An industrial system for vehicle tyre detection and text recognition using a pipeline of conventional image processing and deep learning

Wajahat Kazmi[1], Ian Nabney, [2], George Vogiatzis [3], Peter Rose [1] and Alexander Codd[1]

***Abstract*— This paper presents an industrial system to read text on tyre sidewalls. Images of vehicle tyres in motion are acquired using roadside cameras. Firstly, the tyre circularity is detected using Circular Hough Transform (CHT) with dynamic radius detection. The tyre is then unwarped into a rectangular patch and a cascade of convolutional neural network (CNN) classifiers is applied for text recognition. We introduce a novel proposal generator for localizing the tyre code by combining Histogram of Oriented Gradients (HOG) with 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 end-to-end system presents impressive accuracy and efficiency proving its suitability for the intended industrial application.**

I. Introduction

Tyre size, brand, model, age and condition monitoring are critical for many vehicle users, especially fleet operators. This article describes an end-to-end system for the detection and recognition of tyre codes printed on the tyre sidewalls. This is a challenging task, because outdoor use of tyres leads to wearing of the text due to material erosion, dust, dryness and humidity. Furthermore, the text has a very low contrast (black-on-black) which is at times challenging even to a human eye as shown in Figure 2.

Industrial products available to read tyre codes are either hand-held laser devices [1] or 3D scanner-based systems for use in indoor and controlled inspection tasks [2]. To the best of authors' knowledge, all the previous attempts at 2D color or grey scale image processing based solutions were not encouraging enough to pursue the adopted approaches any further [3]. In this regard, the recent developments in deep learning based image classification and text recognition hold promise. There has been a lot of work done in text recognition in the wild using deep Convolutional Neural Networks (CNNs) [4], [5], [6], [7]. 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 works employ deep learning especially for text recognition

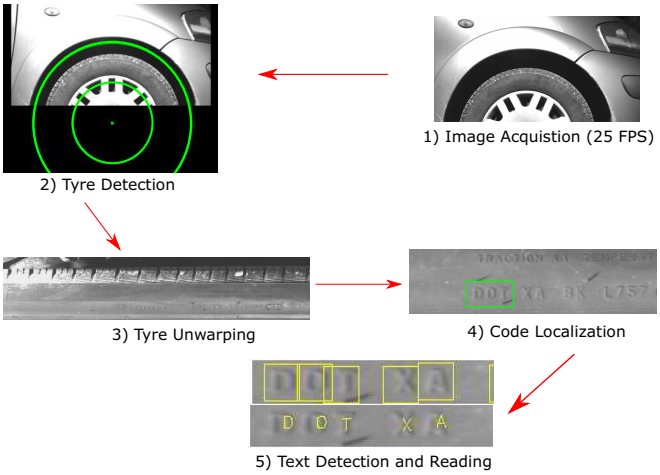

Fig. 1. **Five step tyre code reading pipeline**

[8]. But no one seems to have addressed the problem of recognizing embossed text in the wild i.e. text with no contrast w.r.t background but only a subtle clearance from the surface. In this case, lighting becomes important because the legibility of the text is achieved through the casting of the shadows. There is some work which uses low-level image processing given bidirectional light incident at very acute angles [9], but it is not suitable for uncontrolled outdoor situations such as depicted in Figure 1. Our early experiments with standard Optical Character Recognition methods (OCR) produced very poor results, so a more sophisticated approach was needed. Therefore, in this paper, we propose a complete five-step system (Figure 1) with object illumination, high frame-rate image acquisition, followed by tyre detection and text reading using deep artificial neural networks.

The structure of this article is as follows. Sections II and III describe the image acquisition setup and tyre detection/unwarping, respectively. A novel proposal generator combining hand-crafted features with a CNN network 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

Since a twin imaging setup (a separate imaging system each for the left and the right hand side of vehicle) is required for every site and with several potential sites in view, keeping the overall cost of the system low is a major concern. High resolution industrial cameras can be very expensive. Therefore, to ensure minimal required image resolution while

*This research work was funded in part by Innovate UK through Knowledge Transfer Partnership (KTP) Grant No. KTP009834 in collaboration with WheelRight Ltd. and Aston University, Birmingham, UK, Department of Computer Science. Authors greatly acknowledge the role and cooperation of the staff from both the institutes, especially Dr. Sylvia Wong, Ms. Jose Freedman and Mr. Martin May.

[1]WheelRight Ltd. Begbroke Science Park, Oxford, UK, OX5 1PF Emails: wajahat.kazmi@outlook.com, {Peter.Rose & Alex.Codd}@Wheelright.co.uk

[2]Bristol University, School of Computer Science, Bristol, BS8 1UB, UK Email: ian.nabney@bristol.ac.uk

[3]Aston University, Department of Computer Science, Birmingham, B4 7ET, UK Email: g.vogiatzis@aston.ac.uk

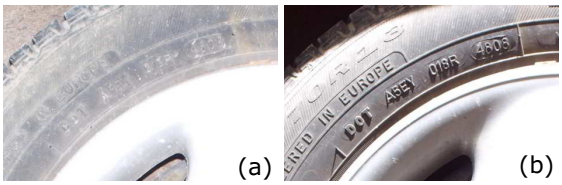

Fig. 2. Tyre sidewall with code printed. Same tyre and position with different illumination source position only. Light Angles (a) Frontal (b) Directional. The breakdown of the code is: A5 (manufacturing plant), EY (tyre size), 018R (manufacturer-specific batch number) and 4808 (date of manufacture in WWYY format) [10], [11]. This format may vary across manufacturers and models, especially the date part which may just be reduced to week or year only or may totally be omitted.

still enabling small and low contrast text reading, the tyre coverage was split up across a dual-camera arrangement as shown in the radial tyre coverage of Figure 4. An add-on benefit of this arrangement is that it focuses on the upper half of the tyre only, which relaxes the illumination source design.

Figure 2 shows that strong directional lighting at an acute angle enhances the contrast and legibility of the text. Keeping this in mind, 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, using industrial grade GigE cameras. Depending on the speed of the vehicle, generally 5 to 10 images/axle are acquired at this FPS which ensures full radial coverage of the tyre sidewall.

III. Tyre Detection and Unwarping

Once images are acquired, the circular segment of the tyre is detected using a Circular Hough Transform (CHT) [12]. The images are pre-processed to normalize the illumination. CHT is used to detect the circular junction of the hub cap and tyre (see Figure 3). But sometimes the wrong circle is detected due to another circularity (such as a wheel arch) being more dominant (stronger contrast in the image). 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 radius corresponding to the bin with maximum votes is selected as the best fitting tyre radius.

Once the junction of the hub cap and tyre is detected, a second circle corresponding to the outer radius of the tyre (Figure 3 (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 3 (a). This not only unwarps the circularity, but also crops out only the tyre part of the image, which increases the processing efficiency of the subsequent stages.

The first three steps of the pipeline, namely, image acquisition, tyre detection and unwarping are implemented in $C\#$ and takes about 500 ms/image.

IV. Code Detection

For code detection, the character sequence *DOT* (Department Of Transport, USA) is used as an anchor. It must be first detected to narrow down the search space as in most cases it precedes the code. In this section, a machine-learning approach for localizing the code by detecting *DOT* in the unwarped images is discussed. It has two stages in a cascade i.e. proposals or Regions of Interest (RoI) generation (Figure 6 (b)) followed by verification or code localization. To generate proposals for code localization, we used hand crafted features due to their low memory footprint and efficiency given the size of the image (average size 500x2800 pixels). Low-level feature-based approaches such as edge boxes [13] were found unsuitable since the text is of very low contrast and the strong edges from the other segments of the tyre with or without text, dominate.

Mathematically engineered features such as Histograms of Oriented Gradients (HOG) have been successfully used for text detection [14], [15]. But sliding a classifier such Support Vehctor Machines (SVM) is computationally costly. For a low cost industrial system, we would ideally like to have end-to-end results in less than a minute on a CPU for the given image size.

Networks with fully connected layers as convolutional layers such as YOLO [16] or ROI pooling before the fully connected layers such as Faster-RCNN [17] and Mask-RCNN [18] on the other hand, offer an efficient alternative which makes them strong potential candidates. But our images are much bigger in size and using deep networks to scan and localize text would either require longer time on a CPU or else demand a high memory GPU (8 GB or more), which increases the total system cost. Therefore, in this paper, we propose a solution by combining mathematically hand engineered features with a CNN based 2-layered classifier for efficiently generating proposals. Before we delve into the details, it should be mentioned here that the CNN networks in this paper were trained using Stochastic Gradient Descent (SGD) with back propagation in Matlab using MatConvNet library [19]. 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 layers during the

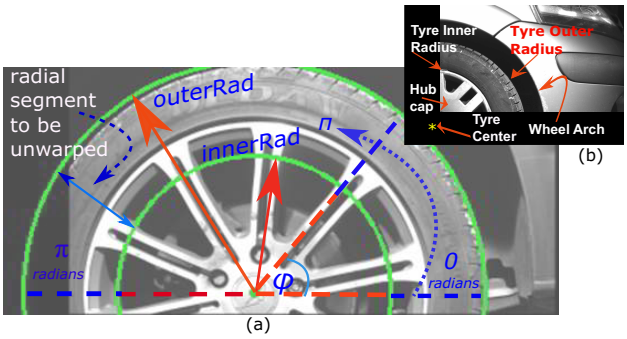

Fig. 3. Unwarping scheme with tyre's inner and outer radii. Unwarping is done through Polar-Cartesian mapping.

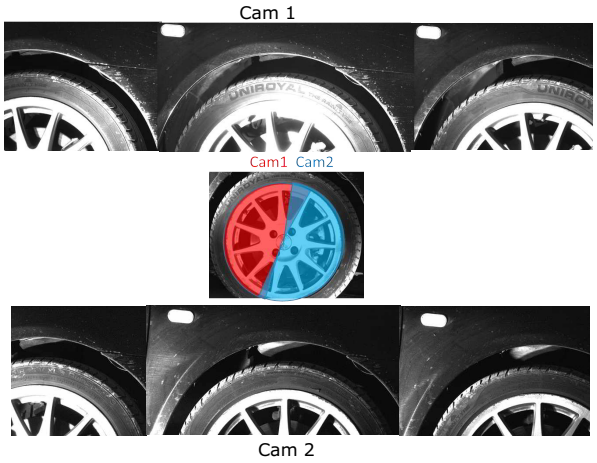

Fig. 4. Tyre radial coverage per camera and sample images.

training [20].

A. Synthetic data generation

The 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 sliding the rendering position in a small neighbourhood (Δx, Δy pixels)) in varying directions to depict the revolving light-shadows. The amount of the smear represents length of the shadow. This image mask was then fused with real tyre backgrounds to produce realistic embossed/engraved text images as they should appear on a tyre sidewall. Figure 5 (a) shows a sample set.

B. Proposal generation for code detection

For proposal generation, we first extract the Histogram of Oriented Gradients (HOG) features and interface them to a CNN-based Multi-Layered Perceptron (MLP) appropriate for a multi-class task [21]. Therefore we call it HOG-MLP. For this purpose, we used VLFEAT's HOG implementation [22] with an empirically selected input image size of $60(H) \times 130(W)$. In VLFEAT HOG, gradients are binned for $3\times$ the number of orientations + 4 texture components [23]. So, for the given image size, an 8×8 HOG cell size and 12 orientations (40 components in total), the first layer in the network was $8 \times 16 \times 40 \times ...$[1]. The selected cell size and the number of orientations were found to yield best results. Training was done on an 11 class dataset (7 synthesized DOT classes and 4 background classes). This 2-layered shallow network was efficiently trained even on a Core i7 3.6 GHz CPU and 50-60 epochs were deemed sufficient. During test or application, images are scanned at three scales i.e. original image, 1.25% and 0.75%.

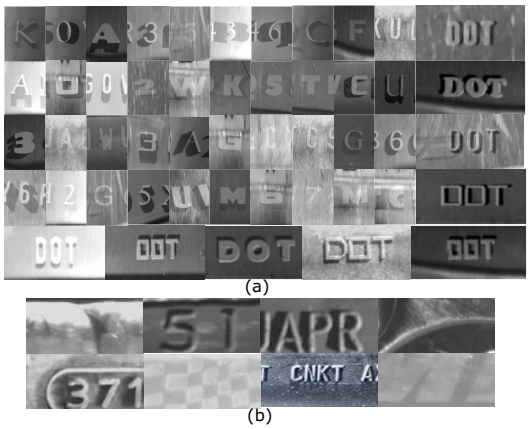

Fig. 5. (a) Synthetically generated data (b) Real tyre patches (used for background classes).

C. Code localization

Once the proposals for code localization have been generated, non-maximum overlapping bounding boxes are suppressed (NMS) using box area intersection-to-union ratio compared to a fixed threshold[2]. On the filtered proposals, we use a deep network similar to Jaderberg etal.'s 90K dictionary network [5][3] for code localization. The training set contained multiple DOT and background classes (10 classes in total: 7 DOT classes for round/square/thin and broad fonts, clear and diffused appearance, long and short shadows, spacing between characters etc. 3 background classes for plain background, edges/ridges 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 are rejected and only a few strong candidates are retained (the green boxes in Figure 6(b)). False positives seeping through at this stage are addressed through text recognition.

V. Code Reading

Code reading consists of two stages, text detection and recognition. The code patch of the image is first preprocessed 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 match the input image size of the text detection network.

A. Text detection

The code characters are detected using our text detector network[4]. Since the text has very low contrast with respect to the background, a dense prediction mechanism is required. Max pooling layers downsample the image which in fully convolutional networks such as OverFeat [**?**], increases the network stride (number of pixels skipped between to consecutive detection windows on the input image). Removing max pooling layers will allow per pixel predictions but will

[1]Input (8x16x40) → (Conv 8x16x40x100 (FC1) + ReLU → Conv 1x100x11 → Loss/Prob)

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

[3]Input (32x100) → 5x5x64 Conv + ReLU + MP 2x2 (CR5x64 MP) → CR5x128 MP → CR3x256 CR3x512 MP → CR3x512 CR4x13x4096 (FC1) → CR1x4096 (FC2) → CR1x10 (FC3) → Loss/Prob

[4]Input (32x32) → 5x5x64 Conv + ReLU + MaxOut /2 (CR5x64 MO2) → CR3x512 MO4 → CR16x1024 (FC1) MO4 → CR1x512 (FC2) → CR1x6 (FC3) → Loss/Prob

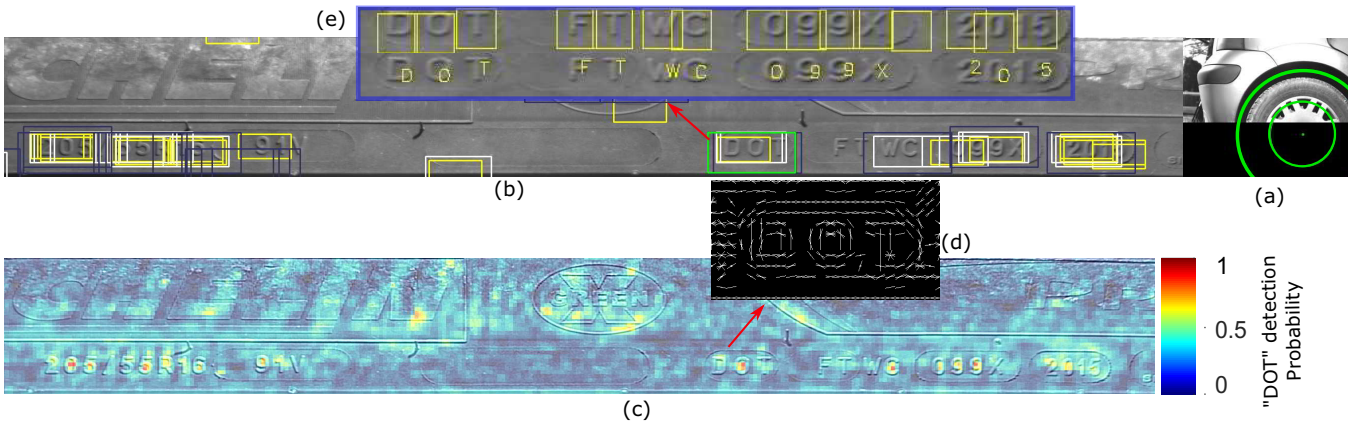

Fig. 6. (a) Tyre detected with centre outside the image boundaries. (b) Corresponding unwarped image with proposals for the code localization generated at 3 scales i.e. original (White), 1.25% (Yellow) and 0.75% (Blue). The green bounding boxes are the result of the deep code localizer network. (c) Code proposal scoremap generated at original scale. Higher probability of DOT presence is represented by increasing redness (d) Corresponding HOG feature visualization of the code anchor DOT. (e) Text detection and character classification [DOT FTWC 099X 20*1*5].

enormously increase the parameter space which will have its toll both on the efficiency and the accuracy. Regularization techniques such as DropOuts and MaxOuts are helpful in improving the accuracy [24], [25]. Therefore maxouts with dropouts were used in this architecture. We observed that if a Rectified Linear Unit (ReLU) layer precedes a maxout layer, the network converges quickly to a minimum.

Training was done on a 700K image dataset. The background class contained single edges, ridge patterns, cast or die shapes (sometimes used to emboss text on tyres), space between two characters with both characters with appearing partially and a plain background class making a total of 6 classes including a text class. The output was mapped to a binary class probability i.e. Text/non-Text. The text detector produces bounding boxes centred around regions with the highest probabilities of text being present. Non-maxima suppression (NMS) is first applied to the detected bounding boxes. We could have used a character classifier for text detection as well. But, in our experiments, we have observed that a dedicated classifier for text detection yields superior results.

B. Character Recognition

Detected text locations are used to extract characters which are fed into a character classifier network based on Jarderberg etal.'s 90K dictionary architecture [5] (section IV-C). This network receives a 32×32 input image and has classes for numerals 0-9, capital alphabet A-Z (excluding I, Q, S and O which are not used in the tyre codes) and 7 background classes, making a 39-way classifier which is mapped to 33 classes (32 characters 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 them in detail below.

A. Accuracy

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. This tendency has been avoided to some extent by injecting random noise and affine deformations during the training.

The accuracy of such an industrial system is subjected to light conditions, weather (dry/wet), object's (tyre's) condition/material/age/wear & tear; in short, in the absence of a benchmark, it is difficult to quantify accuracy. We have therefore assessed the accuracy on nine representative images which depict very well the possible scenarios. Figure 7 (a) show increasing complexity of text legibility from images 1 to 6. Images 7 to 9 are even difficult for human observers. Figure 7 (b) shows the accuracy graph which is calculated for every code image as: Total number of characters - Number of misclassifications (including background detected as text)/Total number of characters.

The code proposal generator is less likely to miss any region containing characters *DOT*, 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 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 7 (a) images 7 to 9 show situations in which the text legibility is compromised due to rain water (image 7

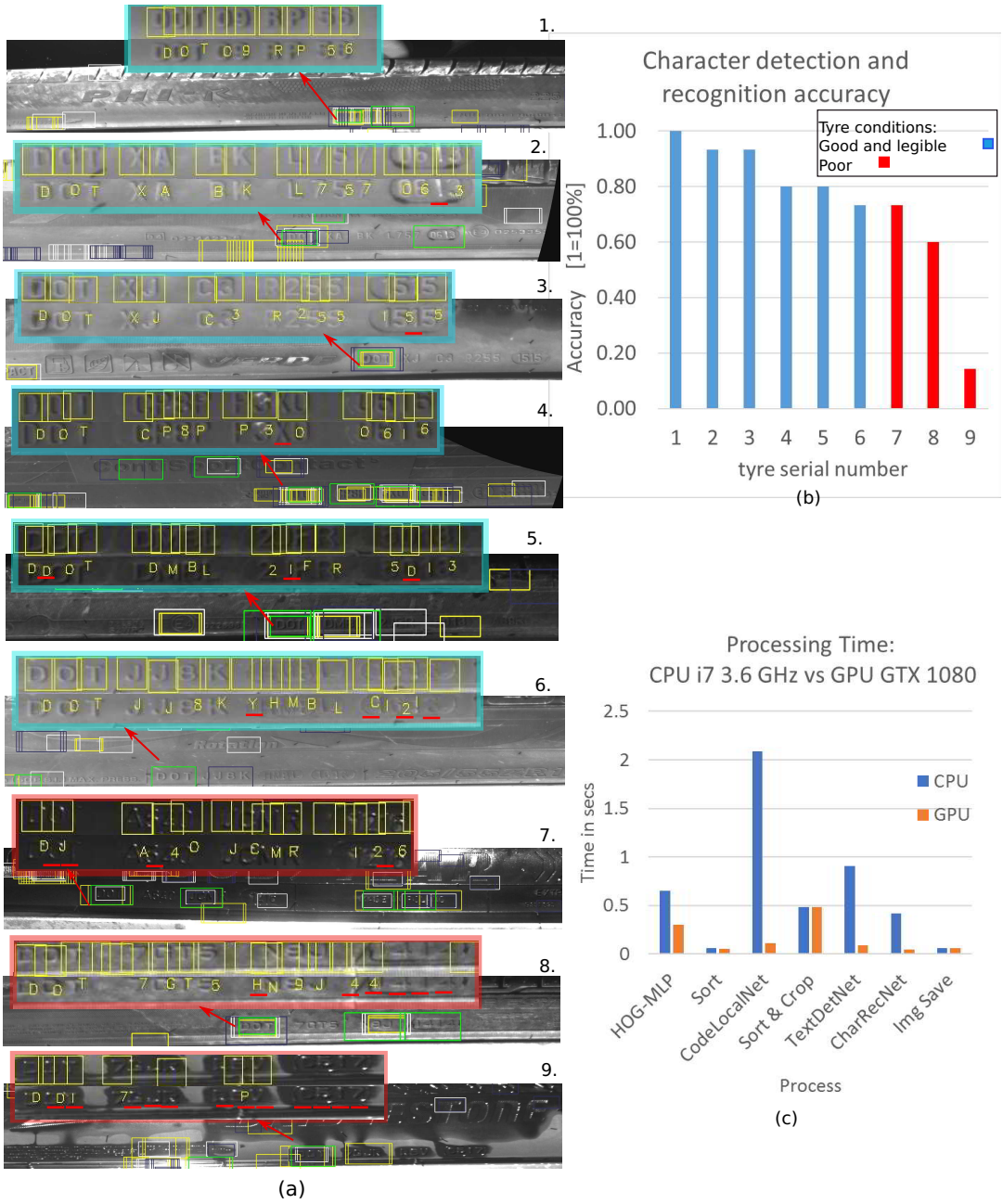

Fig. 7. (a) Nine different unwarped tyre images with varying degree of contrast and complexity. HOG-MLP proposals are represented in white, yellow and blue boxes for three scales i.e. original size, 1.25% and 0.75%. Code boxes verified by deep network are in green. 1) Tiny font size but a clean tyre [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 *1613*. 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).

and 9) creating undesired reflections and scintillations off the tyre surface or muddy water splash (image 8) creating unwanted texture, both of which change the appearance of the text. Deep nets for 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.

B. Efficiency

For an industrial system, with an end user waiting for results, efficiency is also crucial. GPUs (Graphical Processing Units) have extensively been used in deep learning based systems, but deploying GPUs at the production sites means scaling up the total system cost. 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 major concern. 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 500x2800 pixels) with the deep network of section IV-C takes around 22 secs on a Corei7 3.6 GhZ CPU (required parameter memory 496 MB), which is done by techniques such as [17], [4], [6]. Our cascade of HOG-MLP (required parameter memory 1 to 3 MB) followed by a deep scan of proposals thus generated reduces this total time to around 3 sec. It is an improvement by an order of magnitude in terms of efficiency (more than 85% 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 a 900x1400 pixel image for tyre detection and unwarping, and then scanning a 500x2800 pixel unwarped image at three different scales followed by detecting and reading the code (MATLAB) takes on average 6 to 7 secs on the above mentioned CPU. On a GTX-1080 GPU, this time reduces to 1 to 2 secs (see Figure 7 (c)).

VII. Conclusions

In this paper, we introduced a full 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 multi-layered perceptron. An efficient cascade of deep learning architectures was presented. For this problem, there is no benchmark to compare the accuracy against. However, the assessment of the test cases 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 diffused or effaced 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 head and a separate date classifier within a more unified framework than in a cascade.

References

[1] T. Roger and S. Cesare, "System and method for reading a tire code and obtaining tire-related information," *Patent (WO2017074759A1)*, p. https://patents.google.com/patent/WO2017074759A1, 2015.

[2] MicroEpsilon, "Reading the DOT code on tyres," *https://www.micro-epsilon.co.uk/applications/areas/Profil/DOT-Nummer/*, 2017.

[3] T. Wahdan, G. A. Abandah, A. Seyam, and A. Awwad, "Tire Type Recognition Through Treads Pattern Recognition and DOT Code Ocr," *Ubiquitous Computing and Communication Journal*, vol. 9, no. 3, pp. ISSN 1992–8424, 1992.

[4] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic Data for Text Localisation in Natural Images," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2016, pp. 2315–2324.

[5] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading Text in the Wild with Convolutional Neural Networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, jan 2016. [Online]. Available: http://dx.doi.org/10.1007/s11263-015-0823-z

[6] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: An Efficient and Accurate Scene Text Detector," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jul 2017, pp. 2642–2651.

[7] 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, nov 2017.

[8] ICDAR, "Robust Reading Comptetion," p. http://rrc.cvc.uab.es/?ch=4&com=evaluation&task=1, 2017.

[9] R. Panetta, "Method and Apparatus for Identifying Embossed Characters," *http://www.google.com/patents/US20110150346*, p. Patent US 20110150346 A1, 2009.

[10] Pirelli, "Tyre markings," *https://www.pirelli.com/tires/en-us/car-light-truck/find-your-tires/tire-use-guide-warranty/tire-marking*, 2017.

[11] Mitchelin, "Tyre sidewall information," *http://www.michelinrvtires.com/tires/tires-101/tire-basics/how-to-read-dot-identification/*, 2018.

[12] OpenCV-Docs, "Circular Hough Transform," *OpenCV*, p. https://docs.opencv.org/2.4/doc/tutorials/imgproc/, 2017.

[13] L. Zitnick and P. Dollar, "Edge Boxes: Locating Object Proposals from Edges," in *ECCV*. European Conference on Computer Vision, sep 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/edge-boxes-locating-object-proposals-from-edges/

[14] K. Wang, B. Babenko, and S. Belongie, "End-to-end Scene Text Recognition," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1457–1464. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126402

[15] A. Mishra, K. Alahari, and C. V. Jawahar, "Image Retrieval Using Textual Cues," in *IEEE International Conference on Computer Vision (ICCV)*, dec 2013, pp. 3040–3047.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2016, pp. 779–788.

[17] 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 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.

[18] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[19] A. Vedaldi and K. Lenc, "MatConvNet – Convolutional Neural Networks for MATLAB," in *Proceeding of the ACM, Int. Conf. on Multimedia*, 2015.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

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

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

[23] A. Vedaldi, "VLFeat HOG implementation details," *VLFEAT library*, p. http://www.vlfeat.org/api/hog.html, 2017.

[24] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout Networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–1319—-III–1327. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3043084

[25] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep Features for Text Spotting," in *European Conference on Computer Vision*, 2014.