier outputs “…−1 − 1 − 111 − 111111 − 111 − 11 − 1 − 1 − 1 . . .”, then the
filtering result is “... – 1 – 1 – 1 – 111111111111 – 1 – 1 – 1 – 1...”. This operation is illustrated in Fig. 17. Finally, we count a vehicle by searching for a continuous segment of label “1”.

Table 3 presents the vehicle counting results of the proposed approach and three existing approaches: Hsu et al.,16 Cho et al.,9 and Yuan et al.37 Counting error rate $\zeta$ is used here to evaluate the vehicle counting accuracy, which is defined as:

$$\zeta = \frac{|\text{counting result} - \text{ground truth}|}{\text{ground truth}} \times 100\%.$$  \hfill (11)

Through careful analysis, we obtain the following findings:

(1) The proposed approach performs consistently well on the entire set of testing videos. Due to the superior performance of the extracted features and the constructed pattern classifier, our approach is able to count vehicles accurately in complex dynamic environments. Particularly, if the traffic flow is sparse and there is no vehicle occlusion, the counting error rate of our approach is close to 0.

(2) The approaches in Refs. 9 and 16 are sensitive to the quality of foreground segmentation. When foreground segmentation is done well, such as at sunny noon or in cloudy daytime, they can achieve comparable performance to our approach. However, foreground segmentation is easily disturbed by many adverse aspects such as moving shadows, headlight reflections, and fluttering snowflakes, to become unreliable. Under these situations, our approach performs much better. As Fig. 18 shows, the fluttering snowflakes in “Dtnue_schnee” are segmented as foreground, causing the approach in Ref. 16 to detect snowflakes falsely as vehicle.

(3) The approaches in Refs. 9 and 16 cannot be applied to some scenarios, because Ref. 16 does not mention nighttime vehicle detection, and Ref. 11 detects vehicles at night based on a pair of vehicle headlights. Consequently, in the videos “Night” and “Tunnel”, where the vehicle headlights are turned on but...
<table>
<thead>
<tr>
<th>Testing Video (see Fig. 15)</th>
<th>Number of Frames</th>
<th>Ground Truth Vehicle Number</th>
<th>Counting Result</th>
<th>Error Rate (%)</th>
<th>Counting Result</th>
<th>Error Rate (%)</th>
<th>Counting Result</th>
<th>Error Rate (%)</th>
<th>Counting Result</th>
<th>Error Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway I</td>
<td>440</td>
<td>63</td>
<td>58</td>
<td><strong>7.94</strong></td>
<td>76</td>
<td><strong>20.63</strong></td>
<td>72</td>
<td><strong>14.29</strong></td>
<td>56</td>
<td><strong>11.11</strong></td>
</tr>
<tr>
<td>Highway</td>
<td>1700</td>
<td>27</td>
<td>27</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>29</td>
<td>7.41</td>
</tr>
<tr>
<td>AVSS 2007</td>
<td>4000</td>
<td>60</td>
<td>52</td>
<td><strong>13.33</strong></td>
<td>75</td>
<td>25</td>
<td>72</td>
<td>20</td>
<td>50</td>
<td>16.67</td>
</tr>
<tr>
<td>Cloudy</td>
<td>2000</td>
<td>55</td>
<td>56</td>
<td><strong>1.82</strong></td>
<td>56</td>
<td><strong>1.82</strong></td>
<td>56</td>
<td><strong>1.82</strong></td>
<td>58</td>
<td>5.45</td>
</tr>
<tr>
<td>WinterStreet</td>
<td>1785</td>
<td>61</td>
<td>65</td>
<td>6.56</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>45</td>
<td>26.23</td>
</tr>
<tr>
<td>Night</td>
<td>2000</td>
<td>37</td>
<td>38</td>
<td>2.7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>41</td>
<td>10.81</td>
</tr>
<tr>
<td>Foggy</td>
<td>2000</td>
<td>30</td>
<td>25</td>
<td><strong>16.67</strong></td>
<td>19</td>
<td>36.67</td>
<td>21</td>
<td>30</td>
<td>47</td>
<td>56.67</td>
</tr>
<tr>
<td>Dtnue_nebel</td>
<td>349</td>
<td>5</td>
<td>4</td>
<td><strong>20</strong></td>
<td>4</td>
<td><strong>20</strong></td>
<td>4</td>
<td><strong>20</strong></td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>Dtnue_schnee</td>
<td>299</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>35</td>
<td>337.5</td>
<td>22</td>
<td>175</td>
<td>9</td>
<td>12.5</td>
</tr>
<tr>
<td>Blizzard</td>
<td>4000</td>
<td>26</td>
<td>26</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>140</td>
<td>438.46</td>
</tr>
<tr>
<td>Rainy</td>
<td>4000</td>
<td>68</td>
<td>73</td>
<td><strong>7.35</strong></td>
<td>75</td>
<td>10.94</td>
<td>78</td>
<td>14.71</td>
<td>95</td>
<td>39.71</td>
</tr>
<tr>
<td>Tunnel</td>
<td>4000</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>71</td>
<td>373.33</td>
</tr>
</tbody>
</table>
the cameras are oriented to vehicle tails, Refs. 9 and 16 become invalid. In another video “Foggy”, where some vehicles turn on the headlights while others do not, Refs. 9 and 16 cannot work well. By contrast, the proposed approach is seldom restricted by environmental condition or camera orientation, and thus is more universal and can be applied to a wider range of scenarios. (4) The approach in Ref. 37 is able to detect and count vehicles 24 hours a day. But its performance depends heavily on the image contrast. In sunny or cloudy condition where the image contrast is high, it can acquire good performance. However, when image contrast is relatively low, such as in testing videos “Foggy” and “Blizzard”, a large number of noisy points will be segmented out from the scene as pixels of interest by the Otsu binarization method adopted in Ref. 37, causing vehicle counting to fail. By contrast, our approach performs well in all the testing environments. (5) Four approaches all suffer from two difficult cases: the case where two vehicles occupy one virtual loop and undercounting occurs, and the case where one vehicle occupies two virtual loops and overcounting occurs. Most of the counting errors of our approach result from these two cases, as shown in Fig. 19. This reflects inherent limitations of the virtual loop method.

6.4. Computational time

When the incremental semi-supervised learning approach runs, three computational steps consume the most operations: calculating the classifier output via (1), updating the importance of existing rules via (9), and updating the MF parameters via (7) and (8). The time complexities of three steps are all $O(n_1 n_3)$, where $n_1$ and $n_3$ are the numbers of neurons in the input layer and rule layer of the neuro-fuzzy network. Hence, the overall time complexity is $O(n_1 n_3)$. By contrast, the time complexity of support vector machine at the classification stage is $O(n_1 n_s)$, where $n_s = 995$ is the number of support vectors. Meanwhile, the time complexity of random forest at the classification stage is $O(n_1 n_t)$, where $n_t = 500$ is the number of trees. Three approaches all have linear time complexity. In our approach, since $n_1 = 14$ and $n_3$ is in the thousands, the time complexity is not high.

The proposed approach is implemented in MATLAB on a PC with 2.50 GHz Intel Core i5-3210M CPU and 4 G memory. The computational time is monitored by the “tic” and “toc” functions in MATLAB. Taking the video “Highway I” (with two virtual loops) for example, the average time for processing one frame is 1.338 s, of which 1.211 s is spent in feature extraction, and 0.127 s is spent in sample classification and classifier optimization. It should be noted that when there are many for-loop structures in the codes, the MATLAB implementation is much slower than C++ or C# implementation. In our program, the for-loop structures are used many times for feature extraction, sample classification, and classifier optimization, so that the implementation is not very efficient. As such, the proposed approach is computationally feasible.
7. Conclusion

In this paper, we propose a video-based vehicle detection approach with data-driven adaptive neuro-fuzzy networks. The motivation of this work is to improve the virtual loop method and to make it work better in complex environments. Our approach consists of two interrelated stages: training and classification. At both the stages, 14 image features (regarding foreground area, texture change, and environmental condition) are extracted to represent the distinction between vehicle and nonvehicle patterns. At the training stage, three neuro-fuzzy networks are trained via supervised learning to build a combined classifier. At the classification stage, the combined classifier labels newly emerged samples into vehicle and nonvehicle. In order to make the proposed approach adaptable to complex environments, the sample classification results are used backward to retrain the classifier in an incremental semi-supervised learning manner. The structures and parameters of the neuro-fuzzy networks are

Fig. 18. In “Dtneu_schnee”, fluttering snowflakes are detected falsely as vehicle by the approach in Ref. 16.

Fig. 19. Two difficult cases that cause counting errors. (a) Two vehicles occupy one virtual loop. (b) One vehicle occupies two virtual loops.
automatically optimized as the system runs. Comparative experiments are conducted on a variety of testing videos, and the results show that the proposed approach is accurate and robust to detect vehicles in complex environments, and outperforms some state-of-the-art approaches.

Acknowledgments

The authors would like to thank Professor Fei-Yue Wang at the Institute of Automation, Chinese Academy of Sciences, for his instruction and encouragement to this work. This work was supported in part by the National Natural Science Foundation of China under Grant 61304200.

References

K. Wang & Y. Yao


Kunfeng Wang received his Bachelor degree in Materials Science and Engineering from Beihang University, Beijing, in 2003 and his Ph.D. in Control Theory and Control Engineering from the Graduate University of Chinese Academy of Sciences, Beijing, in 2008. Currently, he is an Assistant Professor in the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include intelligent video analytics, intelligent transportation systems, and machine learning.

Yanjie Yao received her Bachelor degree in Electrical Engineering and Automation from China University of Geosciences, Beijing, China, in 2011. She is currently a Ph.D. candidate at the Institute of Automation, Chinese Academy of Sciences. Her research interests include computer vision and deep learning.