Vision-Based Front and Rear Surround Understanding Using Embedded Processors

Ravi Kumar Satzoda, *Member, IEEE*, Sean Lee, Frankie Lu, and Mohan Manubhai Trivedi, *Fellow, IEEE*

***Abstract*—Vision-based driver assistance systems involve a range of data-intensive operations, which pose challenges in implementing them as robust and real-time systems on resource constrained embedded computing platforms. In order to achieve both high accuracy and real-time performance, the constituent algorithms need to be designed and optimized such that they lend well for embedded realization. In this paper, we present a novel two-camera-based embedded driver assistance framework that analyzes the dynamics of vehicles in the front and rear surround views of the host vehicle (ego-vehicle). In order to do this, we propose a set of integrated techniques that combine contextual cues and lane information to detect vehicles that pose high threat to the ego-vehicle. The threat analysis is then used for generating a safe maneuver zone by the proposed system, which is implemented by using two Snapdragon 810 embedded CPUs. A detailed performance evaluation and tradeoff analysis is presented using a novel multiperspective dataset (DualCam Dataset) that is released as part of this paper. In terms of accuracy, the detailed evaluations show high robustness with true positive rates greater than 95% with less than 6% false alarm rate. The proposed embedded system operates at real-time frame rates in our testbed under real-world highway driving conditions. The proposed framework was presented as a live demonstration at the 2016 Consumer Electronics Show.**

***Index Terms*—Embedded system, integrated vehicle detection, threat estimation.**

I. Introduction: Motivation & Contributions

With the advent of autonomous and semi-autonomous vehicles, embedded electronic systems form a significant component of modern vehicles [1]–[3]. Among the increasing number of such embedded electronics, active safety and intelligent perception systems play a vital role in the safety of the driver and passengers in the vehicle [4], [5]. Among these different electronic subsystems, vision-based advanced driver assistance systems (ADAS) form a significant percentage [6] because of the increasing miniaturization and decreasing costs of vision sensors or cameras. Cameras are being used to monitor 360° surround of the vehicle to perceive the surrounding environment of the vehicle [7], [8]. Although visual data provides a rich set of information for inferring the states of the surrounding of the vehicle, performing such tasks with high level of robustness under varying road and environmental conditions is still a challenging task [8]. This requirement for high accuracy also implies the use of data-intensive and computationally-intensive computer vision algorithms, which pose a challenge in implementing them as resource constrained embedded electronic systems in cars. Therefore, there is a classic tradeoff between computational efficiency (or real-time performance) and robustness of such systems.

Manuscript received March 27, 2016; revised July 19, 2016 and November 13, 2016; accepted January 9, 2017. Date of publication April 24, 2017; date of current version June 16, 2017. *(Corresponding author: Ravi Kumar Satzoda.)*

R. K. Satzoda and S. Lee are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: ravisatzodapublications@gmail.com; yhl014@ucsd.edu).

F. Lu is with the Department of Electrical and Electronic Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: frlu@eng.ucsd.edu).

M. M. Trivedi is with the Computer Vision and Robotics Research Laboratory-Laboratory for Intelligent and Safe Automobiles, University of California San Diego, La Jolla, CA 92093 USA (e-mail: mtrivedi@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIV.2017.2686084

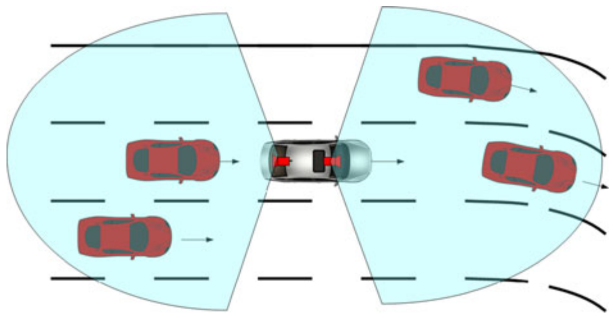


Fig. 1. Front and rear surround understanding using proposed embedded system.

In this paper, we introduce integrated and context-aware techniques that address challenges related to embedded realization of vision-based operations resulting in real-time, yet robust, embedded driver assistance systems. The main contributions of this paper can be itemized as follows: (a) vision-based techniques that synergistically combine application context and surround cues from a two-camera setup to generate a safe maneuver zone visualization in order to aid the driver, (b) an embedded framework for driver assistance that takes advantage of the integrated techniques in (a) so that the embedded system operates in real-time with high levels of robustness, and (c) a novel DualCam Dataset comprising of visual data that is collected by cameras capturing the front and rear views from the ego-vehicle (host vehicle) for detailed evaluations and benchmarking. Fig. 1 illustrates the proposed system, which comprises forward and rear facing cameras that are connected to two Snapdragon 810 embedded CPUs. The two embedded processors collaborate with each other to generate a safe maneuver zone by performing threat analysis using the surround cues. The open access DualCam Dataset is a significant contribution to the intelligent vehicles' community because it is the first of it's kind that involves multiple perspectives and high threat maneuvers.

II. Related Recent Studies

There is a rich body of literature which addresses various aspects of vision-based driver assistance systems. Lane detection [9], vehicle detection [10] and vehicle activity detection [7], in particular, have been studied extensively in various works. Lane detection has been explored in detail in [9], [11]–[14]. While [9], [11], [13] involve filter-based approaches to detect lane features, [14] employs learned classifiers on groups of pixels to detect lane features. Also, [9], [13], [14] operate the entire image, while [11] reduces the computational complexity by analyzing bands of the input image so that the method can be implemented on embedded computing platforms. [12] provides a detailed survey of existing lane detection techniques. Similarly, rear-view vehicle detection has also be studied in detail in works such as [10], [15]–[17]. A detailed survey of vehicle detection techniques is presented in [8]. Most vehicle detection approaches involve multi-scale and sliding window approaches, where appearance based classifiers such as support vector machines (SVMs), Haar-cascades etc. are used. These operations are further used to estimate the threat posed by surrounding vehicles on the ego-vehicle [18]–[20].

Although, robust vision algorithms have been proposed for these systems, there are fewer works in the context of embedded electronics that can be deployed in vehicles using resource constrained embedded computing platforms [21]–[23]. Embedded processors offer lesser computing resources as compared to conventional processors on which vision algorithms are evaluated. Therefore, conventional detection algorithms when implemented on embedded CPUs are often challenged with real-time and power consumption constraints. Therefore, the algorithms need to be re-designed in order to realize them on embedded hardware [22]. Some works have addressed such embedded constraints for deploying vision algorithms in embedded hardware. In [24], multi-perspective camera based ADASs are discussed for implementation on Texas Instruments processors. Another similar work is presented in [25], wherein multi-modal and multi-camera based ADASs are surveyed for embedded implementation. Stereo-vision systems have also been implemented on embedded platforms in works such as [26], [27]. It is to be noted that consumer vehicle manufacturers have deployed some of these algorithms as part of their vehicles using electronic control units (ECUs) for applications such as lane departure warning (LDW). Additionally, there are plug-and-play vision-based electronic assistance systems such as MobilEye [28] which can be installed in vehicles to assist the driver. Dedicated automotive grade embedded vision processors such as Texas Instruments' EVE have also been developed for deployment in vehicles [29].

The above list is only a brief review of related works. A more detailed study can be found in [23], [25]. A study of these works show that the challenges with regards to translating computer vision algorithms into robust and real-time embedded electronics are yet to be fully addressed. This is because embedded computing platforms pose a number of challenges in implementing vision algorithms in cars [22]. This includes speed and computing power of the embedded processor, communication bandwidth and battery power.

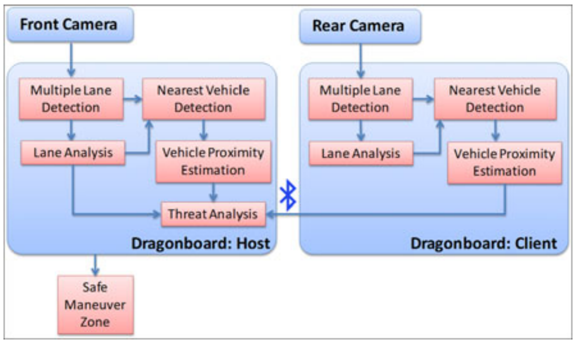


Fig. 2. Overview of the proposed embedded system: Two embedded CPUs, one each for processing forward and rear visual views from the ego-vehicle, perform a series of operations and collaborate with each other to generate a safe maneuver zone.

III. Overview & Scope of Proposed System

Before going into the details of the proposed techniques in this paper, we define the objective and scope of the proposed system. This is important because the proposed techniques use the application context in order to operate at real-time frame rates on embedded CPUs.

The objective of the proposed system is to provide assistance to the driver by performing threat analysis using the visual data that is captured by on-board cameras in the ego-vehicle. Threat analysis will be performed by analyzing the dynamics of the vehicles that are seen by forward and rear facing cameras. The proposed system comprises of two computing nodes - forward node and rear node. Each node estimates lanes, and the lane positions are then used to detect vehicles that pose *high threat* to the ego-vehicle. The positions of the high threat vehicles from the two nodes are then used to perform threat analysis. Fig. 2 lists the operations that are performed in each processing node.

Therefore, the application context is defined as analyzing the dynamics of high threat from forward and rear surround views, and perform threat analysis in order to assist the driver in providing a safe maneuver zone. In this context, we define *high threat* vehicles as follows.

Definition 1: High Threat Vehicles - The following vehicles are considered to pose threat to the ego-vehicle in the proposed system:

1) Vehicle in the ego-lane that is either in front of the ego-vehicle or approaching from behind the ego-vehicle.
2) Vehicles in front and rear of the ego-vehicle, and in the adjacent left and right lanes of the ego-vehicle.
3) If there are multiple vehicles in a particular lane, then the vehicle that is nearest to the ego-vehicle is the high threat vehicle that must be detected.

The scope of this work is limited to two cameras that capture the forward and rear views from the ego-vehicle. Therefore, the vehicles in the blind spots are not considered in this paper, although they pose high threat to the ego-vehicle. We will discuss how the proposed embedded framework is scalable to address other scenarios in Section VIII. Furthermore, the main contribution of the paper is the overall system that assists the driver with the safe maneuver zone. In order to achieve this real-time system on an embedded CPU without compromising on the robustness of the system, we propose a set of techniques for optimizing the vehicle detection process which forms the core operation of the proposed ADAS. It is to be noted that lane detection is not in the scope of this paper. We use existing methods for lane detection to meet the application requirements of the proposed system. Also, the proposed techniques are particularly designed for highway driving conditions, wherein surrounding vehicles are the obstacles that pose threat to the ego-vehicle.

IV. High Threat Vehicle Detection for Embedded Implementation

In this section, we present a technique for detecting high-threat vehicles on embedded CPU. In order to achieve real-time frame rates one resource constrained embedded computing platforms, we approach the vehicle detection problem in an informed and context based manner. The context is defined by the application requirement of detecting high-threat vehicles, which are defined previously in Definition 1. Therefore, the detection of vehicles is first restricted to three regions of interest (RoIs) - ego-lane and the two adjacent lanes. Next, the vehicle detection method does not include the conventional multi-scale and sliding window approach on the entire RoIs. Instead, a two-step method is used to detect vehicles in the RoIs. In the first step, possible hypothesis windows are generated in each RoI using a simplified approach. Therefore, hypothesis windows limit the total amount of image data wherein robust but computationally more complex classifiers can be applied. The application of such classifiers on limited regions of interest increases the frame rate and maintains high levels of robustness. The techniques are described for the image frames captured by the forward facing camera. The same techniques are also applied to the image frame from the rear-camera. Therefore, the actual accuracy of the method is still dependent on the robust appearance based classifiers, but it is ensured that the complexity of multi-scale and sliding window classifier based detection is limited to specific windows that are hypothesized by a computationally less complex method.

A. Region of Interest (RoI) Generation

According to Definition 1, there are three regions of interest (RoIs): the ego-lane, adjacent left lane and adjacent right lane. A lane estimation algorithm such as [9], [11] is employed to determine positions of the ego-lane denoted by $\mathbf{P^L}$ and $\mathbf{P^R}$ where,

$$\begin{aligned}\mathbf{P^L} &= \big[P(x^L_{y_{min}}, y_{min}), \cdots, P(x^L_{y_k}, y_k),\\ &\quad \cdots, P(x^L_{y_{max}}, y_{max})\big]^T\\ \mathbf{P^R} &= \big[P(x^R_{y_{min}}, y_{min}), \cdots, P(x^R_{y_k}, y_k),\\ &\quad \cdots, P(x^R_{y_{max}}, y_{max})\big]^T \qquad (1)\end{aligned}$$

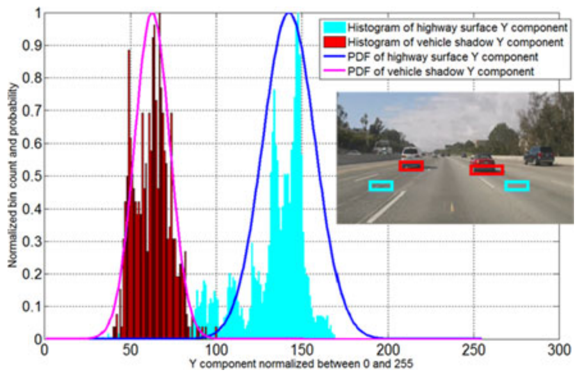


Fig. 3. Probability density functions for road surface and under vehicle region pixels. Sample annotations are shown for the road surface and under vehicle region pixels.

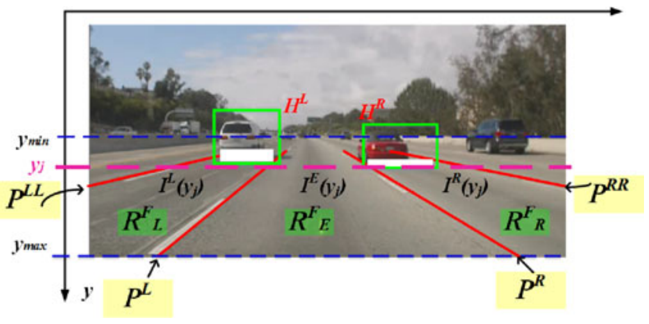


Fig. 4. Illustration of the scan line segments I^L, I^E and I^R at $y = y_j$, under-vehicle regions and hypothesis windows H^L and H^R.

$\mathbf{P^L}$ and $\mathbf{P^R}$ give the x-coordinates of the left and right lanes of the ego-lane within the image region defined by y. y represents the vertical axis along the road such that $y_{min} \leq y \leq y_{max}$. The coordinates in $\mathbf{P^L}$ and $\mathbf{P^R}$ of the ego-lane also form the right and left boundaries of the adjacent left and right lanes respectively. This is illustrated in Fig. 4. The lane detection algorithm in [11] is evaluated in great detail in [11], [30]. The accuracy of ego-lane detection is shown to be less than 7 cm in real-world coordinates.

The adjacent lanes are required for generating RoIs in the proposed system but extracting adjacent lane features of the adjacent lanes (denoted by $\mathbf{P^{LL}}$ and $\mathbf{P^{RR}}$ in Fig. 4) will increase computational complexity. In order to determine $\mathbf{P^{LL}}$ and $\mathbf{P^{RR}}$, a look-up table (LUT) based approach is used. An LUT is generated offline using the camera calibration information. The LUT stores offsets that must be either subtracted from or added to $\mathbf{P^L}$ and $\mathbf{P^R}$ in order to get $\mathbf{P^{LL}}$ and $\mathbf{P^{RR}}$ respectively, i.e.,

$$\mathbf{P^{LL}} = \mathbf{P^L} - \Delta_x \text{ and } \mathbf{P^{RR}} = \mathbf{P^R} + \Delta_x \qquad (2)$$

where Δ_x represents the LUT vector with the offsets. Δ_x is computed using the camera calibration parameters. Let $\mathscr{H} \in \mathbb{R}^{3\times3}$ represent the calibration homography matrix that maps image view to the inverse perspective mapping (IPM) view [31]. The computation of $\mathscr{H}$ is detailed in [31], where $\mathscr{H}$ transforms an image (which has a perspective deformation) into a top-view or the IPM view. In the IPM view, the lanes are supposed to be parallel and of specific width w_L. Let y_{min} and y_{max} (refer to (1)) in the image domain correspond to $y^W = 1$ and $y^W = y^W_{max}$ in the IPM domain. This represents the depth along the road from the ego-vehicle for which the lane analysis and the subsequent threat analysis is valid. Therefore, for every $y^W_i \in [1, y^W_{max}]$, we apply the following equation:

$$\begin{bmatrix}\Delta_i & y_i & 1\end{bmatrix}^T = \mathscr{H}^{-1}\begin{bmatrix}w_L & y^W_i & 1\end{bmatrix}^T \tag{3}$$

where Δ_i is the x-offset which is stored in the LUT Δ_x for the y_i-th coordinate in the image domain. The above equation is applied for all values of y^W_i to create the LUT. This entire process is a one-time offline process. The LUT is then applied according to (2) to generate the boundaries for the left and right adjacent lanes during runtime resulting in the three RoIs: adjacent left lane, ego-lane and adjacent right lane (denoted by R^F_L, R^F_E and R^F_R in Fig. 4). This method to detect the adjacent RoIs is particularly effective when the lane widths are fixed as seen in the case of highways.

It is to be noted that the proposed LUT based approach is also valid during lane change maneuvers made by the ego-vehicle. This is because the relationship between the IPM image and the perspective (or image) domain remains the same during lane change maneuver. During lane change the lanes are laterally shifted in the IPM domain but the width remains the same. Therefore, (3) results in the same offsets during lane change. The only assumption for this is that the lens and camera distortions need to be corrected beforehand.

B. Region of Interest (RoI) Based Vehicle Detection

Given the three RoIs, high-threat vehicles are detected in each region. As listed in Definition 1, the system will detect three vehicles, one from each RoI, which are nearest to the ego-vehicle because the presence of these vehicles poses varying levels of threat to the ego-vehicle.

Instead of applying the vehicle classifier directly in each of the regions, hypothesis candidates are first generated. In order to do this, each RoI is analyzed for the presence of under-vehicle dark regions (shown in Fig. 3 inset annotations). The lower parts of the leading vehicles in front of the ego-vehicle are usually distinctive with dark pixels as compared to the surrounding road surface. The dark pixels are contributed by either the bumper of the vehicle or the shadow cast by the vehicle or both. It is to be noted that the under vehicle region is not dependent on the shadow cast by the vehicle alone. The darker under vehicle regions are a result of the proximity of the vehicle chassis to the road surface. Therefore, the under vehicle regions apply to vehicles being detected in cloudy or rainy weather conditions also. In order to detect the presence of this under vehicle region, the annotated dataset for training the classifiers (which will be used later in the proposed method) is used. The under-vehicle regions of the training samples were annotated. Given that the cameras capture the video in YUV color format, the mean Y component of the under-vehicle regions is collected from the training data resulting in the probability density functions $P_{UV}(\mu_{UV}, \sigma_{UV})$ and $P_{RS}(\mu_{RS}, \sigma_{RS})$ for the under-vehicle region and the road surface respectively as shown in Fig 3.

Given a test image frame I with the boundaries of the RoIs computed as described above, I is scanned along the y-direction starting from the bottom of the image, i.e., the point that is nearest to the ego-vehicle. For a particular y-position y_j, the image pixels in that scan line are divided into the following three segments corresponding to the three RoIs:

$$\begin{aligned}
I^L(y_j) &= I(x, y_j) \text{ where } \mathbf{P^{LL}}(x, y_j) \le x \le \mathbf{P^L}(x, y_j)\\
I^E(y_j) &= I(x, y_j) \text{ where } \mathbf{P^L}(x, y_j) \le x \le \mathbf{P^R}(x, y_j)\\
I^R(y_j) &= I(x, y_j) \text{ where } \mathbf{P^R}(x, y_j) \le x \le \mathbf{P^{RR}}(x, y_j)
\end{aligned} \tag{4}$$

The image pixels in each of three scan line segments $I^L(y_j)$, $I^E(y_j)$ and $I^R(y_j)$ are then thresholded using the PDFs for under-vehicle region and road surface - $P_{UV}(\mu_{UV}, \sigma_{UV})$ and $P_{RS}(\mu_{RS}, \sigma_{RS})$ using the following equation:

$$I(x_k, y_j) = \begin{cases} 1 & \text{if } P_{UV}(I(x_k, y_j)) \ge P_{RS}(I(x_k, y_j)) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

(the superscripts L, R and E are omitted in the above equation for simplicity). After thresholding the scan line segments, the number of pixels that are classified as under-vehicle region is determined for each segment. If the number of under-vehicle region pixels form $\beta\%$ of the scan line segment, then that scan line segment is marked as an under-vehicle region segment. Therefore, when the scan line at y is scanned completely from $x = 0$ to $x = x_{max}$, possible vehicle regions in the three RoIs are determined. This process is repeated with the next scan line at $y = y_j - 2$. In each RoI, the number of *consecutive* under-vehicle region segments are counted which are denoted by n^L, n^E and n^R corresponding to the three RoIs. If n^L is greater than κ then the n^L scan line segments are considered as possible constituents of under-vehicle region in the adjacent left lane or RoI. The same is repeated with the ego-lane and the adjacent right lane RoIs. If an RoI is found to have possible under-vehicle region, then a hypothesis window H is drawn in which vehicle classifiers are applied. The dimensions of this window are computed in the following way:

$$\begin{aligned}
w_h &= (\text{width of lane at } y = y_{j_{max}}) + padding\\
h_h &= (n + \alpha \times \text{ width of lane at } y = y_{j_{min}}) + padding
\end{aligned} \tag{6}$$

where w_h and h_h are the width and height of the hypothesis window for the under-vehicle region that starts at $y = y_{j_{max}}$ and ends at $y = y_{j_{min}}$; α is the aspect ratio factor that is dependent on the vehicle dimensions; n is the height of the under-vehicle region that is detected in the RoI, i.e., n is either n^L or n^E or n^R depending on the RoI. All the different parameters are illustrated in Fig. 4.

Note on setting the parameters: α is learned form the training data. Fig. 5 shows the distribution of the relationship between the width and height of 20,000 annotated vehicles in the training data. α is set to $1/1.25$ based on this data. Varying κ affects the accuracy and computation time of the algorithm. A detailed analysis on κ is presented in Section VII. The value β is set to 40% so that it can handle vehicles that are changing lanes. $padding$ is heuristically set to 25 pixels.

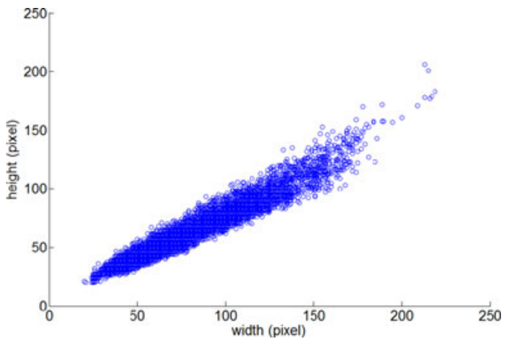


Fig. 5. The width versus height of over 20,000 vehicle annotations is plotted to get the factor α.

Therefore, instead of applying the computationally more complex vehicle detection classifiers on the entire image, they are applied in these limited image patches Hs. In case incorrect hypothesis windows are generated due to presence of artifacts such as shadows of trees and other road-side structures, such false positive windows are eliminated by the classifiers that are applied in the hypothesis windows. In this work, Haar-Adaboost classifiers are applied to detect the vehicles in H. In order to use the classifiers, over 11,000 positive annotations of vehicles were used to train Adaboost cascade classifiers with Haar-like features using the active learning methodology described in [17]. The active learning process uses a two-step cascade classifier learning. In the first step, the cascade is trained with features that are extracted from positive annotations and random negative annotations. The trained classifier is then used to mine for hard negatives, which are used to train another classifier, which is called actively learned classifier.

The actively learned classifiers are applied within H (if found in each RoI) to detect vehicles. If a vehicle is detected within H, the remaining scan lines in the RoI above $y = y_{j_{min}}$ are are not processed anymore. The above approach is repeated for the three different RoIs resulting in the detection of a maximum of three vehicles corresponding to the three nearest vehicles in the three RoIs. A Kalman tracker [32] is then applied to the detected windows in order to track the vehicles continuously in the RoIs. The state space equations of the Kalman filter for tracking the vehicles in the RoIs are listed below:

$$X^{t_i+1|t_i} = AX^{t_i|t_i} \qquad \text{and} \qquad Y^{t_i} = MX^{t_i}$$
$$\text{where } X = [x\ y\ w\ h]^T \ \ A = M = I_{4\times 4} \quad (7)$$

X in the above equations represents the position and the size of the bounding box of the vehicle. Therefore, in this work, the Kalman tracker is based on the bounding boxes that are being detected. However, a more complex model could be used that includes the speeds and the heading angles of the surrounding vehicles and ego-vehicle.

V. Safe Zone Estimation Using Dual Camera Setup

As discussed in Section III and Fig. 2, there are two Snapdragon 810 processors with a camera connected to each processor such that they capture forward and rear views from the ego-vehicle. Each board performs RoI estimation followed by the detection of high-threat vehicles using the proposed techniques mentioned in the previous sections. After detecting the vehicles, two sets of detection windows are generated $\Psi^F = \{W_L^F, W_E^F, W_R^F\}$ and $\Psi^R = \{W_L^R, W_E^R, W_R^R\}$, where the superscripts ^{F} and ^{R} refer to forward and rear RoIs respectively; and the subscripts $_L$, $_E$ and $_R$ refer to the adjacent left, ego and adjacent right lanes respectively. A detection window W is a set of four parameters, i.e. $W = [x, y, w, h]$, where (x, y) denotes the coordinates of the top left corner of the window and w and h denote the width and height of the detection window. If the vehicle detector does not detect a vehicle in a lane, that window is set to a null value.

Given the detection windows in Ψ^F and Ψ^R, the relative distances of the vehicles are estimated using the inverse perspective transformation matrix $\mathscr{H}$ that was previously described in Section IV-A. The mid point of the bottom edge of the detection window is used to determine the position of the vehicle along the road from the ego-vehicle in the following way. Let us consider a detection window $W = [x_W, y_W, w_W, h_W]$ (we remove the superscripts and subscripts for clarity). The following equation is applied to get the position of the vehicle in the IPM domain:

$$\begin{aligned} \begin{bmatrix} x_I & y_I & 1 \end{bmatrix}^T &= \mathscr{H}\begin{bmatrix} x_W & y_W + h_W & 1 \end{bmatrix}^T \\ d_V &= \lambda y_I \end{aligned} \quad (8)$$

where d_V is the relative distance of the detected vehicle from the ego-vehicle in ground plane coordinates, λ is the calibration coefficient and $\mathscr{H}$ is the homography matrix for IPM. The calibration coefficient λ converts the coordinates (x_I, y_I) in the IPM domain into real-world coordinates in meters. λ is computed using the camera calibration and it is a one time setup computation. Therefore, the forward and rear processors produce two vectors $\mathbf{d_V^F} \in \mathbb{R}^3$ and $\mathbf{d_V^R} \in \mathbb{R}^3$ corresponding to the distances of the vehicles in front and rear of the ego-vehicle. If any window is null (no vehicle in the RoI), then that particular distance d is set to d_{max} which is the maximum distance from the ego-vehicle which is being processed for threat estimation. The rear Snapdragon now acts as a client to the forward facing Snapdragon processor and sends $\mathbf{d_V^R}$ to the host processor, i.e. the forward Snapdragon processor. This communication occurs via a bluetooth link. The host combines $\mathbf{d_V^F}$ and $\mathbf{d_V^R}$ to generate a 6-valued vector $\mathbf{d_V} = \left[d_L^F, d_E^F, d_R^F, d_L^R, d_E^R, d_R^R\right]^T$. The first three values in $\mathbf{d_V}$ denote the distances from the forward view and the last three values denote the distances of the vehicles from rear-view.

The host processor uses the distance vector $\mathbf{d_V}$ to compute risk vector Γ using a risk function $f_r(\cdot)$. The risk function can be defined in multiple ways depending on the factors that are considered for risk estimation. In this work, we consider a distance based risk function which computes the risk vector Γ using the following equation:

$$\Gamma = f(\mathbf{d_V}, d_{max}) = \left(1 - \frac{\mathbf{d_V}}{d_{max}}\right) \quad (9)$$

Γ is a six-valued vector with six risk values γ_i corresponding to the six RoIs, where $\gamma_i \in [0, 1]$. γ_i tends to 0 if the vehicle

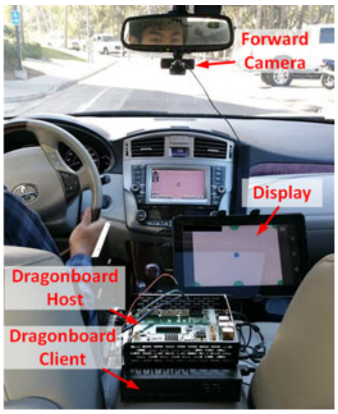


Fig. 6. Embedded system setup in our testbed with two Snapdragon 810 development boards that are connected to the cameras (rear camera is not seen in the image).

TABLE I
DUALCAM-VEHICLE DATASET DETAILS

	# frames	# veh.	Remarks	Forward-Rear views
S1	1016	1011	late afternoon, large shadow area	
S2	1018	704	lane drifting, medium traffic	
S3	1006	595	cloudy, lane drifting, large variety of road surfaces (dark, light gray, old lane marks, crack lines)	
S4	1000	1175	cloudy, medium-desne traffic, bridge shadow	
S5	1012	697	tree shadow, bridge shadow, dark patches on road surface	
Total	5052	4182		

is far away from the ego-vehicle. If a vehicle is close to the ego-vehicle, γ_i approaches 1.

The threat values in Γ are used to visualize a safe maneuver zone for the ego-vehicle. Some sample visualizations and an analysis of drive using the two-camera based setup on Snapdragon embedded processors are presented in the next section. This risk metric is based on the relative distance only and we would like to highlight that this is one of the possible definitions for the risk function $f(\cdot)$. We discuss other possible factors influencing risk function in Section VIII.

VI. EMBEDDED SYSTEM SETUP

The proposed techniques are implemented on two embedded Qualcomm Snapdragon 810 CPUs on two development kits called Dragonboards. Snapdragon 810 processor is a 64-bit octa-core CPU running at 2 GHz, along with the latest Android 5.0 Lollipop operating system. In addition, it has a 4 GB on board DDR4 memory and supports a rich set of I/O interfaces and connectivity including Wifi and Bluetooth 4.1. It is to be noted that the Snapdragon processors are widely used in commercially available Android smart phones and tablets. Therefore, the proposed system that is currently implemented on Dragonboards is readily transferable to any Snapdragon based smart phone or tablet, as long as the device is running an Android operating system. Additionally, Logitech C920 USB webcams are used to capture the input video streams with a resolution of 640×480.

Fig. 6 shows the system setup in our testbed. The two Dragonboards are named as host and client, where the front and rear facing cameras are connected to the host and client CPUs respectively (the rear camera is not shown in Fig. 6). The cameras require a one-time calibration for lane estimation, so that homography is generated for inverse perspective mapping step in lane estimation. Both the Dragonboards perform high threat vehicle detection followed by threat assessment. After the client Dragonboard determines the threat posed by the rear vehicles, the threat values are transferred to the host board via Bluetooth connection to determine the safe maneuver zone as described in the previous section.

In terms of software, the proposed algorithms are developed in C/C++ with a mixture of OpenCV and FastCV functions. While OpenCV is the open source computer vision library, FastCV is a mobile optimized computer vision library developed by Qualcomm. It can be used on any ARM-based processors, but it is best tuned for Snapdragon processors. Most of the proposed techniques were implemented using FastCV libraries but if they did not have the necessary functions (such as Haar-Cascade classifiers), then such functions were implemented using OpenCV.

VII. DATASETS AND PERFORMANCE EVALUATION

In this section, we present a detailed evaluation of the proposed techniques and the system. In addition to accuracy related metrics, an analysis of the computational speed is also presented. In order to maintain consistency, all the evaluations are performed with the algorithms running on one CPU core only of Snapdragon 810 processor.

A. *Datasets*

A set of datasets called as Dualcam-Vehicle is released as part of this paper. This includes 5 video segments which are selected from naturalistic drives. These segments include driving conditions that pose different levels of threat to the ego-vehicle from surrounding vehicles in the ego-lane and the adjacent lanes. Moreover, each of the 5 datasets in Dualcam-Vehicle dataset comprises two videos - the first capturing the forward view and the second capturing the rear view from the ego-vehicle. It is to be highlighted that this is the first public dataset that has two camera perspectives in this manner. Additionally, the segments are particularly chosen to include different levels of threat from the two perspectives. The five data sequences S1, S2, S3, S4 and S5 in Dualcam-Vehicle dataset have a total of over 5000 image frames comprising over 4100 vehicles in the three lanes and two camera perspectives. The details of the datasets and sample images are listed in Table I. Table I also lists the different traffic and lighting conditions that are captured in the dataset.

Considering the anonymity condition for review, we are not releasing more details and sample clips of the dataset in the

TABLE II
ACCURACY RESULTS AND TIMING FOR ALL SEGMENTS IN DUALCAM-VEHICLE DATASET

	Left Lane	Ego Lane	Right Lane			
	TP/FP/FN	TP/FP/FN	TP/FP/FN	TPR	FDR	Timing ms (fps)
S1-F	128/5/1	0/1/0	154/5/2	0.99	0.04	62 (16)
S1-R	284/35/13	0/0/0	422/0/7	0.97	0.05	
S2-F	301/19/1	0/0/0	57/0/14	0.96	0.05	66 (15)
S2-R	62/0/6	116/0/0	143/35/4	0.97	0.10	
S3-F	152/1/2	159/4/2	0/0/0	0.99	0.02	66 (15)
S3-R	0/0/1	0/0/0	276/1/3	0.99	0.00	
S4-F	285/3/18	30/1/29	99/32/4	0.89	0.08	71 (14)
S4-R	189/2/28	133/0/12	321/10/27	0.91	0.02	
S5-F	53/0/3	20/0/0	4/3/3	0.93	0.04	76 (13)
S5-R	90/16/1	88/27/0	216/24/16	0.96	0.15	
Total	1544/81/74	546/33/43	1692/110/80	0.95	0.06	66 (15)

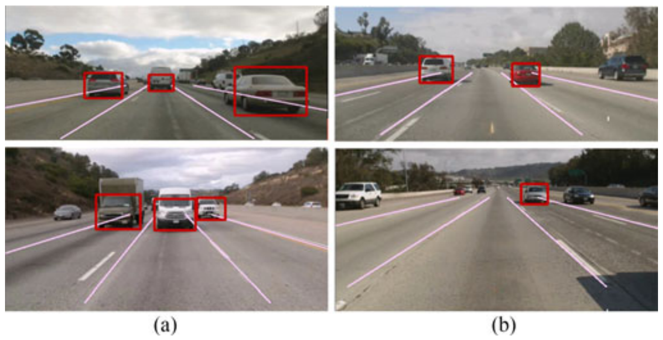


Fig. 7. Detection results of high threat vehicles within the distance indicated by the estimated lanes. Results are taken from two segments. Top row: results on image frames from forward looking camera. Bottom row: results on rear camera images. (a) drive segment 1. (b) drive segment 2.

review manuscript. Once the paper is published, the dataset will be made public for research studies.

B. Accuracy Evaluation

The Dualcam-Vehicle datasets are used to evaluate the high-threat vehicle detection method. The accuracy measures are computed for the high-threat vehicles only. In order to perform this evaluation, the five Dualcam-Vehicle datasets were annotated for the three nearest vehicles in the three RoIs - adjacent left and right lanes, and the ego-lane. This annotation is performed for both the forward view and rear view video segments. The maximum distance d_{max} within which the vehicles are detected to be posing high threat is set to 40 meters from the ego-vehicle in both perspectives (forward and rear).

Table II lists the accuracy measures for the five video segments in Dualcam-Vehicle datasets. The results are listed for forward and rear views separately. The number of true positives (TP), false positives (FP) and false negatives (FN or missed detections) are listed for each lane (left, ego and right) separately. The numbers are then used to compute two metrics True Positive Rate (TPR) and False Detection Rate (FDR) using the following equations: $\text{TPR} = TP/(TP + FN)$ and $\text{FDR} = FP/(FP + TP)$. TPR and FDR can also be used to derive precision and recall in the following manner: Recall = TPR and FDR = 1-precision. We present all results in TPR and FDR because we use the classifiers that are used in [17] and [15] for the hypothesis verification step. The use of similar metrics will help to show in subsequent sections that the proposed two-step method does not compromise robustness as compared to original methods, and yet operates at real-time frame rates on embedded CPUs.

It can be seen that the TPR for the entire dataset (all five sequences) is over 95% with a false alarm rate or FDR of less than 6%. These numbers are computed for more than 4100 instances of cars in the whole dataset. Some detection results are shown in Fig. 7. Optimizing the conventional multi-scale, sliding window approach using the proposed techniques has not reduced the detection accuracy. Applying active learning classifiers using conventional method results in an accuracy of 96% with a false alarm rate of 8% on the same datasets. The higher false alarms are because of the features on the sides of the road that give false positives.

C. Computation Time Analysis

Although high accuracy is achieved, total computation time is also important for embedded realization. The above accuracy evaluations are determined for the entire system including image acquisition, lane estimation using [11], proposed high threat vehicle detection method, threat assessment and safe zone visualization on Snapdragon 810 development platform. The average computation time per frame is computed for each dataset by finding the mean of the computation times of all the frames in the dataset. The last column in Table II lists the timing per frame in seconds and the resulting frame rate in frames per second (fps).

Table II shows that the average frame rate for the entire dataset is around 15 fps which is considered as real-time in ADAS applications [33], with the lowest frame rate being 13 fps for S5 segment in the DualCam dataset. S5 is a more complex segment as compared to the rest of the segments in the dataset, which is also reflected in lower accuracy rates in Table II. We conducted a detailed timing evaluation of the conventional multi-scale sliding window Haar-Adaboost cascade classifier by varying the different parameters of the cascade classifier such as maximum hit rate, maximum false alarm rate and number of cascade stages. The evaluation showed that the multi-scale sliding window approach of applying the cascade classifier to the test sequences gave a *best* case timing of 0.98 seconds resulting in 1 frame per second which violates the real-time requirement of the active safety system.

D. Computation Time and Accuracy Tradeoff

We now present an analysis of the tradeoff between computation time and accuracy of the proposed method. This analysis will also enable identifying appropriate parameters for the algorithm. Fig. 8 plot the FDR and TPR values (accuracy metrics) along with the computation time of the vehicle detection module on Snapdragon 810 processor. One of the key

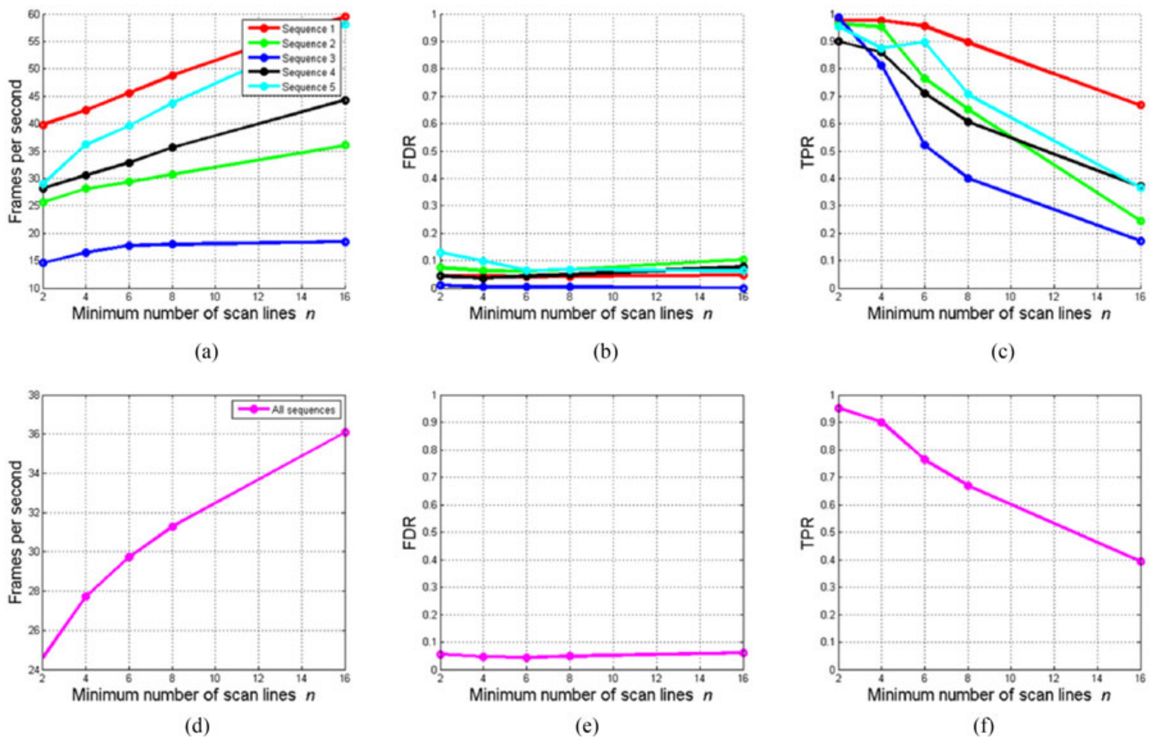


Fig. 8. Tradeoff analysis results showing computation speed of vehicle detection on Snapdragon 810 embedded CPU along with the TPR-FDR values for individual sequences in (a)–(c) and the overall dataset in (d)–(e).

parameters that effects the accuracy of the algorithm is the number of consecutive under vehicle region segments that must be detected in each RoI which is denoted by n (thresholded by κ) in Section IV. This parameter is varied to evaluate the performance of the algorithm in terms of both accuracy and computation time. It can be seen from Fig. 8 the TPR is as high as 97% for the least setting of $n = 2$ with false detections as low 1%. Lower value of n also refers to more processing resulting in lower frame rates. However, the lowest frame rate among all the sequences is still close to 15 fps. TPR, FDR and the computation times are averaged over all sequences in the dataset to show the overall measures in Fig. 8(d)–(f).

An important observation from Fig. 8 is that if n is increased to 4, the frame rates increase from 25 fps to nearly 28 fps but the TPR is still over 90% with FDR less than 6%. When n is increased further to 6, it can be seen that although TPR drops to 75%, false alarms are still at 6%. These measures and accuracy-timing tradeoff plots indicate that setting $n = 2$ to the lowest value gives a real-time system with the robust detections.

1) Comparison with other Methods: In order to compare the performance of the proposed methods on embedded CPUs, the multi-scale and sliding window methods for vehicle detection [17] and [15] are implemented on the same embedded CPUs (Snapdragon 810). The frame rates of these methods for detecting vehicles for the same image resolutions were found to be not more than 1.5 frame per second. In contrast, the proposed framework, which includes image acquisition, lane detection, high-threat vehicle detection and safe maneuver zone estimation, operates at real-time frame rates on Snapdragon 810 embedded CPUs.

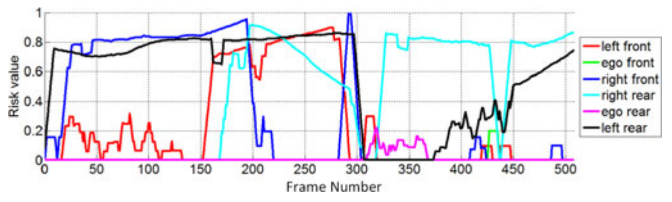


Fig. 9. Threat vector Γ is plotted for Sequence 1 with over 500 frames.

E. Safe Zone Estimation

In this sub-section, we present the threat/risk estimation results that are computed using one of the sequences (S1) in Dualcam-Vehicle dataset. Fig. 9 plots the risk vector Γ for the 500-frame sequence S1. It can be seen that the risk values change in different ways during the drive sequence. The risk values are approaching 1 in certain segments of the drive in Fig. 9, which implies that there are multiple instances when there is high level of threat to the ego-vehicle from the surrounding vehicles.

An analysis of this plot can reveal a variety of semantics that are related to the dynamics of the surrounding vehicles with respect to the ego-vehicle. For example, there is possibility of tailing vehicle behind the ego-vehicle in ego-lane for nearly two-thirds of the time in this sequence. This is indicated by the magenta plot in Fig. 9. Similar high values of threat are also seen from the vehicles approaching the ego-vehicle from the rear right lane (cyan plot in Fig. 9). Additionally, the vehicle in front of the ego-vehicle in the adjacent left lane also posed high threat for about 150 frames or 10 seconds. The threat values in Fig. 9 are verified by visually going through the segments.

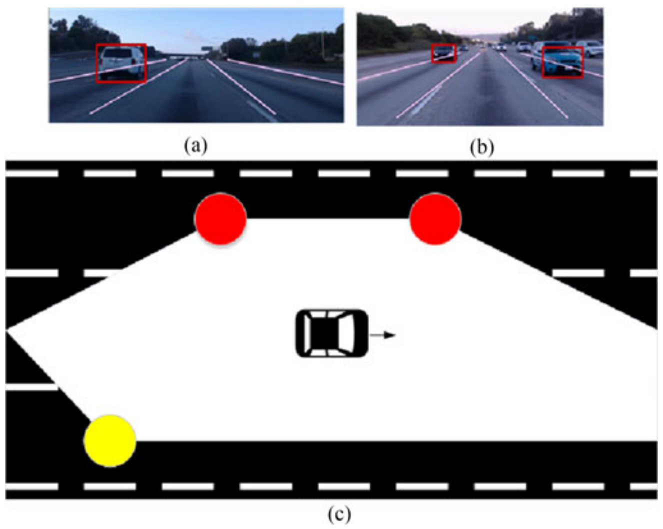


Fig. 10. High threat vehicles detected in (a) front and (b) rear of the ego-vehicle for particular time instance from S1 segment. (c) Safe zone visualization implemented on two Snapdragon 810 embedded CPUs using the threat analysis.

An example is shown in Fig. 10(a) and (b). The risk vector at $t = 260$ is given by $\Psi = \{0.87, 0, 0, 0.64, 0, 0.85\}$ which shows three high values in Fig. 9 corresponding to the forward left, rear left and rear right lanes. This can be seen from the vehicles detected in Fig. 10(a) and (b) (it is to be noted that the rear-view image needs to be flipped along the vertical axis so that the rear left lane is aligned with the forward left lane).

The risk values are also used to visualize the safe maneuver zone by generating a polygon as shown in Fig. 10(c) for the same pair of images at $t = 260$. The vehicle icon in the center denotes the ego-vehicle, whereas the filled bubbles refer to the presence of vehicles in the RoIs. The colors filled in the bubbles represent the level of risk to the ego-vehicle due to the surrounding vehicles, where green represents a safe vehicle and red represents a high threat vehicle. The visualization in Fig. 10 is the representative output to driver in the proposed embedded system operating on Snapdragon CPUs.

F. Live Demonstration of Embedded Framework

We would like to highlight that the entire embedded system is currently running at real-time frame rates in our testbed under real-world driving conditions. After a few hundred hours of testing using our testbeds, the embedded system has been demonstrated live in various forums. Most notable is the demonstration of this system in the 2016 Consumer Electronics Show (CES) in Las Vegas in January 2016. This demonstration was one of the six demonstrations that were demonstrated in a Maserati vehicle at the Qualcomm Automotive Booth. Fig. 11 shows the display of the visualization that was generated for the demonstration in CES.

VIII. Discussions

In this section, we discuss the salient features of the proposed system followed by a discussion on it's limitations and how they will be addressed.

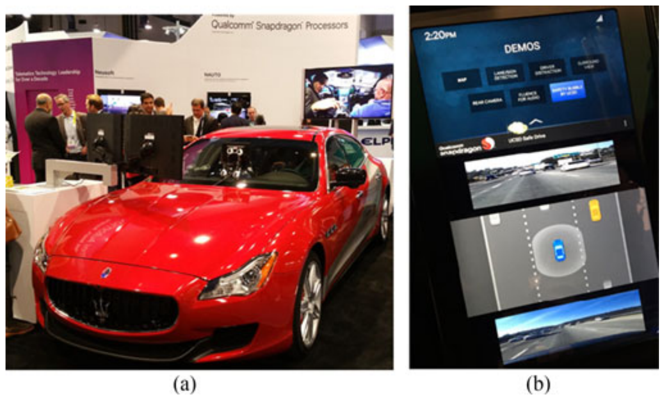


Fig. 11. (a) Maserati vehicle with the demonstration of the proposed embedded framework, (b) Visualization of the safe zone that is being demonstrated inside the vehicle.

A. Differences Compared to Existing Solutions

It is to be highlighted that we proposed an embedded electronic system that enables a different set of operations as compared to existing ADAS. While most commercially available ADAS such as Mobileye use one forward looking camera to aid the driver in operations such as lane departure warning, pedestrian detection, headway monitoring etc., the proposed system is a multi-camera setup in which multiple processing engines are collaboratively functioning to monitor the forward and rear views from the ego-vehicle. Based on available literature, this is the first embedded solution that employs a two-camera system to give safe zone maneuvers in real-time during drives. Although there is academic literature on multi-camera based driver assistance, an embedded electronic system with the kind of functionality that is described in this paper is not presented before. Additionally, the proposed system can also be used as complementary technology to existing ADAS. For example, any commercial lane departure warning can be integrated with the proposed system to provide additional functionality.

B. Limitations & Opportunities

We limited the scope of this paper to specific conditions as presented in Section III so as to present the system in the limited number of pages. We present some of the limitations and opportunities that need to be addressed and explored as part of future initiatives.

(i) In terms of the constituent vision algorithms, the vehicle detection classifiers are currently trained for detecting complete rear or front views of vehicles. Therefore, when a vehicle is passing or receding the ego-vehicle on the adjacent lanes, the classifiers do not detect them. This could be resolved by applying overtaking vehicle detection methods such as [34] in specific regions of the input frames without incurring additional time.

(ii) The proposed system is implemented on one CPU core in each Snapdragon 810 processor. Although the proposed system does give real-time frame rate of 15 fps, the speed could be increased further by parallelizing the proposed system using the multiple cores on the Snapdragon 810 processor.

(iii) The current two camera setup does not capture the vehicles in blind spots of the ego-vehicle, which pose high threat to the ego-vehicle. This can be addressed by implementing Item (ii) above, i.e., using the multi-core architecture will enable capturing additional visual data using another two cameras that are monitoring the blindspots. Monitoring the blind spots also along with the current two views will make the proposed system a complete 360° surround view monitoring system.

(iv) The distance-based risk metric that is used in this paper is one of the many possibilities. Also, additional parameters such as velocities, yaw-rates and projected trajectories of the ego-vehicle and surrounding vehicles can help to make the risk metric more complete. There are more items that can be addressed but the above four are immediate items that can be taken advantage of.

C. Scalable Embedded Framework

We would like to highlight that the proposed embedded system is also a scalable embedded framework. This is because more cameras can be added to the system, where each camera can be connected to one CPU. Each CPU processes the visual data from its visual perspective and the semantic information of the dynamics of the vehicles its perspective can be fused with the semantics from other CPUs to generate a complete 360° surround scene analysis. The proposed embedded system can scale to include more cameras, without overloading one single CPU and consequently resulting in real-time operation. Therefore, blind spots and all other missing perspectives can also be addressed by scaling the system with more cameras and processors.

IX. Conclusions & Future Directions

In this paper, we presented a detailed analysis, evaluation and discussion on an embedded driver assistance system that includes a two-camera setup to generate a safe maneuver zone around the ego-vehicle. In order to design this system using embedded CPUs on Snapdragon 810 processor, application requirement of detecting high-threat vehicles is combined with a variety of optimization techniques so that computationally complex Haar-Adaboost classifiers are applied to selective image patches. It was shown that the resulting high-threat vehicle detection method enables the entire system to operate at a minimum of 15 fps and an average of 25 fps unlike the multi-scale sliding window approach which operates at 1 fps. More importantly, it is ensured that the high rates of accuracy in detecting high-threat vehicles are maintained. A detailed discussion about the proposed system is also presented to draw possible future directions in order to improve the efficiency and robustness of the system. Future work will also include incorporating such a system as part of naturalistic driving studies [35].

Acknowledgment

The authors would like to thank our sponsors for their support during the project. In particular, they thank Qualcomm for presenting a variant of this work in the 2016 Consumer Electronics Show. The authors thank all their colleagues at Laboratory for Intelligent and Safe Automobiles (LISA) for their constant support.

References

[1] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part I : Distributed system architecture and development process," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7131–7140, Dec. 2014.

[2] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part II : A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Trans. Ind. Electron.*, vol. 62, no. 8, pp. 5119–5132, Aug. 2015.

[3] R. K. Satzoda and M. M. Trivedi, "Efficient lane and vehicle detection with integrated synergies (ELVIS)," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops Embedded Vis.*, 2014, pp. 708–713.

[4] S. Chakraborty, M. Lukasiewycz, C. Buckl, S. Fahmy, P. Leteinturier, and H. Adlkofer, "Embedded systems and software challenges in electric vehicles," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Mar. 2012, pp. 424–429.

[5] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 90–104, Mar. 2016.

[6] G. Liu, F. Worgotter, and I. Markelic, "Stochastic lane shape estimation using local image descriptors," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 1, pp. 13–21, Mar. 2013.

[7] A. Almagambetov, S. Velipasalar, and M. Casares, "Robust and computationally lightweight autonomous tracking of vehicle taillights and signal detection by embedded smart cameras," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3732–3741, Jun. 2015.

[8] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the Road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1773–1795, Dec. 2013.

[9] A. Borkar, M. Hayes, and M. T. Smith, "A novel lane detection system with efficient ground truth generation," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 365–374, Mar. 2012.

[10] B.-F. Wu, C.-C. Kao, C.-L. Jen, Y.-F. Li, Y.-H. Chen, and J.-H. Juang, "A relative-discriminative-histogram-of-oriented- gradients-based particle filter approach to vehicle occlusion handling and tracking," *IEEE Trans. Ind. Electron.*, vol. 61, no. 8, pp. 4228–4237, Aug. 2014.

[11] R. K. Satzoda and M. M. Trivedi, "Selective salient feature based lane analysis," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2013, pp. 1906–1911.

[12] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: A survey," *Mach. Vis. Appl.*, vol. 25, no. 3, pp. 727–745, Feb. 2012.

[13] J. McCall and M. Trivedi, "Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 20–37, Mar. 2006.

[14] R. Gopalan, T. Hong, M. Shneier, and R. Chellappa, "A learning approach towards detection and tracking of lane markings," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1088–1098, Sep. 2012.

[15] R. K. Satzoda and M. M. Trivedi, "Multi-part vehicle detection using symmetry derived analysis and active learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 926–937, Apr. 2016.

[16] B. Pepikj, M. Stark, P. Gehler, and B. Schiele, "Occlusion patterns for object class detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3286–3293.

[17] S. Sivaraman and M. M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 267–276, Jun. 2010.

[18] H. Takahashi, D. Ukishima, K. Kawamoto, and K. Hirota, "A study on predicting hazard factors for safe driving," *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 781–789, Apr. 2007.

[19] G. Ogawa, K. Kise, T. Torii, and T. Nagao, "Onboard evolutionary risk recognition system for automobiles toward the risk map system," *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 878–886, Apr. 2007.

[20] S. Sivaraman and M. M. Trivedi, "Dynamic probabilistic drivability maps for lane change and merge driver assistance," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2063–2073, Oct. 2014.

[21] P. Jeong and S. Nedevschi, "Efficient and robust classification method using combined feature vector for lane detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 4, pp. 528–537, Apr. 2005.

[22] F. Stein, "The challenge of putting vision algorithms into a car," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 89–94.

[23] R. K. Satzoda and M. M. Trivedi, "On enhancing lane estimation using contextual cues," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 11, pp. 1870–1881, 2015.

[24] S. Dabral, "Trends in camera based automotive driver assistance systems (ADAS)," in *Proc. IEEE 57th Int. Midwest Symp. Circuits Syst.*, 2014, pp. 1110–1115.

[25] J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Vision-based driver assistance systems: Survey, taxonomy and advances," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, Sep. 2015, pp. 2032–2039.

[26] G. P. Stein, Y. Gdalyahu, and A. Shashua, "Stereo-assist: Top-down stereo for driver assistance systems," in *Proc. IEEE Intell. Veh. Symp.*, 2010, pp. 723–730.

[27] A. D. Costea, A. V. Vesa, and S. Nedevschi, "Fast pedestrian detection for mobile devices," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, 2015, pp. 2364–2369.

[28] Mobileye, 2015. [Online]. Available: mobileye.com

[29] S. Kamath and B. Valentine, "Implementation details of mid-level vision on the embedded vision engine (EVE)," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2014, pp. 1283–1287.

[30] R. K. Satzoda and M. M. Trivedi, "On performance evaluation metrics for lane estimation," in *Proc. 22nd Int. Conf. Pattern Recognit.*, 2014, pp. 2625–2630.

[31] A. Broggi, "Parallel and local feature extraction: A real-time approach to road boundary detection," *IEEE Trans. Image Process.*, vol. 4, no. 2, pp. 217–223, Feb. 1995.

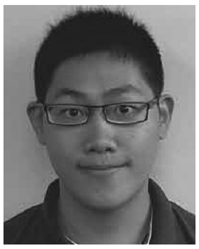
[32] S. Y. Chen, "Kalman filter for robot vision: A survey," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4409–4420, Nov. 2012.

[33] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 975–982.

[34] X. Zhang, P. Jiang, and F. Wang, "Overtaking vehicle detection using a spatio-temporal CRF," in *Proc. IEEE Intell. Veh. Symp.*, 2014, pp. 338–342.

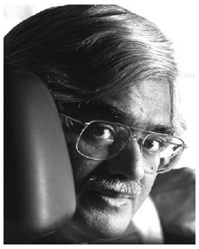
[35] R. K. Satzoda and M. M. Trivedi, "Drive analysis using vehicle dynamics and vision-based lane semantics," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 9–18, Feb. 2015.

Ravi Kumar Satzoda (M'13) received the B.Eng. (with First Class Hons.), M. Eng. (by research), and Ph.D. degrees from Nanyang Technological University, Singapore, in 2004, 2007, and 2013, respectively. He is currently a Postdoctoral Fellow at the University of California, San Diego, CA, USA, where he is associated with the Laboratory for Intelligent and Safe Automobiles. His research interests include computer vision, embedded vision systems, and intelligent vehicles.



Sean Lee is currently working toward the integrated M.S. degree at the University of California, San Diego, CA, USA. He conducts research in the field of computer vision and machine learning in the Laboratory for Intelligent and Safe Automobiles. His research interests include computer vision, machine learning, embedded programming, and intelligent vehicles.

Frankie Lu is currently working toward the integrated M.S. degree at the University of California, San Diego, CA, USA. He is currently applying computer vision and machine learning techniques for embedded computing platforms in the domain of intelligent vehicles. His research interests include computer vision, machine learning, embedded programming, and intelligent vehicles.



Mohan Manubhai Trivedi (F'08) received the B.E. (with Hons.) 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, University of California, San Diego, CA, USA, 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 analytics. He received the IEEE Intelligent Transportation Systems Society's highest honor and Outstanding Research Award in 2013.