2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)

Vehicle tire (tyre) detection and text recognition using 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 tire sidewalls. Images of vehicle tires in motion are acquired using roadside cameras. Firstly, the tire circularity is detected using Circular Hough Transform (CHT) with dynamic radius detection. The tire 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 tire 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 system presents impressive accuracy and efficiency proving its suitability for the intended industrial application.**

I. Introduction

Tire brand, its size, model, age and condition monitoring are critical for many vehicle users, especially fleet operators. This paper describes a complete system for the detection and recognition of tire codes printed on the tire sidewalls. Its a challenging task, because outdoor use of tires leads to wearing of the text due to dust, dryness, humidity and material erosion. Furthermore, generally, its a black text against a black background and therefore, has very low contrast as shown in Figure 1.

Industrial products available to read tire codes are either hand-held laser [1] or 3D scanning devices for use in indoor inspection tasks [2]. We could not find any image processing and machine learning based system for this problem.

Therefore, in this paper, we propose a 5-stage system as shown in Figure 2. Following the block diagram, section II will describe the image acquisition setup and section will III detail the tire detection/unwarping. We will describe a novel text proposal generator which combines hand-crafted features with a CNN network for code detection in section IV. Text detection and reading are described in section V. Results are discussed in section VI. Section VII will conclude 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 still enabling small and low contrast text reading, the radial coverage of a tire was split up across a dual-camera arrangement as depicted in Figure 3. An add-on benefit of this arrangement is that it focuses on the upper half of the tire only, which eases the illumination source design.

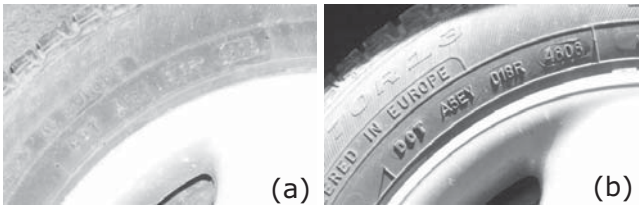

Fig. 1. Effect on light incidence angles on embossed tire text. (a) Frontal (b) Directional. Code: A5 (manufacturing plant), EY (tire size), 018R (manufacturer-specific batch number) and 4808 (date of manufacture in WWYY format) [3].

Strobe light incident at steep angles with respect to the plane of the sidewall as shown in Figure 1(b) increases legibility of the text. In order to achieve this objective, a light-reflector assembly was developed, installed at sufficient height to target the upper half of a tire. When a vehicle approaches the driveway, the cameras synchronized with the lighting mechanism are triggered. Images are acquired at 25 FPS (frames per second), using grade GigE cameras. Depending on the speed of the vehicle, generally 5 to 10 images/axle are acquired ensuring full radial imaging of the sidewall.

III. Tire detection

During image acquisition, owing to the distance between the vehicle and the light source, saturation and under illumination may occur. Therefore, the images were first pre-processed to normalize the illumination. The circular segment of the tire was then detected using a Circular Hough Transform (CHT) [4]. CHT was used to detect the circular junction of the hub cap and tire (see Figure 4). 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 were processed for n radii ranges. The number of detected circles were collectively voted in a radius

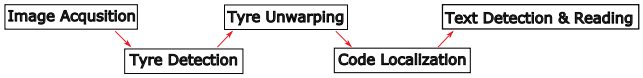

Fig. 2. Block diagram of the 5 stage sidewall text reading system.

[1]WheelRight Ltd. Begbroke Science Park, Oxford, OX5 1PF, United Kingdom 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

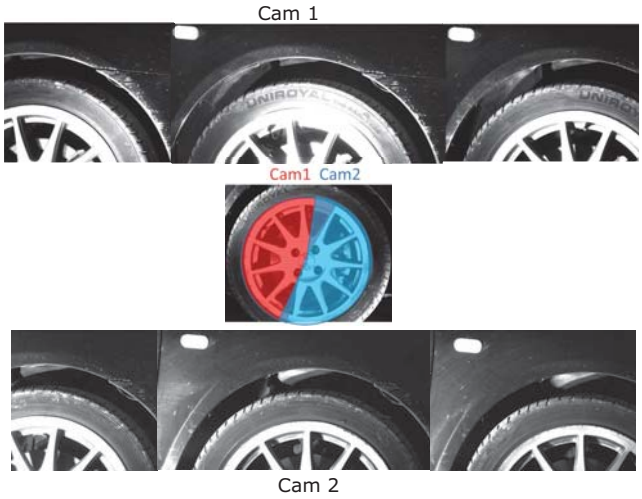

Fig. 3. Tire radial coverage/camera and corresponding sample images.

range histogram. The radius corresponding to the bin with maximum votes was selected as the best fitting tire radius.

Once the junction of the hub cap and tire was detected, a second circle corresponding to the outer radius of the tire (Figure 4 (b)) was chosen at a fixed offset from the first radius. This is a safe approximation since the tire code generally falls near the inner radius.

After tire detection, the radial image patch between the inner and the outer radii was unwarped to a rectangular lattice using a Polar-to-Cartesian mapping as shown in the scheme of Figure 4 (a). This not only unwarped the circularity, but also cropped out only the tire segment of the wheel image. This reduced image size thus improved efficiency of the subsequent stages which involved computationally expensive deep learning.

The first three steps of the pipeline, namely, image acquisition, tire detection and unwarping took 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 first be detected to narrow down the search space as in most cases it precedes the code. The detection mechanism has two stages in a cascade i.e. proposals or Regions of Interest

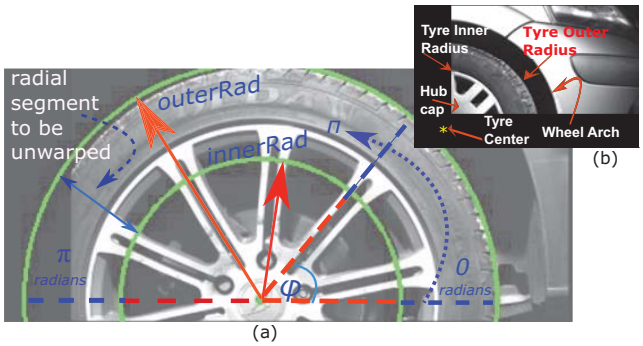

Fig. 4. Unwarping scheme for the tire's circularity.

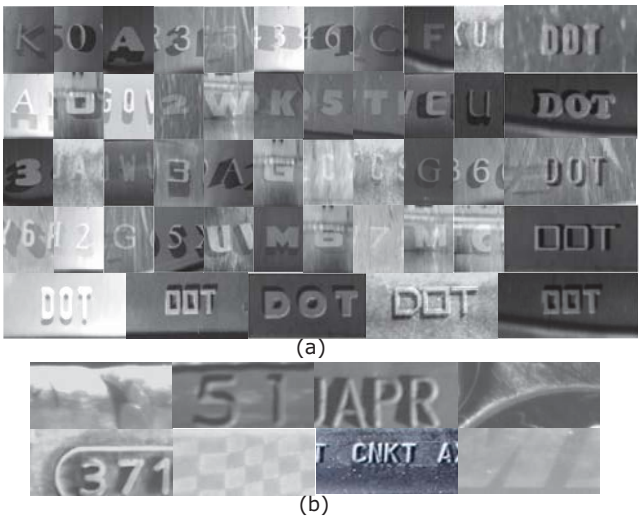

Fig. 5. (a) Synthetically generated data (b) Real tire patches (used for background classes).

(RoI) generation (Figure 6 (b)) and code localization. To generate code proposals, low-level feature based approaches such as edge boxes [5] were found unsuitable since the text is of very low contrast and the strong edges from the other segments of the tire with or without text, dominate. Given the size of the image (average size 500x2800 pixels), we used hand crafted features as they are efficient and demand low memory footprint.

Features such as Histograms of Oriented Gradients (HOG) have been successfully used for text detection [6], [7]. But sliding a classifier on large images such Support Vector Machines (SVM) would still be computationally expensive. For an industrial system, we would ideally like to have 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 OverFeat [8], SSD [9] and YOLO [10] or ROI pooling/align such as Faster-RCNN [11] and Mask-RCNN [12] on the other hand, offer an efficient inference-time alternative. This makes them strong potential candidates. But our images are larger in size and therefore, using deep networks to scan and localize text would either require longer times on a CPU or else demand a high memory GPU (8 GB or more), which increases the total system cost along with complexities of power and cooling for outdoor system, especially in hotter regions of the world. Therefore, in this paper, we propose a solution by combining mathematically 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 [13]. The text training data was synthetically generated whereas the background class was extracted from real tire images as shown in Figure 5. Every network used one or more 50% dropout layers during the training [14].

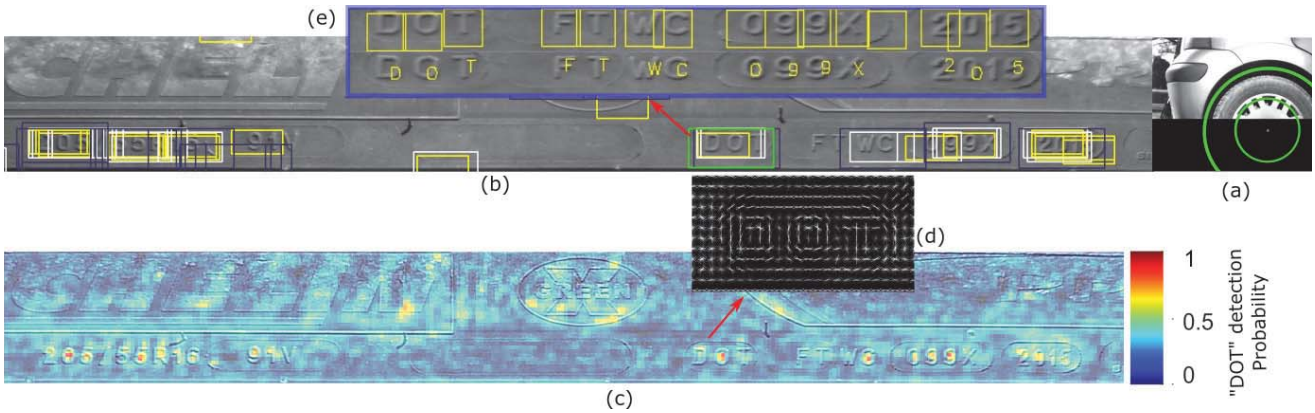

Fig. 6. (a) Sample tire detected with center outside the image canvas. (b) Corresponding unwarped image with proposals for the code localization generated at 3 scales i.e. original (White), 1.25% (Yellow) and 0.75% (Black). The green bounding boxes are the result of the deep CNN based code localizer network of section IV-C. (c) Code proposal score heatmap generated at original scale. Higher probability of DOT presence is represented by increasing red (d) Corresponding HOG feature visualization of the code anchor text *DOT*. (e) Text detection and character classification [DOT FTWC 099X 20*1*5].

A. Synthetic data generation

The training data was synthetically generated using several different fonts with a text rendering engine. Initially, a text masks were created using various fonts and sizes. They were then incrementally smeared (adding multiple copies or sliding the rendering position in a small neighborhood in multiple directions to depict shadows casted by revolving light. The amount of the smear represented the length of the shadow. The masks were then fused with real tire backgrounds to produce realistic embossed/engraved text images as they should appear on a tire sidewall. Figure 5 (a) shows a sample set.

B. Proposal generation for code detection

For the proposal generation, we first extracted the Histogram of Oriented Gradients (HOG) features and interfaced them to a CNN-based Multi-Layered Perceptron (MLP) appropriate for a multi-class task [15]. Therefore we called it HOG-MLP. For this purpose, we used VLFEAT's HOG implementation [16] 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 [17]. 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 were scanned at three scales i.e. original image, 1.25% and 0.75%.

C. Code localization

Once the proposals for code localization were generated, non-maximum overlapping bounding boxes were suppressed (NMS) using box area intersection-to-union ratio compared to a fixed threshold[2]. On the filtered proposals, we used a deep network similar to Jaderberg etal.'s 90K dictionary network [18][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 were rejected and only a few strong candidates were retained (the green boxes in Figure 6(b)). False positives seeping through at this stage would be filtered through text recognition.

V. Code Reading

Code reading consisted of two stages, text detection and recognition. The code patch of the image was first preprocessed to crop it down to the text height using low-level filtering. Bilateral filtering was done optionally in order to smooth out any unwanted background texture. Then the patch height was resized to match the input image size of the text detection network.

A. Text detection

The code characters were detected using our text detector network[4]. Since the text has very low contrast with respect to the background, a dense prediction mechanism was required. Max pooling layers downsample the image which in fully convolutional networks which increases the network stride (number of pixels skipped between to consecutive detection

[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

windows on the input image). Removing max pooling layers will allow per pixel predictions but will 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 [19], [20]. Therefore maxouts with dropouts were used in this architecture. We observed that if a Rectified Linear Unit (ReLU) layer preceded a maxout layer, the network converged 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 tires), 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 produced bounding boxes centered around regions with the highest probabilities of text being present. Non-maxima suppression (NMS) was first applied to the detected bounding boxes. 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 yielded superior results.

B. Character Recognition

Detected text locations were used to extract characters which were fed into a character classifier network based on Jarderberg etal.'s 90K dictionary architecture [18] (section IV-C). This network received a 32×32 input image and had classes for numerals 0-9, capital alphabet A-Z (excluding I, Q, S and O which are not used in the tire codes) and 7 background classes, making a 39-way classifier which was 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 was in part synthetic, even with regularization, models may still tend to overfit. This tendency was avoided to some extent by injecting random noise and affine deformations during the training.

The accuracy of such an industrial system is subjected to variation in light, weather (dry/wet), object's (tire'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 was less likely to miss any region containing characters *DOT*, as it responded strongly to a central *O* (Figure 6 (c)), especially with scanning done at three scales. Therefore, *DOT* was successfully detected in all of the sample images and thus this made the code detection, a fairly robust part of the cascade. Text detection and recognition, on the other hand suffered because of the above mentioned conditions. From 1 to 5, the text recognition accuracy was 80% or above which includes very tiny font (image 1), damp (image 3) and dark (image 4). Image 5 is an extremely dark tire 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 was compromised due to rain water (image 7 and 9) creating undesired reflections and scintillations off the tire surface or muddy water splash (image 8) creating unwanted texture, both of which changed 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 did 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 tire and incidence of the light source. In these examples, the accuracy varied 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 marked 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 along with added complexity of high power and cooling mechanism for outdoor sites in the hotter regions. 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 and complexity low is a major concern. Thus a CPU based system was ideally sought. It can be observed in Figure 6 that the interesting part of the tire 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 took around 22 secs on a Corei7 3.6 GhZ CPU (required parameter memory 496 MB), which was done by techniques such as [11], [21], [22]. Our cascade of HOG-MLP (required parameter memory 1 to 3 MB) followed by a deep scan of proposals thus generated reduced this total time to around

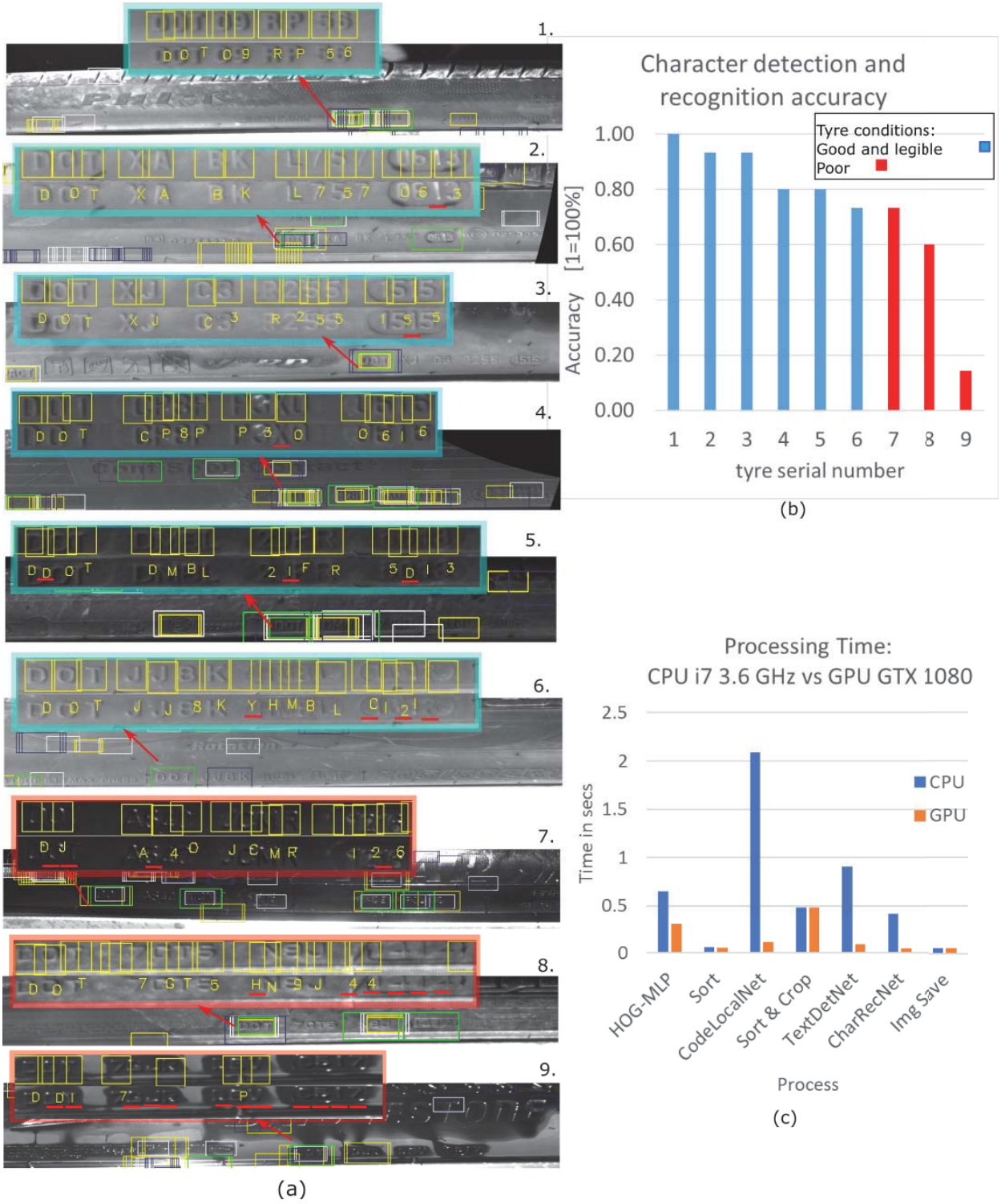

Fig. 7. (a) Nine different unwarped tire images with varying degree of contrast and complexity. HOG-MLP proposals in white (original size), yellow(1.25%) and blue (0.75%) boxes. Verified proposals by deep network are in green. 1) Tiny font size but a clean tire [DOT 09 RP 56]. 2) Clean tire with low background texture [DOT XA BK L757 06*1*3]. 3) Damp but clean tire 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 tire rubber with noticeable background texture [D D OT DMBL 2*J*FR 5*0*13]. 6) Worn out tire with effaced or rubbed off characters DOT JJ8K Y HMBL C *1613*. 7&9) Soaked wet tires 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).

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 results for processing a 900x1400 pixel image for tire detection and unwarping (C#), and then scanning a 500x2800 pixel unwarped image at three different scales followed by detecting and reading the code (MATLAB) took on average 6 to 7 secs on the above mentioned CPU. On a GTX-1080 GPU, this time reduced to 1 to 2 secs (see Figure 7 (c)).

VII. Conclusions

In this paper, we introduced a full pipeline for detecting and reading tire 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. Using state-of-the-art deep learning models and fully convolutional networks, a robust and efficient architecture 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.

Acknowledgment

The research was supported by Innovate UK under the Knowledge Transfer Partnership (KTP) Grant No. KTP009834. WheelRight Ltd. Oxford and the Department of Computer Science, Aston University, Birmingham, UK were involved in the collaboration. Special thanks to the support and supervision of Dr. Sylvia Wong, Ms. Jose Freedman and Mr. Martin May.

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

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

[5] 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/

[6] 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

[7] A. Mishra, K. Alahari, and C. V. Jawahar, "Top-down and bottom-up cues for scene text recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, jun 2012, pp. 2687–2694.

[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *International Conference on Learning Representations*, 2014.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007/978-3-319-46448-0{_}2

[10] 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.

[11] 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.

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

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

[14] 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

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

[16] 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/

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

[18] 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

[19] 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

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

[21] 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.

[22] 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.