Abstract—On-road vehicle detection is a critical operation in automotive active safety systems such as collision avoidance, merge assist, lane change assistance etc. In this paper, we present VeDAS - Vehicle Detection using Active learning and Symmetry. VeDAS is a multi-part based vehicle detection algorithm that employs Haar-like features and Adaboost classifiers for the detection of fully and partially visible rear-views of vehicles. In order to train the classifiers, a modified active learning framework is proposed that selects positive and negative samples of multiple parts in an automated manner. Furthermore, the detected parts from the classifiers are associated by using a novel iterative window search algorithm and a symmetry-based regression model to extract fully visible vehicles. The proposed method is evaluated on seven different datasets that capture varying road, traffic and weather conditions. Detailed evaluations show that the proposed method gives high true positive rates of over 95%, and performs better than existing state-of-the-art rear-view based vehicle detection methods. Additionally, VeDAS also detects partially visible rear-views of vehicles using the residues left behind after detecting the fully visible vehicles. VeDAS is able to detect partial rear-views with a detection rate of 87% on a new partially visible rear-view vehicle dataset that we release as part of this paper.

Index Terms—vehicle detection, active safety, driver assistance, intelligent vehicles

I. INTRODUCTION

Rear-end crashes that constitute more than 29% of all crashes [1], [2] occur between the ego-vehicle and the leading vehicles in front of the ego-vehicle. Detection of on-road vehicles in front of the ego-vehicle can eliminate such rear-end crashes substantially [1]. Although vehicle detection using active sensors such as radars is now available in commercial vehicles, vision-based sensing of vehicles is increasingly being considered due to limitations of active sensors in detecting multiple obstacles [3].

Detecting vehicles using on-board monocular cameras is still considered as a challenging research problem [3] because it requires robustly detecting the vehicles amidst a variety of moving and stationary features in the image scene such as shadows, buildings, railings etc. A number of vehicle detection techniques using monocular cameras have been proposed, especially for the case of on-road vehicle detection [4]–[8]. On-road vehicle detection refers to detection systems where the camera is mounted on the ego-vehicle rather than being fixed on infrastructure such as in traffic/driveway monitoring systems [9].

[9] and the more recent survey in [3] provide a list of recent literature that detect on-road vehicles using monocular cameras with different levels of accuracy and efficiency. They can be broadly divided based on whether they use motion or appearance. Appearance based techniques employ features such as histogram of oriented gradients (HOG) [8], SURF [20], Haar-like features [6] etc. The features are then classified using classifiers such as support vector machines (SVMs) [8], Adaboost [6] etc. Furthermore, contextual information such as lanes have been used to improve accuracy and robustness of vehicle tracking [21]. Additionally, robustness is improved by using new learning methods such as active learning [6]. Similarly, motion-based techniques such as [22], [23] use motion cues to detect moving vehicles. This paper proposes an appearance based method and rest of the paper focuses on such works.

Although a significant pool of literature on vision-based on-road vehicle detection can be found, there still exist many open challenges in detecting on-road vehicles robustly using monocular cameras [3]. Firstly detecting vehicles with low false detection rate and acceptable levels of true detection rate is still a challenging task. Second, most existing methods such as [4], [6], [10] detect fully visible vehicles, i.e. the entire view of the front or rear of the vehicle is visible in the input image frame, whereas detection of partially visible on-road vehicles has been explored only recently in [5], [8], [16], [23].

In this paper, we propose a novel multi-part vehicle detection technique to detect vehicles, whose rear-views are fully and partially visible. The proposed technique involves two stages. In the first stage, two parts of the rear-view of a vehicle are detected using Haar-like features and Adaboost cascade classifiers. In order to train the classifiers, we propose an extension to the active learning framework proposed in [6] for vehicle detection. The second stage of the proposed technique involves an iterative window search algorithm that combines the two parts in an iterative manner to detect fully visible vehicles using symmetry regression models. The proposed method is further extended to detect partially visible vehicles by taking advantage of the detection of multiple parts. The proposed method is henceforth referred to as VeDAS, which is an acronym for Vehicle Detection using Active learning and Symmetry. It is to be noted that VeDAS falls in the category of appearance based vehicle detection methods such as [6], [13], [18], which are particularly designed and optimized for detecting rear-views of the vehicles as seen from on-board cameras.

The rest of the paper is organized as follows. In Section II, we survey recent on-road vehicle detection methods and present the scope of the proposed study. In Section III, we introduce the proposed active learning framework. VeDAS for detecting fully visible vehicles is described in detail in Section IV. VeDAS is extended further for the detection of partially visible vehicles in Section V. Detailed evaluations and comparisons are presented in Section VI followed by

Multi-part Vehicle Detection using Symmetry Derived Analysis and Active Learning

Ravi Kumar Satzoda, Member, IEEE and Mohan M. Trivedi, Fellow, IEEE
concluding remarks in Section VII.

II. RELATED WORK & SCOPE

Vehicle detection is primarily an object detection task in computer vision. Although there are a number of different combinations of features and classifiers that are proposed for object detection [24], [25], we limit the scope of this survey to methods that are specifically described for vehicle detection. Also, on-road vehicle detection itself has been explored using different combinations of features and classifiers [3], that include both appearance and motion based methods, and monocular [4], [6], [16], [18] and 3D stereo vision based techniques [26]. In this section, we briefly survey existing recent works such as [2], [4]–[8], [11], [15], [16], [18], [19] on vehicle detection using monocular cameras, that are particularly focused on detection of on-road vehicles using the rear-view of the vehicle which is either fully or partially visible.

Table I lists recent appearance based vehicle detection techniques. For each technique, the associated features and classifiers are listed. The detection type column lists whether or not a particular method can detect occluded vehicles. The evaluation datasets are also listed in Table I. It can be seen from Table I that most existing methods such as [4], [6], [7], [10], [18] are designed for detecting rear-views of vehicles that are not occluded. In [10] a detailed evaluation and comparison of different kinds of features and classifiers for vehicle detection is presented. Haar-like features with cascaded classifiers such as AdaBoost have been used in multiple works such as [5], [7], [15] etc. for vehicle detection resulting in varying degrees of accuracy. Variants of Haar-Cascade classifiers such as WaldBoost detection in [13] and adaptive global Haar classifier in [2] are also explored. Similarly, HOG and SVM are also used to detect vehicles in [8], [14]. Other kinds of features that are used for vehicle detection include edges [12], edge oriented histograms [11], [18], under-vehicle shadows [12] etc. Neural networks and heterogeneous classifiers are used in [18] to detect vehicles to improve detection rates. Symmetry of fully visible vehicles is another feature that has been explored in multiple ways in [11], [12], [14], [19]. In order to improve the speed of detection and also the accuracy of detection in monocular vision based object detection, techniques have been proposed to reduce the numbers of sliding windows. While [28], [29] discuss such techniques for generic object detection, [18], [30] propose methods that use the perspectives in on-road vehicle detection scenario to reduce computations.

Similarly, part-based methods have also been proposed to detect both fully visible vehicles and occluded vehicles. Although Wang et al. propose a part-based method in [4] using principle component analysis (PCA), the method is not demonstrated for occluded vehicles. While [8] uses discriminative part-based model (DPM) to detect vehicles from different orientations, it is not clear if the same method can be used to detect occluded vehicles. Sivaraman et al. [5] have proposed a part-based method to independently detect the front and rear parts of a vehicle, which is applicable for detecting vehicles that are entering or exiting intersections. Chen et al. [15] have also described similar classifiers and features as [5] to detect fully visible as well as occluded vehicles as seen from the onboard camera. However, this method does not independently detect the parts of occluded vehicles. Instead, it uses Chamfer distance to estimate the presence of two vehicles in a bounding

<table>
<thead>
<tr>
<th>Title</th>
<th>Features &amp; Classifiers</th>
<th>Detection type</th>
<th>Dataset &amp; Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun et al. 2006 [10]</td>
<td>PCA, Gabor wavelets, SVM, Neural networks</td>
<td>Full vehicles</td>
<td>Proprietary dataset, 1051 true and 1051 false subimages, 3-fold cross validation</td>
</tr>
<tr>
<td>Chang et al. 2010 [7]</td>
<td>Haar-like features with online boosting</td>
<td>Full vehicles</td>
<td>Proprietary dataset</td>
</tr>
<tr>
<td>Sivaraman et al. 2013 [5]</td>
<td>Haar-like features with AdaBoost, Detection by parts</td>
<td>Vehicles entering or exiting intersections</td>
<td>Public LISA datasets</td>
</tr>
<tr>
<td>This work - VeDAS</td>
<td>Multi-part based, active learning, Haar-like features, AdaBoost classifiers, iterative window search with symmetry derived analysis</td>
<td>Full and partially visible rear-views of vehicles</td>
<td>LISA dataset, iROADS dataset [2], TME motorway dataset [13], New partially visible rear-view vehicle dataset</td>
</tr>
</tbody>
</table>
One of the recent works on occluded vehicle detection is proposed in [16], in which the occlusion patterns are classified as single or double occlusions using a modified discriminative part-based model, that has been evaluated using KITTI dataset [17].

A. Scope of Proposed Work

Vehicle detection is a widely discussed topic resulting in varying types of research problems that are being answered. Therefore, we define the scope of the proposed VeDAS explicitly before presenting the contributions of this paper. In this paper, a multi-part based method is proposed that is particularly catered towards detecting the rear-views of the vehicles from on-board monocular cameras. This work falls in the category of works such as [2], [6], [13], and does not address detection of vehicles in multiple orientations and varying levels of occlusion such as [17], [37], [38]. Therefore, the evaluations are also performed on highway/freeway datasets such as those presented in [2], [6], [13]. For the same reason, KITTI vehicle dataset [17] is not used for evaluation. Also, with regards to detecting occluded vehicles in the context of this work, we define occlusion in this paper as partial visibility of the rear-views of the vehicles (unlike generic occlusion as defined in KITTI vehicle dataset [17]). Fig. 1 shows some sample rear-views of vehicles that form the scope of this work. More details about the amount of visibility of the rear-views of the vehicles will be elaborated in forthcoming sections.

Fig. 1. Sample rear-views of vehicles that will be detected by VeDAS: fully visible and partially visible rear views of vehicles shown by green solid rectangles and red dotted rectangles respectively.

III. Active Learning for Training Multi-part Vehicle Classifiers in VeDAS

The first stage of VeDAS involves multi-part detection using Haar-Adaboost cascade classifiers. In order to train the cascade classifiers, an extension to active learning method [6] is proposed in this paper.

A. Active Learning

Active learning is defined as a form of learning in which the learning program has some control over the inputs on which it trains [31]. It is particularly aimed at selective input querying for training classifiers [32], [33]. Active learning based training sample selection has been shown to be more discriminative than random sampling for training classifiers [6], [33], [34]. It is based on the idea that for a given set of $m$ training samples $S^m$, there exists a region of uncertainty $R(S^m)$ in which the trained classifier on $S^m$ classifies ambiguously. Therefore, the classifier is again trained with samples that are more likely to be classified incorrectly in order to improve its performance. Active learning involves two steps. In the first step, which is called passive learning, manually annotated positive sample and random negative samples are used to train the classifiers. In the second step, true and false positive samples are manually queried from the outputs generated by applying the passively trained classifiers on test images. This entire semi-supervised learning mechanism is called active learning framework [31], [33]. In [6], active learning was first proposed for vehicle detection. This framework is primarily proposed to reduce the false detection rates for vehicles, while maintaining high detection rates [6].

Fig. 2. Proposed active learning method for training of the two parts using AdaBoost classifiers and Haar-like features.

B. Proposed Extension to Active Learning Framework in VeDAS

The first stage of VeDAS is to extract possible candidates for vehicle parts. In order to do this, Haar-like features with AdaBoost cascade classifiers are used. Fig. 2 shows the proposed two-step active learning framework. The first round involves training two AdaBoost classifiers that classify the Haar-like features of the two parts $V^P_1$ and $V^P_2$ of the rear-view of a vehicle. The positive samples for the two parts are shown for two different vehicles with two different viewing angles as seen from an on-board camera from the ego-vehicle.
in Fig. 3(a). The negative samples are randomly generated from images that do not have any vehicles (positive samples) in them. This training yields two classifiers $C^1_1$ and $C^1_2$ corresponding to the two parts $V^{P1}$ and $V^{P2}$, where subscript 1 denotes the first step classifiers. This first step of learning, which is called the passive learning step [6], [33] is shown in the top block in Fig. 2.

In the second step of learning (shown in the bottom block in Fig. 2), we again train the AdaBoost cascades but with more informed and selective positive and negative samples. In order to do this, a multi-scale sliding window approach is applied on the training image set from the first step to generate Haar-like features, which are then classified using the classifiers from first step, i.e. $C^1_1$ and $C^1_2$. It should be noted that the entire training image, and not just the previously annotated window is sent for classification in Learning Step 2. This classification results in two sets of windows $W^{P1}_1$ and $W^{P2}_1$ for the two parts $V^{P1}$ and $V^{P2}$ respectively. These windows are then filtered as true positives and false positives using the classification results in two sets of windows $W$. It should be noted that the entire training image, and not just the previously annotated window is sent for classification in Learning Step 2. This set of positive samples is then utilized and selected using the method described in the paper. Also, the classifiers from the first step $C^1_1$ and $C^1_2$ are 20 stage classifiers that are trained using 10000 positive training samples. These positive training samples include samples from rear-views of vehicles with different viewing angles with respect to the ego-vehicle as shown in Fig. 3(a).

Haar-like features are then extracted from the positive and negative windows that are selected using the above method. The Haar-like features are then used to train two AdaBoost cascade classifiers $C^2_1$ and $C^2_2$. These classifiers are used in the first stage of VeDAS to generate hypothesis windows for the two parts $V^{P1}$ and $V^{P2}$, which will be further processed in the second stage of VeDAS.

The following objectives are met in the proposed active learning framework. Firstly, the second round of learning is automated, and unlike [6] there is no manual supervision to select positive and negative windows for the second step of learning. Second, a more informed set of negative samples is generated for training the classifiers. This is similar to the framework in [6] but the annotation of the negative samples is performed in an automated manner in this paper. Finally, a more informed set of positive samples is generated for training the classifiers in the second round. This set of positive samples accounts for error in the selection of positive samples for training the classifiers.

### IV. DETECTING FULLY VISIBLE VEHICLES USING VE-DAS

In this section, we will show how VeDAS is used to detect fully visible vehicles. VeDAS involves two stages. The first stage uses the classifiers $C^1_1$ and $C^1_2$ to generate two sets of hypothesis windows $B^{P1}$ and $B^{P2}$ for the two parts $V^{P1}$ and $V^{P2}$ respectively. It is ensured that if there is a fully visible vehicle in the image frame, then both parts are always detected by setting a low threshold on the cascade classifiers. The proposed active learning framework enables elimination of a large number of false positive windows. The two sets of detection windows are denoted by:

$$B^{P1} = \{B^{P1}_{1}, B^{P1}_{2}, \ldots, B^{P1}_{N_{P1}}\} \quad (1)$$

$$B^{P2} = \{B^{P2}_{1}, B^{P2}_{2}, \ldots, B^{P2}_{N_{P2}}\} \quad (2)$$

where $B^{Pj}_{i}$ represents $i$-th window for $j$-th part, $N_{P1}$ and $N_{P2}$ denote the total number of windows that are detected for the two parts respectively by the classifiers. A window $B^{Pj}_{i}$ is characterized by the usual notation of a bounding box, i.e. $B^{Pj}_{i} = [x_i, y_i, w_i, h_i]$ where $(x_i, y_i)$ denotes the top left corner of the bounding box with respect to (w.r.t) the top left corner of the input image, and the bounding box has a width $w_i$ and height $h_i$ respectively. For the rest of the paper, the horizontal axis is denoted by $x$ and vertical axis is denoted by $y$.

Given the windows $B^{P1}$ and $B^{P2}$ for the two parts $V^{P1}$ and $V^{P2}$, the next step is to localize the fully visible vehicle by selecting the most appropriate combination of $V^{P1}$ and $V^{P2}$ windows. This forms the second stage of the proposed method, wherein the resulting windows that are obtained by applying the classifiers $C^2_1$ and $C^2_2$ are analyzed in an iterative manner in order to improve true detections and reduce false positive rate. It is to be noted that the windows in $B^{P1}$ and $B^{P2}$ are not subjected to any non-maximal suppression or merging of windows. The following steps will select the most appropriate combination of windows in $B^{P1}$ and $B^{P2}$ to detect fully and partially visible vehicles.
1) Step 1: Window Filtering using Camera Calibration:

Given that VeDAS is catered for on-board vehicle detection from video sequences, it is assumed that camera calibration can be obtained beforehand as a one-time operation. In Step 1 of VeDAS, inverse perspective mapping (IPM) [35] is used to compute a look-up table (LUT) with possible window sizes for specific camera calibration information. In this study and evaluation, we assume that the ground plane is flat. A homography matrix $H$ is generated using camera calibration parameters to convert camera coordinate system (CCS) or image domain into world coordinate system (WCS) or top view. Fig. 4 illustrates the LUT generation process. Given an input image $I$, the IPM image $I_w$ is generated using the homography matrix $H$ [36]. Therefore, every point $P(x,y)$ in $I$ is transformed to $P_w$ in $I_w$ using $H$, i.e.,

$$
[x_w \ y_w \ 1]^T = kH [x \ y \ 1]^T
$$

where $k$ is the calibration constant. Therefore, the four points $P_1$ to $P_4$ in the IPM domain correspond to the minimas and maximas in the IPM domain along the $x_w$-$y_w$ coordinates. For each row in $I_w$, point $P_w$ such that $P_{3w} - P_{1w} = w_w^{W}$ is determined, where $w_w^{W}$ is the width of the vehicle as seen from top view. Considering most consumer vehicles usually have a standard axle length, $w_w^{W}$ can be pre-determined. Given $P_{1w}$ and $P_{3w}$, we use the inverse of $H$, i.e. $H^{-1}$ to determine the corresponding points $P_1$ and $P_3$ in the image domain. For each row index $y$ in $I$, the LUT has the following list of values:

$$
w_w(y) = x_3 - x_1
$$

where $w_w(y)$ is the width of the window that should be used for vehicle detection in the $y$-th row of $I$.

Given the sets of windows $B^{P1}_{W}$ and $B^{P2}_{W}$ for parts $V^{P1}$ and $V^{P2}$ of the vehicle, we find $\delta_w$ for every $B^{Pj}_{i}$-th window where

$$
\delta_w = |w_w(y) - 2w^{Pj}_{i}(y^{Pj}_{i} + h^{Pj}_{i})|
$$

where $w^{Pj}_{i}$ is the width of the window $B^{Pj}_{i}$ whose bottom left corner is at $y = y^{Pj}_{i} + h^{Pj}_{i}$. If $\delta_w < 10$ we consider the window for further processing. This condition on $\delta_w$ is stable because it is based on the possible width of the vehicle in the IPM domain, which is constant for most vehicles and irrespective of the distance of the vehicle from the ego-vehicle. Therefore, this threshold is not location-dependent or case dependent.

2) Step 2: Iterative Window Search using Symmetry Derived Analysis:

The next step is to look for the best combination of the parts $V^{P1}$ and $V^{P2}$ that make a full vehicle in the filtered set of windows in $B^{P1}_{W}$ and $B^{P2}_{W}$. In order to do this, we generate what are called as window voting maps for each part $V^{P1}$ and $V^{P2}$ in $B^{P1}_{W}$ and $B^{P2}_{W}$ (these sets now contain the windows that are filtered using Step 1). For an $m \times n$ (columns versus rows) sized input image $I$, voting maps are denoted by $M^{P1}_{W}$ and $M^{P2}_{W}$, where

$$
M^{P1}_{W}(x,y) = \ n \ \text{if} \ \{(x,y) | (x,y) \in n \ \text{windows in } B^{P1}_{W}\} \quad (6)
$$

$$
M^{P2}_{W}(x,y) = \ n \ \text{if} \ \{(x,y) | (x,y) \in n \ \text{windows in } B^{P2}_{W}\} \quad (7)
$$

In other words, $M^{P1}_{W}(x,y)$ is set to $n$ if the coordinate $(x,y)$ is a member of $n$ windows in $B^{P1}_{W}$. Fig. 5 shows the 2D and 3D versions of the voting maps for the hypothesis windows that are detected for a sample input image frame. A higher value at a location $(x,y)$ in the voting map in Fig. 5(b) indicates that there are more overlapping windows at that location.
dows in $M^1$ voting map that must be combined with windows in $M^2$ to detect fully visible vehicle. Algorithm 1 lists the main steps of the algorithm.

Algorithm 1 Iterative Window Search Algorithm

1. Input: $M^1$, $M^2$, $B^1$, $B^2$
2. Output: Full vehicle bounding boxes $B^F = \{B^F_i\}$
3. for $T^1_i = T_{\text{min}}$ to $T_{\text{max}}$ do
4. \hspace{0.5cm} Binarize $M^1_p$: $M^1_p = M^1_i > T^1_i$
5. \hspace{0.5cm} Get set of bounding boxes $B^1_B = \{B^1_B\}$ for each blob in $M^1_p$
6. \hspace{0.5cm} for $T^2_i = T_{\text{min}}$ to $T_{\text{max}}$ do
7. \hspace{1cm} Binarize $M^2_p$: $M^2_p = M^2_i > T^2_i$
8. \hspace{1cm} Get set of bounding boxes $B^2_B = \{B^2_B\}$ for each blob in $M^2_p$
9. \hspace{1cm} For each $B^1_B \in B^1_B$, find nearest similar sized $B^2_B \in B^2_B$.
10. \hspace{1cm} for each blob pair found do
11. \hspace{1.5cm} Find window pairs $(B^1_B, B^2_B)$ that satisfy Symmetric condition
12. \hspace{1.5cm} if Symmetric Condition is true then
13. \hspace{2cm} Select pair and draw full vehicle bounding box $B^F_i$
14. \hspace{1.5cm} Remove blob pair from $B^1_B$ and $B^2_B$
15. \hspace{1cm} end if
16. \hspace{1cm} end for
17. \hspace{1cm} Break if no blobs in $B^2_B$
18. \hspace{1cm} end for
19. \hspace{1cm} Break if no blobs in $B^1_B$
20. end for

As shown in Algorithm 1, each voting map is thresholded with $T^1_i$ and $T^2_i$ from $T_{\text{min}}$ to $T_{\text{max}}$, i.e., from a lower vote count to higher vote count. For every setting of the threshold, each voting map generates a set of blobs $B^1_B$ and $B^2_B$, where $B^1_B$ and $B^2_B$ denote the sets of bounding boxes that encapsulate the blobs in the thresholded voting maps $M^1_p$ and $M^2_p$. Given a blob $B^i_B = [x^i_B, y^i_B, w^i_B, h^i_B] \in B^i_B$, where $i = 1, 2$ and the left part of the vehicle, $B^i_B = [x^i_B + w^i_B, y^i_B, w^i_B, h^i_B]$ is first computed. Then for $B^1_B \in B^1_B$ and $B^2_B \in B^2_B$, the following expression is computed:

$$\delta_p = \frac{\text{area}(B^1_B \cap B^2_B)}{\text{area}(B^1_B \cup B^2_B)} \quad (8)$$

The above expression captures how closely are the left and right blobs related in terms of shape and position. In other words, the left blob and right blob must be of similar size and placed adjacently. In our experiments involving multiple real-world datasets, $\delta_p$ was set to 0.6. If such pairs of blobs are found, vehicle part window pairs $(B^1_B, B^2_B)$ are selected such that $B^1_B$ and $B^2_B$ lie within the left and right blob respectively, and satisfy the following conditions:

1) High overlap with the blocks.
2) Symmetry condition is met.

The first condition above ensures that the vehicle part windows $B^1_B$ and $B^2_B$ are indeed forming majority of the selected blobs in the voting maps. The second condition is what we call as symmetry condition. Most on-road fully visible vehicles are symmetric when viewed from the rear side of the vehicle. We use this property to determine if $B^1_B$ matches with $B^2_B$ in appearance.

Considering the $V^1$ and $V^2$ correspond to the left and right halves of the vehicle, $B^1_B$ and $B^2_B$ are symmetric w.r.t the central vertical axis. In order to check this symmetry, given $B^1_B$ and $B^2_B$, a bounding box $B^F = [x^F, y^F, w^F, h^F]$ is generated such that it comprises both $B^1_B$ and $B^2_B$ in it. This is illustrated using a sample image in Fig. 6. The image patch $I_F$ corresponding to bounding box $B^F$ is extracted from image $I$, which is then divided into two equal parts along the vertical axis resulting in $I^1_F$ and $I^2_F$. $I^1_F$ is flipped and both $I^1_F$ and $I^2_F$ are divided into $8 \times 8$ blocks (the incomplete blocks are padded). For each $8 \times 8$ block at location $(p, q)$ (as shown in Fig. 6) in flipped $I^1_F$ and $I^2_F$, scaled histograms of gradient angles $h^1_F$ and $h^2_F$ are generated. The scaling is performed using gradient magnitudes at each pixel coordinate, whose gradient angle is used to generate the histogram of gradient angles. $h^1_F$ and $h^2_F$ are normalized and a dot product is taken to determine the symmetry score $S_{b(p,q)}$ for that pair of symmetrically opposite blocks in $I^1_F$ and $I^2_F$, i.e.

$$S_{b(p,q)} = \frac{|h^1_F|}{||h^1_F|| + \epsilon} \cdot \frac{|h^2_F|}{||h^2_F|| + \epsilon} \quad (9)$$

$S_{b(p,q)}$ is the dot product between the two normalized histogram vectors of the symmetrically opposite $(p, q)$-th blocks in $I^1_F$ and $I^2_F$. Such scores are generated for all the symmetrically opposite blocks, and the total score for the bounding box $B^F$ that encloses the chosen windows $B^1_B$ and $B^2_B$ is computed in the following way:

$$S_{IF} = \sum_{all \; blocks} S_{b(p,q)} \quad (10)$$

The symmetry score $S_{IF}$ for the bounding box $B^F$ is used to train two linear-regression models. Fig. 7 shows the quadratic relationship between the symmetry score $S_{IF}$ and the width and the height of the bounding box $B^F$. These plots are generated using 10000 manually annotated positive training windows that were previously used for training the classifiers in the first stage of VeDAS. Therefore, given width $w$ and $h$ of the bounding box $B^F$, we have,

$$S_w = \alpha_0 + \alpha_1 w + \alpha_2 w^2 \quad \text{and} \quad S_h = \beta_0 + \beta_1 h + \beta_2 h^2 \quad (11)$$

where $\alpha_i$ and $\beta_i$ are coefficients of the linear regression models that are learned from training data, and $S_w$ and $S_h$ are the predicted symmetry scores from the linear regression models.

Fig. 6. Illustration for computing symmetry score $S_{IF}$. During testing, given $I_F$, the symmetry score is computed as $S_{IF}$. The width and height of $I_F$ are then used to determine the predicted symmetry scores $S_w$ and $S_h$ using the learned regression models. If $S_{IF}$ lies withing 25 symmetry score units of both $S_w$ and $S_h$, then $I_F$ and the bounding box $B^F$ is considered to have met the symmetry condition. The bounding
The two parts box $B$ box 10000 training samples indicate quadratic relationships. $M$ in the next section. used for detecting the partial views of the vehicles as described are no more windows to process or there are residues which are maximum of 7 iterations before hitting the case when there next iteration. In our experiments we found that we need a is not performed on the same set of windows again, after within the blobs are eliminated. This ensures that the search is found between the left and right blobs, all the windows is checked for symmetry. Additionally, after a match should be identical in shape, and then the appearance within the blobs is checked for symmetry. Therefore, after reaching the termination condition and detecting all possible fully visible vehicles, the voting maps will have residues as shown Fig. 8 (b) and (c) corresponding to the left and right parts respectively. The voting maps with the residues are denoted as $M_{P1r}$ and $M_{P2r}$.

Although the proposed iterative method searches for window assignments, this is not entirely a greedy search algorithms. Optimality is ensured because the greedy search is not performed on a one-on-one basis between every single window from the left vehicle part classifier and the windows from the right vehicle part classifier. Instead, the search is performed between the blobs that are extracted from the left and right voting maps after thresholding, where each blob indicates the presence of sufficient number of windows for the vehicle part. In order to ensure that the left and right blobs are accurately matched, we use the symmetry score condition in addition to the blob shape matching, i.e. first the left and right blobs should be identical in shape, and then the appearance within the blobs is checked for symmetry. Additionally, after a match is found between the left and right blobs, all the windows within the blobs are eliminated. This ensures that the search is not performed on the same set of windows again, after changing the threshold for the generating the blobs in the next iteration. In our experiments we found that we need a maximum of 7 iterations before hitting the case when there are no more windows to process or there are residues which are used for detecting the partial views of the vehicles as described in the next section.

V. DETECTING PARTIALLY VISIBLE REAR-VIEWS OF VEHICLES USING VEDAS

In the previous section we explained the method to combine the two parts $V^{P1}$ and $V^{P2}$ to detect fully visible vehicles from rear view as shown in Fig. 8(a). After detecting each detection window, the corresponding regions in the voting maps $M_{P1}$ and $M_{P2}$ are eliminated. Therefore, after reaching the termination condition and detecting all possible fully visible vehicles, the voting maps will have residues as shown Fig. 8 (b) and (c) corresponding to the left and right parts respectively. The voting maps with the residues are denoted as $M_{P1r}$ and $M_{P2r}$.

![Figure 7](image1.jpg) Fig. 7. Symmetry score versus width and height of the bounding boxes from 10000 training samples indicate quadratic relationships.

![Figure 8](image2.jpg) Fig. 8. Using the voting maps to detect partially visible vehicles: (a) Detection windows of fully visible vehicles obtained from the proposed method in Section IV, (b) Residue of the left voting map after removing detection window regions, (c) Residue of the right voting map after removing detection window regions, (d) Detection windows of partially visible vehicles using the residues of the left and right voting maps. Red window indicates vehicles that are partially visible on the right, and green window indicates vehicles that are partially visible on the left.

We will explain the proposed method using Algorithm 2 to detect vehicles whose left parts, i.e. $V^{P1}$, are partially visible. The same is extended for the vehicles with their right parts partially visible. For sake of explanation, the vehicles with left parts visible are henceforth called right occluded vehicles, and the vehicles with right parts visible are called left occluded vehicles. Given $M_{P1}$ with the residues in the left parts voting map as shown in Fig. 8(b), and the set of windows $B_{P1}$ for left part of the vehicles, we use the following iterative window search to determine the right occluded vehicles.

As described in the previous section, the voting map $M_{P1}$, is thresholded using a threshold $T_0$ to get binarized voting map $B_{P1}$. The blobs in $B_{P1}$ are used to get a set of bounding boxes denoted by $B_{P1}$. For each bounding box $B_{P1}$, we search the windows in $B_{P1}$ to get the $i$-th window $B_{P1}$ to satisfy the same condition as in (8), i.e. $\delta_r = \frac{area(B_{P1} \cap B_{P1})}{area(B_{P1} \cup B_{P1})}$. In other words, the window $B_{P1}$ should be of similar size and at similar position as the blob $B_{P1}$. In case of multiple windows that satisfy the above condition, the window with the maximum $\delta_r$ is considered, i.e. arg max$(\delta_r)$. The same method is repeated for the residues in right window voting map $M_{P2}$ to get the left occluded vehicles. Algorithm 2 lists the steps of the algorithm for detecting the left and right occluded vehicles. The effectiveness of the proposed algorithm
is evaluated briefly in Section VI-C1.

Although it appears that the above method is a mere blob detection, the detection of partially visible vehicles is made possible because of the multi-part nature of VeDAS. The residues that remain after the detection of fully visible vehicles are probed further in VeDAS because the proposed multi-part classifiers could have detected partially visible rear-views of the vehicles. Therefore, VeDAS ensures that it checks these residues to find any partially visible rear-views.

**Algorithm 2 Iterative Window Search Algorithm to Detect Partially Visible Vehicles**

1: Input: Voting maps with residues: \( M^{pr}, M^{pr2} \), and windows detected from HG step: \( W^{pr}, W^{pr2} \)
2: Output: Partially visible vehicle left and right bounding boxes \( \{B^{pr}_1\} \) and \( \{B^{pr2}_1\} \)
3: for \( T_{i} = T_{max} \) to \( T_{min} \) do
4: Binarize \( M^{pr}_i, M^{pr2}_i \) = \( M^{pr2}_i \) > \( T_{i} \)
5: Get set of bounding boxes \( B^{pr}_i = \{B^{pr}_i\} \) for each blob in \( M^{pr}_i \)
6: For each \( B_{i} \in M^{pr}_i \), find \( B_{ij} \in B^{pr}_i \) that best maximizes \( \delta_{i} \)
7: if \( B_{ij} \) is found then
8: Update left partial visible vehicles set with \( B_{ij} \)
9: Remove blob region from \( M^{pr}_i \) that corresponds to \( B_{ij} \)
10: end if
11: Exit loop if \( M^{pr}_i \) does not have any more blobs
12: end for
13: Repeat above steps for right partial visible vehicles

**VI. PERFORMANCE EVALUATION**

**A. Datasets and Metrics for Evaluation**

Publicly available datasets are used to evaluate and compare the performance of the proposed vehicle detection method. Table II lists the properties of the different datasets. LISA-Dense dataset [6] is particularly complex involving dense traffic, multiple vehicles, varying lighting conditions and vehicle movements. LISA-Sunny dataset is less dense than LISA-Dense but it has additional features such as overhead bridges, signposts etc. which usually give false detections in most conventional vehicle detection methods [6]. The third dataset is taken from TME Motorway dataset [13]. The TME Motorway dataset involves a large set of monochrome video sequences that are annotated using laser scanner. We have considered one section of the TME motorway dataset involving 2300 frames. The next four datasets are taken from iROADS datasets [2], which capture varying road and weather conditions such as rainy day, tunnel roads etc.

We evaluate and compare the proposed method using the four metrics: (a) true positive rate \( (TPR = TP/(TP + FN)) \), (b) false detection rate \( (FDR = FP/(FP + TP)) \), (c) average TP per frame, and (d) average FP per frame. \( TP, FP \) and \( FN \) refer to true positives, false positives and false negatives respectively. The four metrics are similar to the metrics described in [6]. A detected window is considered TP, if there is an overlap of at least 50% with the ground truth window. It is to be noted that TPR and FDR are also representative of more conventional performance metrics such as precision and recall. This is because TPR and FDR are mathematically related to recall and precision in the following way: \( TPR = recall \) and \( FDR = 1 - precision \).

**B. Results & Discussion**

1) **Evaluation of Active Learning Framework**: Before evaluating and comparing the VeDAS against other methods and on different datasets, we first evaluate the performance of the proposed active learning framework. An \( n \)-stage classifier that is trained using active learning was shown to be effective in [6] for vehicle detection as compared to similar \( n \)-stage passively learned classifier. This is also observed by the proposed framework as shown in Fig. 9.

Fig. 9 (a) shows the vehicle detection results obtained from VeDAS using the 20-stage cascade classifiers from passive learning (i.e. using one round of training the classifiers with exact windows for the parts). Fig. 9 (b) shows the results from VeDAS by using 20-stage cascade classifiers from the proposed active learning framework. It can be seen that there are more false positives in Fig. 9(a) as compared to (b). This shows that the classifiers from the proposed active learning framework generate lesser false alarms compared to passive training that is conventionally used. Fig. 10 plots true positive rate (TPR) versus false detection rate (FDR) for detections from VeDAS with the two types of training. The curves shows more than 10% improvement in detection rates between the two training techniques.

Next, we evaluate the performance of a \( n \)-stage active learned classifier with a \( 2n \)-stage passive learned classifier,
without the second stage of symmetry-based association in VeDAS. Therefore, this comparison evaluates the performance of the training alone. This comparison is performed to show that the active learning helps to train the classifier better with queried samples and lesser number of cascade stages as compared to a passively trained classifier with more number of stages with random sampling (for negatives). This is shown using a 10-stage classifier using the proposed active learning scheme as compared to a 20-stage conventionally trained cascade. Both these classifiers are trained for detecting the full rear-view of the vehicles, i.e., parts are not used because the intention is to show the effectiveness of the learning technique. The main advantage of active learning is to reduce false alarms [6]. This is indicated clearly by the average number of false positives per frame that are obtained by the two classifiers on two different test datasets as shown in Table III. It can be seen that the active learning scheme gives 30% to 50% lesser false positives per frame on an average as compared to a 20-stage passively trained cascade classifier. This is also reflected in the true positive rates and false detection rates of the two classifiers for the two datasets as shown in Table III.

2) Evaluation & Comparison using Public Datasets: Fig. 11(a) & (b) show the detection results from VeDAS on LISA-Dense and LISA-Sunny datasets respectively. It can be seen in Fig. 11(a) that all the vehicles which are completely visible from rear are detected under varying conditions such as dense shadows and high traffic density. Similarly, in Fig. 11(b), vehicles are detected in the presence of features such as sign boards etc.

Fig. 12 plots the true positive rates (TPR) versus false detection rates (FDR) for VeDAS for the two different datasets LISA-Dense and LISA-Sunny individually. A combined plot is also shown. It can be seen that VeDAS performs better in the case of Dense dataset as compared to Sunny dataset. Additionally, the plot between TPR and false positives (FP) per frames is also shown in Fig. 13 for the combined dataset (LISA-Dense + LISA-Sunny). It shows TPRs over 95% for less than 0.4 FP/frames. This shows that the proposed method detects more than 95% of the vehicles in 10 frames while giving less than 4 FP windows. Table IV compares the different metrics between VeDAS and the active learning based vehicle detection method in [6] using the datasets - LISA-Dense and LISA-Sunny. Table IV shows that the proposed method gives higher detection rates (TPR) with lesser false detections (FDR) as compared to [6] for the dense dataset (LISA-Dense). It is to be noted that during the evaluation of this dataset, the proposed method not only detects the vehicles marked as ground truth by [6] but also additional smaller sized vehicles which are not marked in [6] (for fairness in evaluation such correct detections have not been considered for results in Table IV). The evaluation of LISA-Sunny dataset also shows similar observations in terms of TPR and FDR.

Next, we evaluate the performance of VeDAS using 5 other datasets that are listed in Table II. Sample detection results from the proposed method on TME motorway dataset and iROADS datasets are shown in Fig. 14. While Fig. 14(a) shows detection results on a sample image from TME motorway
Fig. 12. True positive rate (TPR) versus false detection rate (FDR) for LISA datasets.

Fig. 13. True positive rate (TPR) versus false positives per frame for LISA datasets.

dataset, Fig. 14(b)-(d) show the detection results varying road and weather conditions in iROADS datasets. Table V lists the TPR, FDR, average TP/frame and average FP/frame for the TME motorway and iROADS datasets. VeDAS performs at average detection rates of over 95-96% in all cases with false detection rates of less than 9%. Additionally, it was also found that VeDAS without the LUT-based window filtering showed 2-5% higher false detection rates. As mentioned previously, the LUT-based window filtering ensures that windows from the classifiers which do not satisfy the vehicle width condition are eliminated. Therefore, applying the LUT-based window condition improves the detection rates.

Table VI compares VeDAS against the vehicle detection methods in [18], [13] on TME Motorway Dayline dataset. The precision recall values for [18] and [13] in Table VII are mean values for the different variations in width and distances that are listed in [18] and [13]. It can be seen that the proposed method shows better recall rates with similar precision rates as compared to [18] and [13]. We also compared VeDAS with [2] using iROADs Daylight dataset, which is a comparatively simpler dataset as compared to TME Motorway dataset. VeDAS shows similar recall rate of 0.99 and false positives per frame of 0.01, which are also listed in [2].

3) Computation Time Analysis: In terms of computation times, VeDAS gives frame rates between 15.5 and 25.4 frames per second (fps) on an i7 CPU with 4 cores running. The variation is seen based on the type of dataset. Table VII lists the computational times of VeDAS on different datasets and compared against the respective methods. All methods are not

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caraffi [13]</td>
<td>0.90</td>
<td>0.95</td>
</tr>
<tr>
<td>Grabb [18]</td>
<td>0.85</td>
<td>0.95</td>
</tr>
<tr>
<td>VeDAS</td>
<td>0.97</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table VI

Comparison of Computation Times

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Related Work</th>
<th>VeDAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISA Dense</td>
<td>RT [6]</td>
<td>15.5 fps</td>
</tr>
<tr>
<td>LISA Sunny</td>
<td>RT [6]</td>
<td>25.4 fps</td>
</tr>
<tr>
<td>TME Motorway</td>
<td>35 fps [18], 29.6 fps [13]</td>
<td>20.2 fps</td>
</tr>
<tr>
<td>iROADS Datasets</td>
<td>NA</td>
<td>25.4 fps</td>
</tr>
</tbody>
</table>

RT: Real time and no value, NA: Not available
C. Evaluation of Partially Visible Vehicle Detection

An evaluation of Algorithm 2 showed that it is efficient in detecting partially visible vehicles where the rear-view occlusion levels are between 40% to 65%. If the occlusion is more than 65%, the classifier is not able to detect the parts. Similarly, if the occlusion level is less than 40%, the full vehicle is detected because both left and right parts are detected for the vehicle.

Fig. 15 shows sample detection results for the left and right occluded vehicles in image frames taken from LISA-Dense dataset. In order to evaluate the proposed method, 150 frames from two different segments were selected from LISA-Dense dataset. Ground truth annotation was done for left and right occluded vehicles, wherein a right/left occluded vehicle is considered if it is occluded by 40 to 60%. Table VIII lists the different accuracy metrics TPR, FDR, TP/frame and FP/frame for the left and right occluded vehicles. According to Table VIII, an average accuracy of 87% is achieved with a false detection rate of 28%. The false alarms in the case of left occluded vehicles are found to be lesser as compared to the right occluded vehicles.

![Sample detection results](image)

Table VIII: Evaluation on Partially Visible Rear-view Vehicles Dataset

<table>
<thead>
<tr>
<th>Metric</th>
<th>Right occluded vehicles</th>
<th>Left occluded vehicles</th>
<th>All (left + right)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPR</td>
<td>0.870</td>
<td>0.872</td>
<td>0.871</td>
</tr>
<tr>
<td>FDR</td>
<td>0.321</td>
<td>0.146</td>
<td>0.282</td>
</tr>
<tr>
<td>TP/frame</td>
<td>0.760</td>
<td>0.275</td>
<td>1.033</td>
</tr>
<tr>
<td>FP/frame</td>
<td>0.360</td>
<td>0.047</td>
<td>0.407</td>
</tr>
</tbody>
</table>

Conclusions

In this paper, we introduced a novel two-part based vehicle detection technique using an active learning framework and symmetry-based iterative analysis, which we call VeDAS. The active learning framework proposed in this paper when combined with the symmetry based analysis in VeDAS is shown to improve the detection rates and also reduce false alarms as compared to existing methods. Furthermore, VeDAS is enabled to detect partially visible rear-views of vehicles, which is evaluated using a new dataset for partially visible vehicle detection in highway scenarios. This is a new dataset as compared to existing motorway datasets which do not have benchmarks for partially visible rear-views of vehicles. Although VeDAS operates at real-time speeds between
15 to 25 fps, it can be further optimized at computational level to further improve the performance. Future work includes such optimizations and deployment into higher-order applications.

ACKNOWLEDGMENTS

We would like to thank all the reviewers for their valuable comments and inputs. We also thank our sponsors for their continued optimizations and deployment into higher-order applications.

REFERENCES


Mohan Manubhai Trivedi (F’08) received the B.E. (with honors) degree in electronics from Birla Institute of Technology and Science, Pilani, India, in 1974 and the M.S. and Ph.D. degrees in electrical engineering from Utah State University, Logan, UT, USA, in 1976 and 1979, respectively. He is currently a Professor of electrical and computer engineering. He has also established the Laboratory for Intelligent and Safe Automobiles, and Computer Vision and Robotics Research Laboratory, UCSD, where he and his team are currently pursuing research in machine and human perception, machine learning, intelligent transportation, driver assistance, active safety systems and Naturalistic Driving Study (NDS) analytics. He received the IEEE Intelligent Transportation Systems Societys highest honor, Outstanding Research Award in 2013.