An Efficient Industrial System for Vehicle Tyre (Tire) Detection and Text Recognition Using Deep Learning

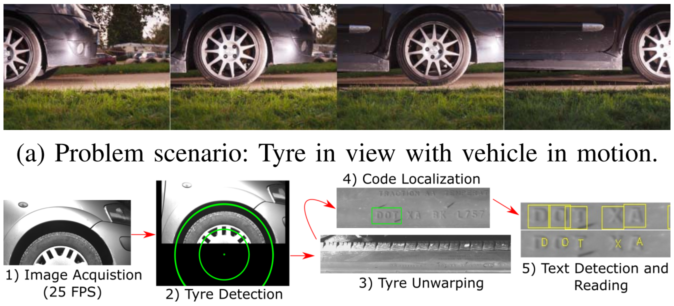
Wajahat Kazmi, Ian Nabney, George Vogiatzis, Peter Rose, and Alex Codd

***Abstract*—This paper addresses the challenge of reading low contrast text on tyre sidewall images of vehicles in motion. It presents first of its kind, a full scale industrial system which can read tyre codes when installed along driveways such as at gas stations or parking lots with vehicles driving under 10 mph. Tyre circularity is first detected using a circular Hough transform with dynamic radius detection. The detected tyre arches are then unwarped into rectangular patches. A cascade of convolutional neural network (CNN) classifiers is then applied for text recognition. Firstly, a novel proposal generator for the code localization is introduced by integrating convolutional layers producing HOG-like (Histogram of Oriented Gradients) features into a CNN. The proposals are then filtered using a deep network. After the code is localized, character detection and recognition are carried out using two separate deep CNNs. The results (accuracy, repeatability and efficiency) are impressive and show promise for the intended application.**

***Index Terms*—Intelligent vehicles, deep learning, computer vision, tyre (tire) sidewall, Optical Character Recognition (OCR).**

I. Introduction

THIS paper describes a complete system for the detection and recognition of codes printed on the tyre sidewalls (outward plane of the tyre). The codes carry information about the tyre brand, manufacturing plant and its age. For vehicle users, especially fleet operators, this information is crucial since it provides additional data for automated tyre condition monitoring. But it is a challenging task because outdoor use of tyres leads to wearing of the text due to material erosion, dust, dryness and humidity. Above all, the text has a very low contrast (black-on-black) which is at times challenging even to human eye as depicted in Figure 3. In our survey, we could not find any 2D color or grayscale image processing based solution for this problem in an outdoor setting. All the previous attempts were not encouraging enough to pursue this approach any further [1]. Therefore, all the industrial products available to read tyre codes are either 3D scanner based system for use in indoor and controlled inspection tasks [2]–[4] or hand-held laser devices for both indoor and outdoor applications [5].


(a) Problem scenario: Tyre in view with vehicle in motion.

(b) Five stage code reading pipeline.

Fig. 1. Problem scenario and the proposed solution.

For reading embossed or engraved text, lighting becomes important because the legibility of the text can be improved through shadow casting. Given bidirectional light incident at very acute angles, there is some success using low-level image processing [6]. But it is not suitable for uncontrolled outdoor situations such as depicted in Figure 1(a).

In this regard, the recent developments in deep learning-based image classification and text recognition holds promise for this problem. There has been a lot of work done in text recognition in the wild using deep Convolutional Neural Networks (CNNs) [7]–[10]. Deep CNNs are the state-of-the-art and have surpassed the traditional approaches using hand crafted features such that almost all the top-ranked results in image processing use deep learning especially for text recognition [11]. But no one seems to have addressed the problem of recognizing embossed text in the wild.

In this paper, we address this problem using a combination of standard computer vision/image processing, hand crafted features and deep learning. We propose, to the best of our knowledge, the first of its kind, tyre text reading system. Its a five stage system (Figure 1(b)) with object illumination and high frame-rate image acquisition, followed by tyre detection and text reading using deep convolutional neural networks. The system requires the vehicles to be moving under 10 mph, which is suitable for installation at gas stations, parking lots and entry/exit points of motorway toll booths. This is

Manuscript received May 2, 2019; revised August 14, 2019 and December 1, 2019; accepted December 16, 2019. This work was supported in part by the Innovate U.K. through the Knowledge Transfer Partnership (KTP) in collaboration with WheelRight Ltd., and the Department of Computer Science, Aston University, Birmingham, U.K., under Grant KTP009834. The Associate Editor for this article was D. F. Wolf. *(Corresponding author: Wajahat Kazmi.)*

Wajahat Kazmi, Peter Rose, and Alex Codd are with WheelRight Ltd., Oxford OX5 1PF, U.K. (e-mail: wajahat.kazmi@outlook.com; peter.rose@wheelright.co.uk; alex.codd@wheelright.co.uk).

Ian Nabney is with the School of Computer Science, Bristol University, Bristol BS8 1UB, U.K. (e-mail: ian.nabney@bristol.ac.uk).

George Vogiatzis is with the Department of Computer Science, Aston University, Birmingham B4 7ET, U.K. (e-mail: g.vogiatzis@aston.ac.uk).

Digital Object Identifier 10.1109/TITS.2020.2967316

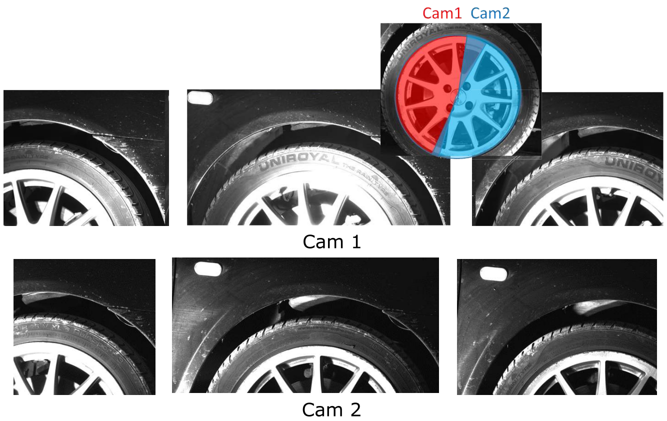


Fig. 2. Tyre radial coverage per camera and sample images.

an extension and thorough testing of our earlier work [12]. As an extension, we introduce a fully CNN based proposal generator using low level features (HOG-CNN) and compare it with HOG-MLP from our previous work as well as a well known off-the-shelf proposal generator, the Edge Box [13]. Apart from discussing the classification architectures in detail and explaining the rationale behind their design, we also do extensive testing of the repeatability of character recognition and also compare them with a standard, off-the-shelf, Optical Character Recognition engine (OCR), the Google Tesseract [14].

The structure of this article is as follows. Sections II and III describe the image acquisition setup, tyre detection and unwarping, respectively. A novel proposal generator combining hand-crafted features within a CNN architecture for code localization is described in Section IV. Code text detection and reading are described in Section V. Results are discussed in Section VI. Section VII concludes the paper.

II. Image Acquisition

The proposed setup is intended to be replicated to client sites. Imaging hardware including cameras and light sources will be installed at every site, one each for the left and the right hand side of the vehicle. With several potential sites in view, image acquisition is the most expensive bit of the entire system. Therefore, for such an industrial system, hardware cost becomes a major concern.

Keeping both the cost as well as the complexity of the task in mind, the tyre imaging was split up across a dual-camera arrangement (Figure 13). Each camera focuses on the upper half of the tyre. Combined, they ensure full radial coverage as shown in Figure 2. As the vehicle approaches the camera/light assembly, the first camera captures the first radial half of the tyre, while the second camera, the other half. This arrangement reserves the field of view for the upper half of a tyre which is in translation and rotation. It has two distinct advantages: 1) Focusing on the upper half only eases the design and installation of the light source. 2) As the target object size is reduced, cameras of half the required resolution can be used. This option is several times more economical than using one full (high) resolution camera.

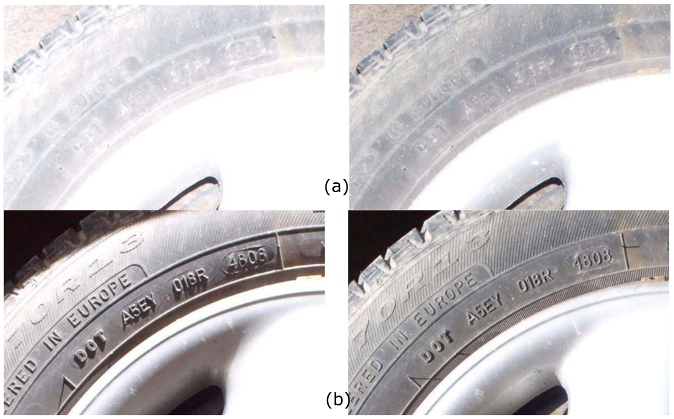


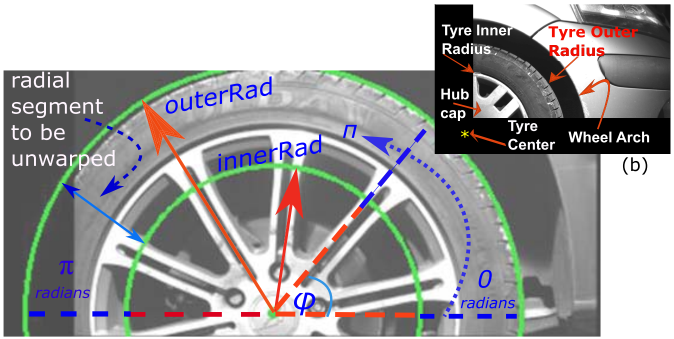
Fig. 3. Tyre sidewall with code printed. Same tyre and position with only changing illumination source position. Light angles (a) orthogonal or frontal (b) oblique w.r.t the plane of the sidewall. The breakdown of the code is: DOT (Department of Transport USA approved), A5 (manufacturing plant), EY (tyre size, manufacturer specific), 018R (manufacturer-specific batch number) and 4808 (date of manufacture in WWYY format) [16]–[18].

Figure 3 shows that strong directional lighting at an acute angle enhances the contrast and the legibility of the text. In this regard, a specially designed light-reflector assembly was developed, targeting the upper half of the tyre of a moving vehicle. The cameras are triggered when a vehicle approaches the driveway. Images are acquired at 25 FPS (frames per second), using industrial grade GigE cameras. Depending on the speed of the vehicle, generally 5 to 10 images/axle are acquired at this frame rate which ensures full radial imaging of the tyre sidewall. In order to avoid motion blur, we used global shutter cameras.

III. Tyre Detection and Unwarping

Acquired images have partial to semi sector of the tyre in the field of view as shown in Figure 2. In order to read the text, the tyres sectors must first be transformed into straight rectangular patches. For this purpose, tyre circularity is detected using Circular Hough Transform (CHT) [15] after illumination normalization. CHT is used to detect the circular junction of the hub cap and tyre (see Figure 4). But sometimes the wrong circle is detected due to some other circularity (such as a wheel arch or inner disc brake) being more dominant (greater contrast in the image due to strong strobe lights). In order to avoid this situation, all the images of each axle are processed for n radii ranges in parallel threads. The number of detected circles are collectively voted in a radius range histogram. The dominant radius i.e. the one corresponding to the bin with maximum votes is then selected as the tyre radius. Once the junction of the hub cap and tyre are detected, a second circle corresponding to the outer radius of the tyre (Figure 4(b)) is chosen at a fixed offset from the first radius. This is sufficient since the tyre code generally falls near the inner radius.

After tyre detection, the radial image patch between the inner and the outer radii is unwarped to a rectangular lattice using a Polar-to-Cartesian mapping as shown in the scheme in Figure 4(a). This not only unwarps the circularity, but also



(a)

(b)

Fig. 4. Unwarping scheme with tyre's inner and outer radii. Unwarping is done using Polar-to-Cartesian mapping.

crops out only the necessary part of the image, which reduces the computational burden on the subsequent stages.

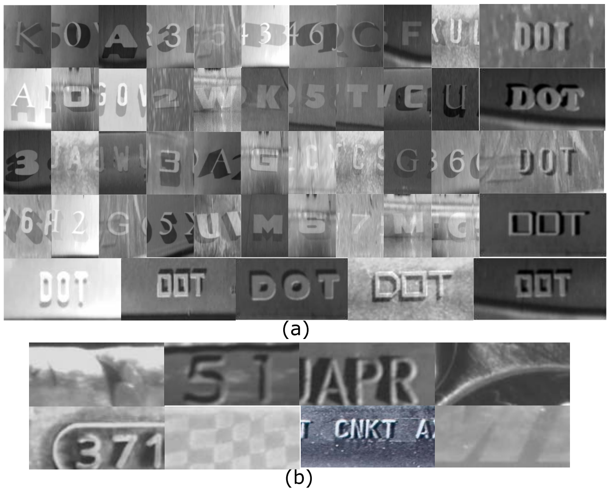
The first three steps of the pipeline, namely, image acquisition, tyre detection and unwarping were implemented in *C#*. Tyre detection and unwarping take about 500 ms/image on a 3.6 GHz Core i7 CPU.

IV. Code Detection

In this section, a machine-learning based approach for code detection and localization on the unwarped images is discussed. The character sequence *DOT* (Department Of Transport, USA) is used as an anchor. It means that the word *DOT* must first be detected to narrow down the search space as in most cases, it precedes the code. This stage has two modules in cascade i.e. proposal (region of interest) generation (Figure 6 (b)) followed by verification or code localization.

Hand-crafted features such as Histogram of Oriented Gradients (HOG) have been successfully used for text detection [19]–[21]. HOG combined with a Support Vector Machine (SVM) classifier in a sliding window manner produced reasonable results for *DOT* detection as well but given the size of the image (500 × 2000 to 4000 pixels), it takes a few minutes to scan the image. This time-scale is too long and is unacceptable for industrial applications. Ideally, we would like to have end-to-end results in less than a minute for CPU-based processing. Deep learning based object detection with fully connected layers as convolutional layers such as [22] or ROI pooling before the fully connected layers such as Faster-RCNN [23] and Mask-RCNN [24] after flexibility to process variable size input images. But our images are much bigger in size and using deep networks for proposal generation would be too costly on a CPU. It would require a large-memory GPU (12 GB or more), which increases the total system cost.

Therefore, in this paper, we propose a solution by combining hand-crafted features within a CNN-based classifier for efficiently generating proposals. We use HOG features and therefore call it HOG-CNN. Before we delve into the details, it should be mentioned here that all the CNN training runs discussed in this paper used Stochastic Gradient Descent with back propagation in Matlab using MatConvNet library [25]. The text training data was synthetically generated whereas the background class was extracted from real tyre images as shown in Figure 5. Every network used one or more 50% dropout [26] layers during the training to prevent over-fitting. Difference-of-Gaussian (DoG) filtering was applied to the input data for illumination normalization and edge enhancement.



(a)

(b)

Fig. 5. (a) Synthetically generated data (b) real tyre patches (used for background classes).

A. Synthetic Data Generation

As the task involves reading text embossed on many different tyres in varying conditions of light, weather and wear, we required a substantial amount of training data to achieve good generalization. Gathering a large and annotated dataset is a very costly and a time-consuming process. Therefore, training data was synthetically generated using several different fonts and a text rendering engine. Initially, a black and white text mask was created using various fonts in random sizes. The mask was then incrementally smeared (adding multiple copies or shifting the rendering position in a small neighbourhood. This took place in varying directions (to represent the revolving shadows) and lengths (to represent different shadow lengths). The image mask was then merged with tyre backgrounds to produce realistic embossed/engraved text images as they should appear on the tyre sidewall. Figure 5 shows a sample set.

B. Proposal Generation for Code

For proposal generation, we used HOG features as input to a CNN based classifier in two different ways. We both used a unified architecture and extracted HOG features within a CNN network (HOG-CNN) as well as extracted the features externally and interfaced them to a CNN based Multi-Layered Perceptron (MLP) appropriate for a multi-class task [27] (we call it HOG-MLP). We also compared our proposal generators with an off-the-shelf proposal generator, the Edge Box.

1) Low-Level Edge Based Proposal Generator: One of the most popular low-level proposal generators is edge boxes [13]. It uses edge maps across the three channels of a color image. Since we have grayscale images, we therefore composed a three channel image by stacking the original, a histogram

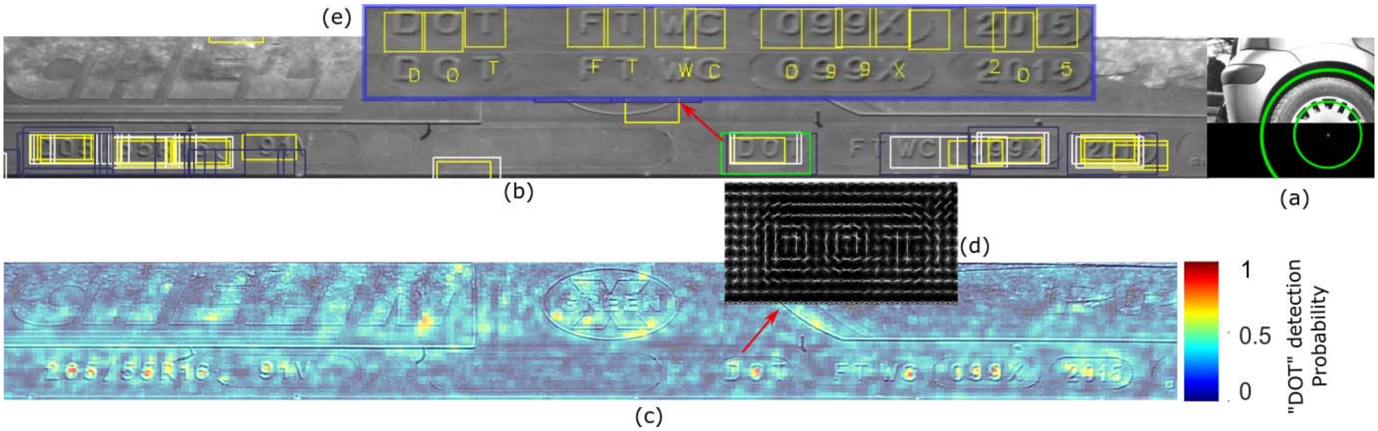


Fig. 6. (a) Tyre detected with centre outside the image boundaries. (a) Corresponding unwarped image with proposals for the code localization generated by HOG-CNN at 3 scales i.e. original (W), 1.25% (Y) and 0.75% (b). The green bounding boxes in green are detected by the code localizer network. (c) Code proposal scoremap generated at original scale. (d) Corresponding HOG feature visualization of the code anchor DOT. (e) Character detection and classification.

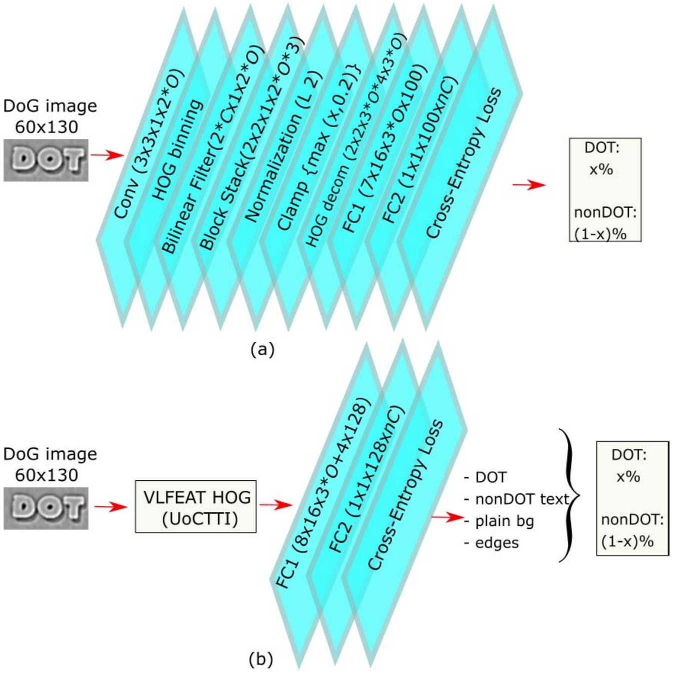


Fig. 7. Block diagram of network architectures (a) complete HOG-CNN (b) externally computed VLFEAT HOG features with a multi-class MLP (HOG-MLP). Dropout layers are omitted in the diagram. Both (a,b) employ UoCTTI method. O is the number of orientations (9 (a) and 16 (b)), C is the cell size (8×8 used in general), nC is the total number of classes. Parameter memory for these networks is 1 to 3 MB.

equalized and a Difference of Gaussian image. We empirically found out that such a combination works better than plain greyscale image. Results are shown in Figure 8(a).

2) HOG-CNN: HOG-CNN was primarily inspired by the work of [28] in which they used CNN layers to produce HOG-like features, just to invert and recreate the original images later. Instead, we plugged-in a fully convolutional network at the end of HOG layers, which makes a complete CNN architecture terminating at a cross-entropy loss layer as shown in Figure 7 (a). Such a network is shallow with fewer convolutional layers and channels than deep networks.

HOG has two widely used implementations, i.e. the original one by [29] and UoCTTI by [30]. CNN-based HOG extraction layers of [28] were numerically equivalent to [31] UoCTTI, so we used the latter. As explained by [28] for extracting HOG features using a stack of convolutional filters, a directional filter was applied in $K = 2\times$ number of orientations (O). The k^{th} directional filter is given by:

$$G_k = G_x \; u_{1k} + G_y \; u_{2k} \;\; where \;\; u_k = \begin{pmatrix} \cos \dfrac{2\pi k}{K} \\ \sin \dfrac{2\pi k}{K} \end{pmatrix}, \qquad (1)$$

and the gradients in x and y are:

$$G_y = G_x^T, \quad and \quad G_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad (2)$$

The directional filter casts the projection of the input along direction vector u_k as $g\vec{u_k}$. After directional filtering, HOG binning was performed by the following activation function:

$$h_k = ||g|| \begin{bmatrix} 1 & if \;\; g\vec{u_k} > ||g|| \cos \dfrac{\pi}{K} \\ 0 & otherwise \end{bmatrix}. \qquad (3)$$

In HOG feature extraction, the binned gradients are pooled into cells which are then combined in 2×2 blocks. This was done through a stack of linear filters. After normalization (l^2 norm), the blocks were decomposed back to the cell structure and the values were clamped at 0.2 (i.e. $\max\{x, 0.2\}$). In UoCTTI HOG, directed gradients were voted for twice the number of orientations (h_{do}) within the range $[0, 2\pi)$ along with one set of undirected gradients (h_{uo}). So a total of $3\times O$ channels were used in HOG-CNN.[1]

Hence, for an input image of $60(H) \times 130(W)$, the CNN-based HOG produced a feature map of $7 \times 16 \times 27$ for 8×8 cell size and 9 orientations. As in OverFeat [32], we added randomly initialized fully connected (FC) convolutional layers with a mask size of $7 \times 16 \times 27 \times \ldots$ (equal to the output size of the previous layer) in order to create a fully convolutional

[1] http://www.vlfeat.org/api/hog.html

network. This was followed by a 50% dropout and another FC layer as shown in Figure 7 (a). Dropout is a regularization technique which prevents overfitting through simply skipping some neurons [26].

Training such a network can be tricky as few layers were predefined while the final classifier was randomly initialized. In our case, it was trained on the dataset (more than 500K images in total with DOT class synthetically generated). Training set contained a DOT and a background class containing a mixture of non DOT text, edges and plain backgrounds. A total of 80-90 training epochs were deemed sufficient. Since the network is shallow with sparse filters, it could be efficiently trained even on a CPU.

3) HOG-MLP: For HOG-MLP, HOG features were extracted using VLFEAT library [31] and were fed into a CNN-based multi-class MLP (HOG-MLP) (Figure 7 (b)).

In VLFEAT HOG, gradients are binned for $3 \times O + 4$ texture components [33]. Therefore, for an input image size of $60(H) \times 130(W)$, an 8×8 HOG cell size and 12 orientations (40 components), the first layer in the network was $8{\times}16 \times 40{\times}\ldots$. The cell size and the number of orientations were chosen to achieve best possible detection accuracy. It was trained on a 11 class ($nC = 11$) dataset of more than a million images containing 7 synthesized DOT classes for round/square/thin and broad fonts, clear and diffused appearance, long and short shadows, single and double spacing between the characters etc along with 4 background classes divided among plain backgrounds and textures.

A second HOG-MLP proposal generator was constructed with cell size $= 8\times$ 8, $O = 16$ (making up a total of 52 components), $nC = 4$ (i.e. DOT, plain background, edge/texture, non-DOT text) shown in Figure 7 (c). The outputs of both the HOG-MLPs were mapped to a binary classification (DOT/non-DOT). For both of these networks, satisfactory results were obtained after training for 30-50 epochs. Just like HOG-CNN, these sparse networks could also be efficiently trained efficiently on a CPU.

Comparison: In order to detect variations in the perceived font sizes either due to change in the engraved font size or distance between the car and the camera, images were scanned at three scales (1.25, the original size and 0.75) for proposal generation. The non-maximum overlapping bounding boxes of the proposals were suppressed (NMS) using box area intersection-to-union ratio compared to a fixed threshold.[2] The filtered proposals were then passed onto the next stage of the cascade to finally localize the code and reject the false positives.

As the text was of very low contrast, for proposal generation, low-level feature-based approaches such as Edge Boxes were found to be unsuitable. The reason was that the strong edges from other segments of the tyre with or without text, as shown in Figure 8 (a), usually dominate.

It was observed that the machine learning based proposal generators were comparable though HOG-CNN was slightly better generating fewer proposals and hence generalized the data more than HOG-MLP 10-way classifier (Figure 8 (b,c)). Both HOG-CNN and HOG-MLP have a very high recall rate (100% in Figure 17). Figure 8 (d) presents the best proposal generator with HOG-MLP ($O = 16$, $nC = 4$) and a 4 way classification with the least false positives. Though with many closely matching classes, HOG-MLP was simpler to train. A sample of training classes is shown in Figure 10.

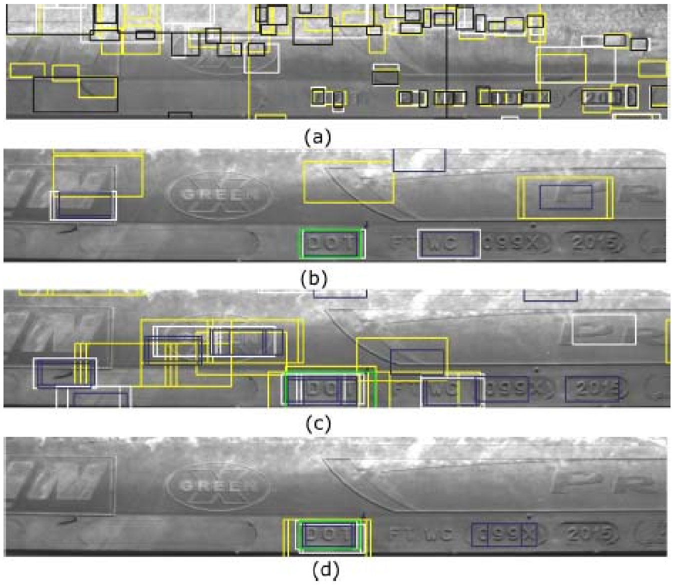


Fig. 8. Comparison of code proposal generation. (a) Edge Boxes [13]. (b) HOG-CNN binary classifier. (c) HOG-MLP 10-way classifier mapped to a binary output ($O = 12$). (d) HOG-MLP 4 way classifier mapped to a binary output ($O = 16$). For color code of the boxes, refer to Figure, 6.

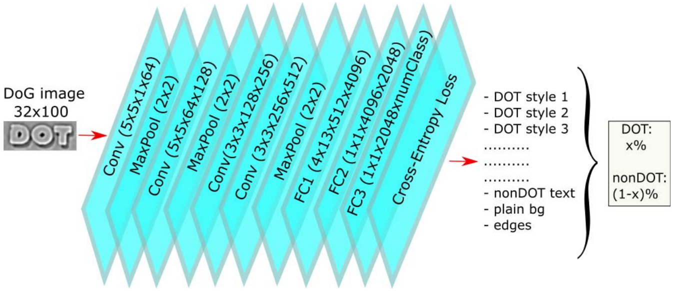


Fig. 9. Code localization network architecture. Every convolution layer was followed by a rectified liner unit (ReLU) layer as well as a 50% dropout layer from the 4th convolution layer onwards. Parameter memory 496 MB.

On comparison between the three proposal generation approaches, the scan times by the HOG-CNN and HOG-MLP for an image of 500 × 3000 pixels) were around 550 and 250 ms respectively on an Intel Core i7 3.6 GHz CPU (see Figure 14), whereas by using for the edge box approach, the code shared by the authors [3] took just under 7 secs to generate proposals over three scales (Figure 8(a)).

C. Code Localization

To finally localize the code from the filtered code proposals, a deep network similar to [7]'s 90K dictionary network

[2] Tomasz Malisiewicz https://github.com/quantombone/exemplarsvm/tree/master/internal

[3] https://github.com/pdollar/edges

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

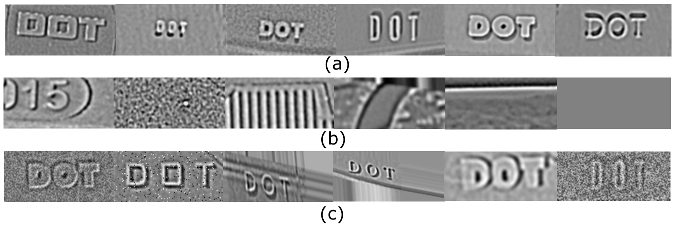


Fig. 10. Sample classes used in the code proposal generator. DoG Images (a) DOT classes based on font and sizes (b) background classes (c) random noise, blurring and affine deformations injected during training.

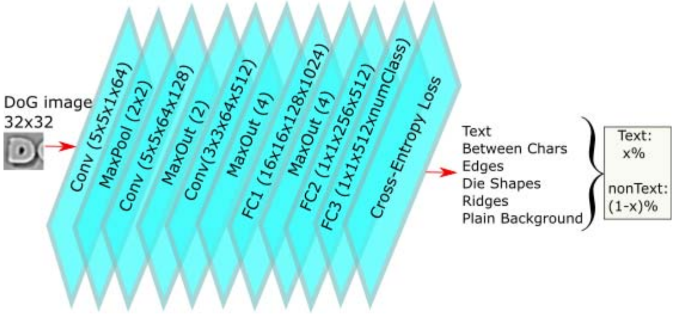


Fig. 11. Architecture of text detector network. Every MaxOut layer was followed by a 50% dropout layer.

(Figure 9) was used. Training set for this network contained multiple DOT and background classes (800K images in 10 classes: 7 DOT classes from Section IV-B.3 and 3 background classes for plain background, edges/texture and non-DOT text). The classification results were then mapped to a binary output. As a result, a lot of false positives among the proposals were being rejected and only a few strong candidates were retained (the green boxes in Figures 6(b) and 8). False positives seeping through at this stage (non-DOT green boxes) were addressed through text recognition, the subsequent stages of the cascade.

V. Code Reading

Code reading consists of two stages, text detection and recognition. The code patch of the image is first pre-processed to crop it down to the text height using low-level filtering. Bilateral filtering is done optionally in order to smooth out any unwanted background texture. Then the patch height is resized to 40-50 pixels in accordance with the text detection network's stride (number of pixels skipped between to consecutive detection windows on the input image).

A. Text Detection

The code characters are detected using the network shown in Figure 11. Since the text has very low contrast with respect to the background, a dense prediction mechanism is required. In fully convolutional networks, max-pooling layers downsample the image which increases the network stride. Removing max pooling layers will allow dense (pixel by pixel) predictions but will enormously increase the parameters space which will have its toll both on the efficiency and accuracy.

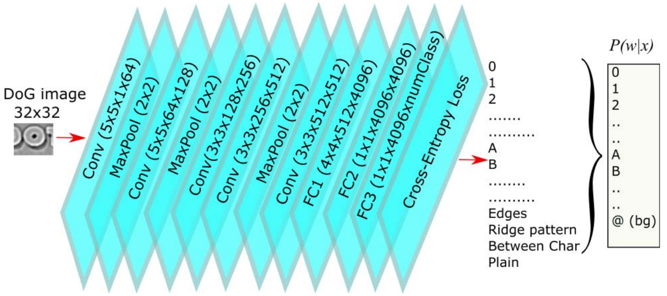


Fig. 12. Architecture of Character recognition network. Every Convolution layer was followed by a ReLU layer. From the 2nd Conv layer onwards, a 50% dropout layer was added after every conv layer. *P(w|x)* represents the probability of each class and *@(bg)* is a superset of all background classes.

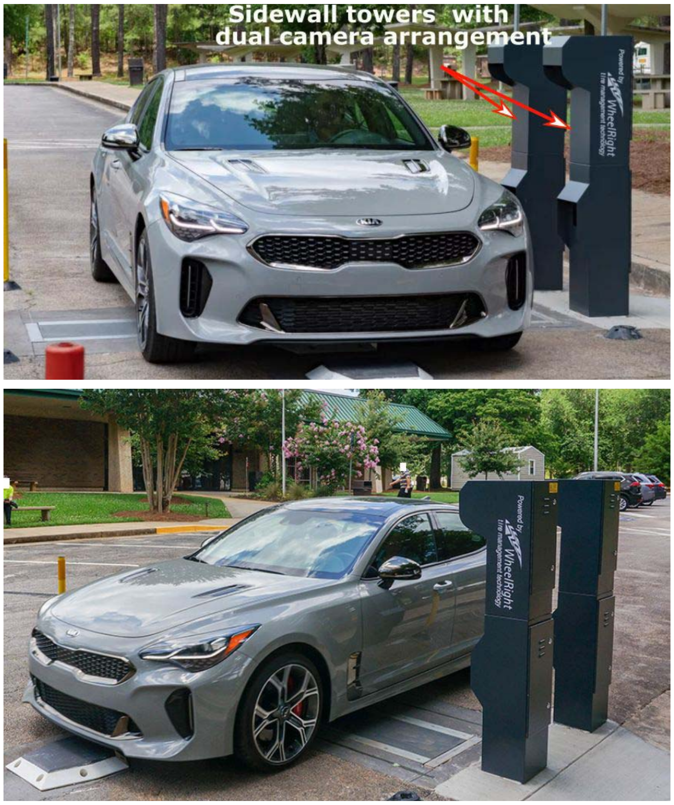


Fig. 13. Sidewall text reader with dual camera/light housing towers in operation (courtesy CNET.com)).

Regularization techniques such as DropOuts in combination with MaxOut activations are helpful in improving the accuracy [34], [35]. Therefore, as shown in Figure 11, Maxout layers were used in this architecture. We observed that if a ReLU precedes Maxout layers, the network converges quickly to a minimum.

Training was done on a 700K image dataset with text class synthetically generated (section IV-A). The background class was extracted from actual tyre patches. It contained simple edges, ridge patterns, cast or die shapes (sometimes used to emboss text on tyres) and a plain background. The output was mapped to a binary class probability i.e. Text/non-Text. The text detector produced bounding boxes centered around regions with the highest probabilities of text being present.

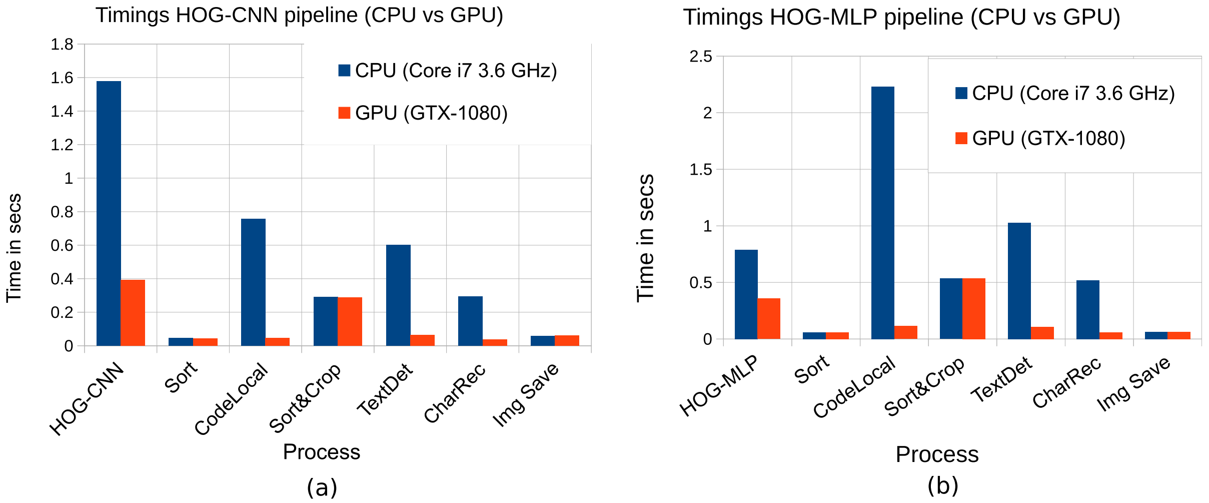


Fig. 14. Timings comparison (CPU:Corei7 4790 3.6 GHz 16 GB RAM vs GPU:GTX-1080 8GB GPU RAM) for obtaining end-to-end results from unwarped images. Shown are the average times of 50 images ranging in size from 500 × 1600 to 500 × 4500 (depending on the tyre size and position) with an average size of 500 × 3000. NOTE: Image acquisition, tyre detection and unwarping time is not included (which is approx. 500 ms/image). Scanning a 500 × 3000 image with the deep code localizer network only (section IV-C) takes more than 20 ms on the CPU. (a) HOG-CNN end-to-end network trained with back propagation (total time CPU: 3.8 secs/image, GPU: 1.0 secs/image) (b) HOG-MLP (O=12, nC=10) with externally extracted HOG features (total time CPU: 5.15 secs/image, GPU: 1.2 secs/image). In GPU mode, only the CNN networks are processed on the GPU.

Non-maxima suppression was applied (section IV-C) to the detected boxes to filter down the proposals. We could have used a character classifier for text detection as well. But, in our experiments, we observed that a dedicated classifier for text detection performed better.

B. Character Recognition

Detected text locations were used to crop characters which were then fed into a character classifier network based on [7]'s 90K dictionary network as shown in Figure 12. This network has classes for numerals 0-9, capital alphabets A-Z (excluding I, Q, S and O which are not used in the tyre codes) and seven background classes, making a 39-way classifier which was mapped to 33 classes (32 character and 1 background class). The model was trained on our synthetic character dataset of around 700K images.

VI. Results and Discussion

As this is an industrial system, both accuracy and efficiency are important. We will discuss both in detail.

A. Experimental Setup

Data was collected from the complete system installed at one of the sites as shown in Figure 13. The data collection was not planned. Rather in the absence of a benchmark to compare against, a subset of images representative of the weather and the tyre conditions was shortlisted and processed for assessment. As it can be observed in the figure, the installation imposes certain restrictions on movement of the vehicles, such as their speed and orientation w.r.t to the cameras. This not just helps in improving the overall performance of system but also reduces the cost of the cameras/light units by limiting the speeds to under 10 mph. Higher vehicle speeds are not an impediment for the algorithm but rather for the image acquisition system and hence would raise the cost of the camera units by an order of a magnitude. The algorithm is designed to process still images without significant motion blur. Once such images are acquired, the sidewall text reader should be able to process the images as usual.

B. Accuracy

Accuracy is dependent on the data sample being analysed. The training error of every classifier in the cascade was under 5%. But since the training data is in part synthetic, even with regularization, models may still tend to overfit which in turn compromises the performance on real images. To some extent this tendency has been avoided by injecting random noise and affine deformations during training as shown in Figure 10 (c).

As argued before, in the absence of a benchmark for tyre text recognition, it is difficult to quantify accuracy which is subjected to light conditions, weather (dry/wet), object's (tyre's) condition / material / age / wear & tear. We therefore assessed the accuracy on nine representative images which depict very well the possible situations. Figure 17 (a) show increasing complexity of text legibility from images 1 to 6. Images 7 to 9 are even difficult for human observers. Figure 17 (b) shows the accuracy graph which is calculated for every code image as:

Accuracy = *{Total number of characters - Number of misclassifications (including background detected as text)}/ Total number of characters.* —— (eq. 1)

The code proposal generator is less likely to miss any region containing characters *DOT* (100% recall), as it responds strongly to a central *O* (Figure 6 (c)), especially with scanning done at three scales. Therefore, *DOT* has been successfully detected in all of the sample images in Figure 17 (100% recall)

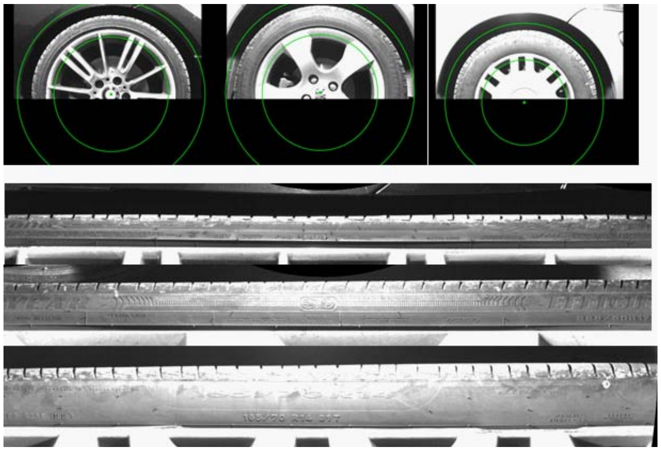


Fig. 15. Tyre detection results: Different vehicles with varying height and tyre radius and centre both inside/outside the image frame.

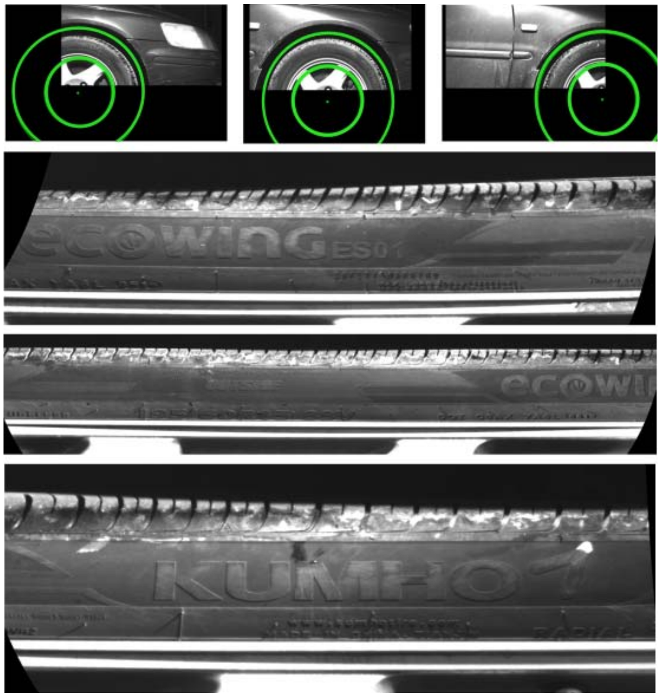


Fig. 16. Tyre detection results: Same vehicle travelling across the field of view with tyre centre outside the image frame (processing time 500 ms/image on Corei7 3.6 GHz CPU).

and thus this makes the code detection, a fairly robust part of the cascade. Text detection and recognition, on the other hand may suffer because of the above mentioned conditions. From 1 to 5, the text recognition accuracy is 80% or above which includes very tiny font (image 1), damp (image 3) and dark (image 4). Image 5 is an extremely dark tyre with poor contrast and texture. Therefore, number *0* and *8* were misclassified as *D* and *3* along with *J* as *I* in *2JFR* (still 80% accuracy). In image 6, the characters are generally diffused with the last half of the code segment badly effaced or rubbed off. Even then, only two date digits were misread (*1613* as *121_*) along with two ghost detections (background as *Y* and die shape as *C*) and an overall accuracy of 73% was achieved.

Figure 17 (a) images 7 to 9 show situations in which the text legibility is reduced due to rain water (image 7 and 9) creating undesired reflections and scintillation off the tyre surface or muddy water splash (image 8) creating unwanted texture, both of which change the appearance of the text. Deep networks for the text recognition can be improved by including such a data in the training but it may compromise the performance on good text. Therefore, in these cases, we do not attribute the error in text detection/misclassification to be a fault of the text recognition system. The appearance can vary within a large variance depending on the amount of water on the tyre and the angle of the text w.r.t the light source. In these examples, the accuracy varies from 73% down to 14% which is quite understandable as, for example, code image 9 even beats the human eye. Due to this increased unpredictability, we mark such cases difficult.

As now we have an estimate of accuracy, lets assess the repeatability of the system on legible (non-difficult) cases.

C. Repeatability

Another reliable measure of performance in such cases is repeatability. This means, how consistent are the results if the same tyre passes in front of the system multiple times. Figure 18 shows such a code reading repeatability test. Please note that based on the availability of drivers/vehicles, the number of drive-overs (driving past the imaging system) is not consistent. The drive-overs took place in uncontrolled manner on different dates with varying light/weather conditions and driving patterns. Each drive-over contributes one code patch displayed in the figure. Sub-figures (a, b, c, d) show fairly consistent and accurate detections across the drive-overs and only when the tyre moves away from the light source (b: last three detections) the text diffuses into the background, producing text detection and classification errors. (d) is a case with tiny font as in Figure 17 (b). However, (e) and then particularly (f) are tyre examples with strong background textures producing ghost detections (detecting characters that are not actually there) as well as missing or misclassifying the detected ones. Mean accuracies per drive-over are displayed in (g). These tests show an average accuracy of 86%. Drive-over accuracy is calculated as per eq. (1) in Section VI-B.

Looking at these figures, we can infer that owing to the contrast reduction and/or increase in the background texture, the performance does suffer. But the performance still seems quite robust and repeatable given such a challenging task. Further tuning the same classifiers for addressing some of the problems such as characters with little separation will make it respond to unwanted shadows and shapes. However, there is still room for improvement since a single wrong reading of a character can mislead the brand, size or date of manufacturing to a widely different one. One major source of such an error is a fixed window size for text detection as the unwanted background texture may prevent accurate text height estimation. In order to address this problem, bounding box regression techniques and end-to-end training of both text detector and character recognizer will be required [8], [10].

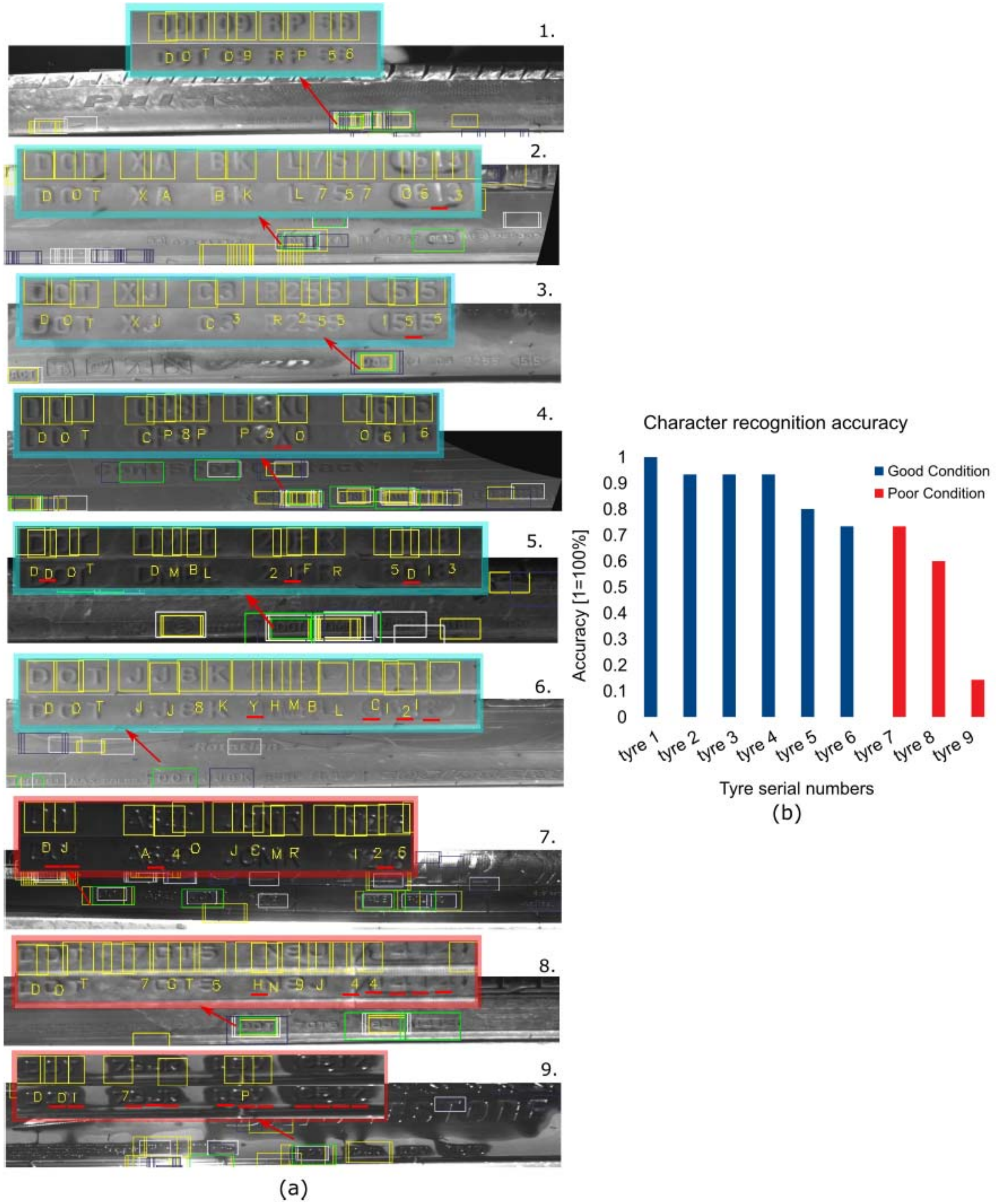


Fig. 17. (a) Nine different unwarped tyre images with varying degree of contrast and complexity. HOG-MLP proposals are represented in white, yellow and blue boxes for three scales i.e. original size, 1.25% and 0.75%. Code boxes verified by deep network are in green. 1) Tiny font size but a clean tyre [DOT 09 RP 56]. 2) Clean tyre with low background texture [DOT XA BK L757 06*1*3]. 3) Damp but clean tyre with low background texture [DOT XJ C3 R255 15*1*5]. 4) Low background texture with clean but dark surface [DOT CP8P P3*X*O 0616]. 5) Very dark tyre rubber with noticeable background texture [D D OT DMBL 2*J*FR 5*0*13]. 6) Worn out tyre with effaced or rubbed off characters DOT JJ8K Y HMBL C 1*6*1*3*. 7&9) Soaked wet tyres D*OT* A*5*40 JCMR 12*1*6 & D*OT* 7*GJR R3V 0517*. 8) With muddy water stains adding a strong texture [DOT 7GT5 H N9J 4 *1414*]. NOTE: Characters in 🟨 are background misclassified as text. Missed or misclassified characters appear in *italics*. (b) Corresponding character recognition accuracies. (c) Comparison between CPU and GPU processing times (Core i7 3.6 GHz vs GTX-1080).

D. Tyre Manufacturer Information Retrieval

Tyre DOT codes are allotted by USA Department of Transport (D.O.T) and it spans across the tyre manufacturers of the entire globe. Only D.O.T has a complete database, which can be checked for individual codes [37]. Some freelancers have accumulated a DOT code database as well, which provides a better overview. Following the breakdown of DOT code as described in the caption of Figure 3 (b), only the first two characters and then the date part is important and can be tracked from a central database. Other characters are

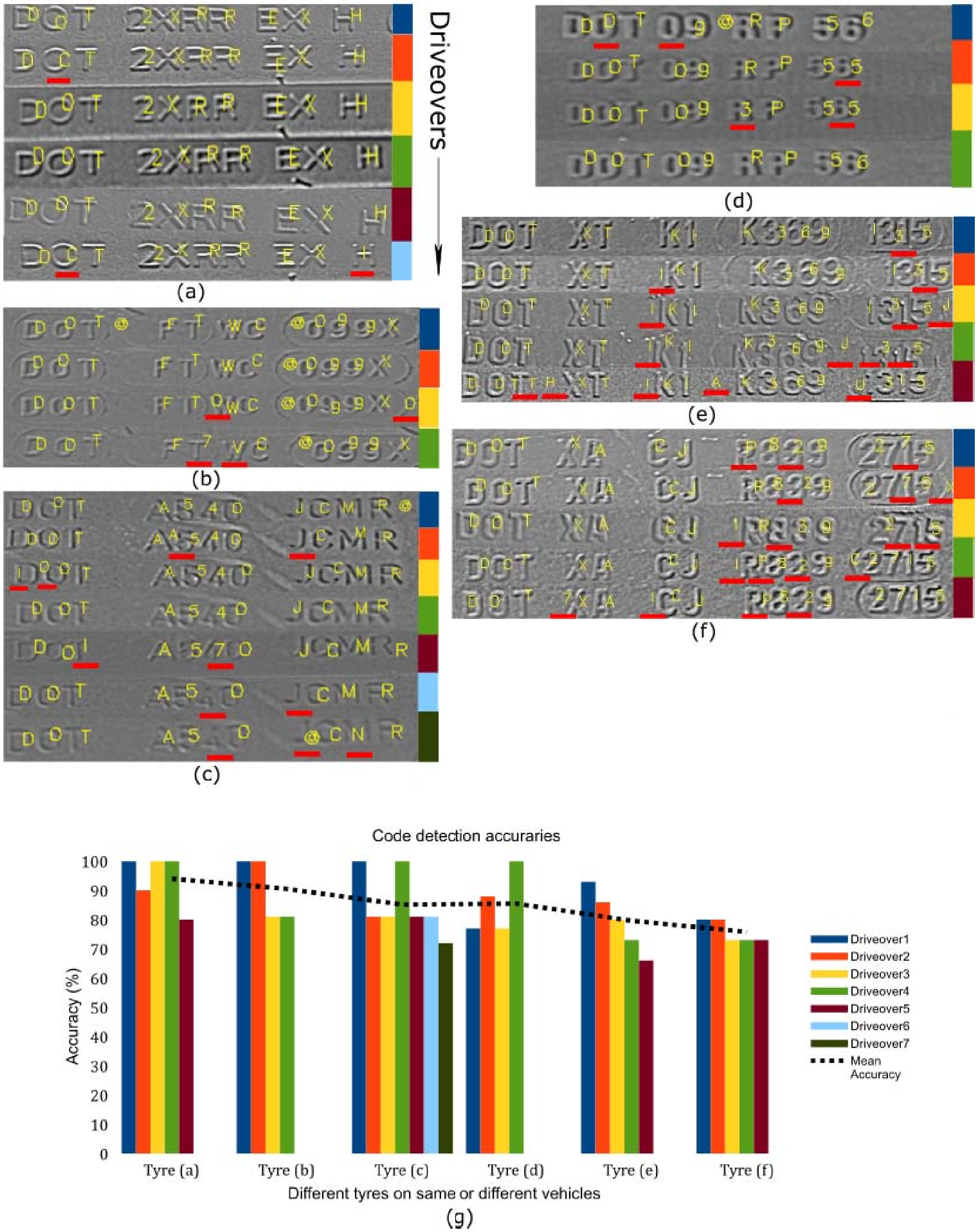


Fig. 18. Code reading repeatability assessment with six different tyres driving past the system multiple times (driveovers). (a) Clear text in each driveover (mean accuracy 95%). (b) Strong shadows can also mislead character recognition and background texture can produce ghost detections (mean accuracy 90%). (c) Vehicle gradually moving away from the light source diminishing the contrast which leads to missed characters (mean accuracy 85%). (d) Tiny font size and narrow spacing produces challenge for text recognition networks (mean accuracy 86%). (e,f) Increasing background texture poses problem for text recognition for such a low contrast text (mean accuracies 80% and 78%, respectively). Average accuracy of the 6 sets is 86%. NOTE: DoG images. Incorrect detections/classifications are underlined in red whereas @ sign denotes background class.

manufacturer specific. The first two characters carry detailed information about the manufacturing plant of the tyre. We used the predictions in Figure 17 with Harringer DOT code database [36] to assess the accuracy of the information on the tyre level. The results are reported in Table I. From among the 9 tyres in Figure 17 (a), 7 manufacturers and their plants were correctly identified. Only 2 among the three difficult cases marked red in the figure could not be retrieved correctly.

TABLE I

DOT Code Information of Tyres in Figure 17 (a) as Retrieved by Our System From [36]. Out of 9, Only 2 Tyre Manufacturers Were Identified Wrong (Tyre Number 7. and 9. in the Figure) Which Are Already Marked as Difficult Cases

Ground Truth		**Predicted**	
DOT letters	Manufacturer	Detected code	Retrieved info.
XA	Pirelli	XA	Pirelli, Bollate, Milan, Italy
XJ	Pirelli	XJ	Turk Pirelli, Kosekoy, Turkey
CP	Continental	CP	Continental, A.G, Korbach, Germany
DM	Dunlop	DM	Dunlop, Germany
JJ	Nexen	JJ	Nexen Tyre Co. Quindao, China
A5	Debica	A4	Hung-A Co., Poland
7G	Bridgestone	7G	BridgestoneFirestone, Poznan, Poland
7G	Bridgestone	7–	*–incomplete code detected–*

E. Comparison With OCR

Optical Character Recognition (OCR) generally performs well when there is a clear contrast against the background, such as scanned images of documents, for example, black over white text. In case of vehicle tyres, as argued earlier and shown in Figure 3 (a) (b), light angles are important in order to enhance the text against its similar background by projecting shadows. Even after this, the background only diminishes slightly (see Figures 3 (b)) which is not a sufficient contrast for OCR. Our experiments with standard OCR, such as Google's Tesseract OCR engine, produced very discouraging results as either a few incorrect characters or nothing at all was detected. Pre-processing of images to completely remove the background such as proposed by Panetta [6] is neither practical (requires strobe to flash at 90^o to the plane of the sidewall) nor useful (results are not stable or repeatable).

F. Efficiency

For an industrial system, with an end user waiting for results, efficiency is crucial. GPUs (Graphical Processing Units) are extensively being used in deep learning-based systems. But deploying GPUs means scaling up the total system cost as well as its complexity as extra power may be required while operation under hot weather may still be another hurdle. With an increasing demand and every site requiring two units (one each for the right and the left hand side of the vehicle), keeping the overall cost low is a key attribute. Thus a CPU based system is ideally sought. It can be observed in Figure 6 that the interesting part of the tyre is a relatively small segment of the image. Scanning the entire unwarped image (average size 500 × 3000 pixels) with the deep network of Figure 9 takes more than 20 secs on a Corei7 3.6 GhZ CPU (required parameter memory 496 MB), which is sought by networks proposed by [8], [10], [38]. Our cascade of HOG-CNN (required parameter memory 1 to 3 MB) followed by a deep scan of proposals thus generated reduced this total time to around 3 sec (see Figure 14). It is an improvement by an order of magnitude in terms of efficiency (almost 95% speedup), as well as a significant reduction in the total system cost and complexity, without any apparent compromise on the accuracy. With this, the end-to-end results for processing an image for tyre detection and unwarping (C#), and then scanning a resultant 500 × 3000 pixel unwarped image at three different scales followed by detecting and reading the code (MATLAB) takes on average 3 to 5 secs on the above mentioned CPU. On a GTX-1080 GPU, this time is between 1 to 2 secs (see Figure 14).

VII. Conclusion

In this paper, we presented a complete pipeline for detecting and reading tyre codes of a moving vehicle using roadside cameras. The article also presented a novel technique for efficient proposal generation by combining HOG with CNN based classifier. Using state-of-the-art deep learning models and fully convolutional networks, a robust and efficient architecture was presented. Although, in the given problem, there is no benchmark to compare the performance against, the image results show that it is quite effective and accurate. There is still room for further improvement, especially in the text detector. Making it robust to both weak characters as well as for closely spaced fonts will improve the over all accuracy of the system. Other aspects for further investigation are multi-scale text detection tied to a bounding box regressor and a separate date classifier within an end-to-end framework than in a cascade.

Acknowledgment

The authors would like to thank the role and cooperation of the staff from WheelRight Ltd., and the Department of Computer Science, Aston University, Birmingham, U.K., especially Dr. S. Wong, Ms. J. Freedman, and M. May.

References

[1] T. Wahdan, G. A. Abandah, A. Seyam, and A. Awwad, "Tire type recognition through treads pattern recognition and DOT code OCR," *Ubiquitous Comput. Commun. J.*, vol. 9, no. 3, 1992.

[2] MicroEpsilon. (2017). *Reading the DOT Code on Tyres*. [Online]. Available: https://www.micro-epsilon.co.uk/applications/areas/Profil/DOT-Nummer/

[3] Cognex. (2018). *Tire Code Reading*. [Online]. Available: https://www.cognex.com/products/machine-vision/3d

[4] Numetrix. (2018). *Tire Manufacturing Applications*. [Online]. Available: http://www.numetrix.ca/tires-inspection/

[5] T. Roger and S. Cesare, "System and method for reading a tire code and obtaining tire-related information," WO Patent 2017 074759 A1, May 4, 2017. [Online]. Available: https://patents.google.com/patent/WO2017074759A1

[6] R. Panetta, "Method and apparatus for identifying embossed characters," U.S. Patent 20110150346 A1, Jun. 23, 2011. [Online]. Available: http://www.google.com/patents/US20110150346

[7] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 1–20, Jan. 2016, doi: 10.1007/s11263-015-0823-z.

[8] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2315–2324.

[9] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017.

[10] X. Zhou *et al.*, "EAST: An efficient and accurate scene text detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2642–2651.

[11] ICDAR. (2017). *Robust Reading Comptetion*. [Online]. Available: http://rrc.cvc.uab.es/?ch=4&com=evaluation&task=1

[12] W. Kazmi, I. Nabney, G. Vogiatzis, P. Rose, and A. Codd, "Vehicle tire (tyre) detection and text recognition using deep learning," in *Proc. 15th Int. Conf. Autom. Sci. Eng.*, Vancouver, BC, Canada, Aug. 2019.

[13] L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/edge-boxes-locatin%g-object-proposals-from-edges/

[14] (2019). *Google Tesseract (OCR) Engine*. [Online]. Available: https://opensource.google.com/projects/tesseract

[15] Wikipedia. (2018). *Circular Hough Transform*. [Online]. Available: https://en.wikipedia.org/wiki/Circle_Hough_Transform

[16] Pirelli. (2017). *Tyre Markings*. [Online]. Available: https://www.pirelli.com/tires/en-us/car-light-truck/find-your-tires/ti%re-use-guide-warranty/tire-marking

[17] Mitchelin. (2018). *Tyre Sidewall Information*. [Online]. Available: http://www.michelinrvtires.com/tires/tires-101/tire-basics/how-to-read%-dot-identification/

[18] Toyo. (2018). *Tyre Sidewall Information*. [Online]. Available: https://www.toyotires.com/tires-101/reading-a-sidewall

[19] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Washington, DC, USA, 2011, pp. 1457–1464, doi: 10.1109/ICCV.2011.6126402.

[20] A. Mishra, K. Alahari, and C. V. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2687–2694.

[21] A. Mishra, K. Alahari, and C. V. Jawahar, "Image retrieval using textual cues," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 3040–3047.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[23] R. B. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Washington, DC, USA, 2015, pp. 1440–1448. [Online]. Available: http://arxiv.org/abs/1504.08083

[24] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[25] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. ACM Int. Conf. Multimedia*, 2015.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[27] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). New York, NJ, USA: Springer-Verlag, 2006.

[28] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1. Washington, DC, USA, Jun. 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[31] A. Vedaldi and B. Fulkerson. (2008). *VLFeat: An Open and Portable Library of Computer Vision Algorithms, Version (0.9.16)*, [Online]. Available: http://www.vlfeat.org/

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2014.

[33] A. Vedaldi. (2017). VLFeat HOG Implementation Details. VLFEAT Library. [Online]. Available: http://www.vlfeat.org/api/hog.html

[34] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. 30th Int. Conf. Int. Conf. Mach. Learn. (ICML)*, vol. 28, 2013, pp. III-1319–III-1327. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3043084

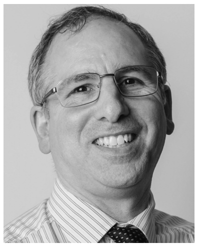
[35] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *Proc. Eur. Conf. Comput. Vis.*, 2014.

[36] *Harringer Tyre D.O.T Code Database*. Accessed: Aug. 23, 2017. [Online]. Available: http://www.harriger.com/

[37] *National Highway Safety Administration, DOT Code Database Search*. Accessed: Dec. 11, 2018. [Online]. Available: https://vpic.nhtsa.dot.gov/mid/

[38] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. New York, NY, USA: Curran Associates, 2015, pp. 91–99.

Wajahat Kazmi received the Ph.D. degree in computer vision from Aalborg University, Denmark, in 2014. Since 2015, he remained engaged with Aston University, Birmingham, and WheelRight Ltd., Oxford, U.K., for designing a tyre sidewall text reader. His research interests span across the application domain of computer vision and deep learning, especially in intelligent transport systems and advanced driver assistance systems.



Ian Nabney received the Ph.D. degree in mathematics from the University of Cambridge. He is currently the Head of the School of Computer Science, Electrical and Electronic Engineering, and Engineering Mathematics, University of Bristol. He has written more than 100 articles in machine learning and systems analytics. He is also an Architect of the Netlab pattern analysis toolbox.

George Vogiatzis received the Ph.D. degree from the Machine Intelligence Laboratory, University of Cambridge. Prior to that, he was a Senior Research Scientist with the Toshiba Research Laboratory, Cambridge, U.K., where he was a member of the Computer Vision Group. He is currently a Computer Vision Researcher and employed as a Senior Lecturer with Aston University, Birmingham, U.K. His main research interests lie in the interface between computer vision and graphics, including shape-from-X.

Peter Rose received the B.Eng. degree in mechanical engineering from the Imperial College, London, and the M.Sc. degree in design of rotating machines from Cranfield University. He is currently a Team Leader of mechanical and control systems engineering with WheelRight Ltd., Oxford, U.K.

Alex Codd is currently an Engineering and Operations Manager with WheelRight Ltd., Oxford, U.K. He has led the development of technology for automated tyre inspection since 2012, including machine vision projects alongside pressure measurement. He is also a Master's Graduate of electronic systems engineering from Kingston University, a Chartered Engineer, and a member of the IET.