Training-Free License Plate Detection Using Vehicle Symmetry and Simple Features

Sheng WANG, Qiang WU, Wenjing JIA, and Xiangjian HE
School of Computing and Communications
University of Technology, Sydney
Sydney, NSW2007, Australia

***Abstract*—In this paper, we are going to propose a training free license plate detection method. We use a challenging benchmark dataset for license plate detection. Unlike many other approaches, our approach is a training free method, which do not require supervised learning procedure and yet can achieve a reasonably good performance. Our motivation comes from the fact that, although license plate itself is largely variant in color, size, aspect ratio, illumination condition and so on, the rear view of vehicles are mostly symmetric when viewed on the vehicle's central axis. In addition, license plates for most vehicles are located along the vertical axis of the vehicle by which the vehicle is largely symmetric. Taking advantage of those prior knowledge, the license plate detection problem is made easier compared to the conventional scanning window approach which not only requires a large number of scanning window locations, but also requires different parameter settings such as scanning window sizes, aspect ratios and so on.**

I. Introduction

License Plate Detection (LPD) in object detection research is highly application oriented, it is widely used for various purposes. In this paper, we are going to introduce a novel license plate detection framework. Different from many previous methods for license plate detection, our approach do not require supervised learning. Moreover, comparing with those approaches which is based on classify scanning detection windows as either license plate or non-license plate, our approach only require a fairly small amount of scanning detection windows. Specifically, we have reduced the computational complexity from $O(M \times N)$ to $O(M + N)$, where M represents height of an input image and N represents the width of an input image.

There are many approaches for license plate detection. In general, those approaches can be divided into two large categories, learning based approaches and non-learning based approaches. As previous mentioned, this paper belongs to the later category, *i.e.* non-learning based approach. However, the difference between our approach and many other non-learning based approaches is that we have embedded the higher level semantic information into the low level feature extraction algorithm. This way, not only feature extraction process is made quicker, but also the process of parameter tunning is simplified.

In [2], the authors proposed a segmentation based license plate detection method using the Mean Shift algorithm for image segmentation and three features was proposed. Those three features are: rectangularity, aspect ratio, and edge density. The Mahalanobis classifier is later used for classification. The edge density feature was further used by both [3] and [4] to build a simple thresholded filter in a preprocessing step in order to reduce the computational cost needed for the detection process based on a supervised learning framework.

In [5], the authors proposed a non-learning based license plate detection framework using TO-MACH filter, a powerful correlation filter algorithm which was extensively used to recognize distorted object.

In [6], the authors used the Maximally Stable Extremal Region (MSER) for license plate detection, they also utilized some prior knowledge for the detection and localization, which is the constraints of the gray level jump.

In [7], the authors proposed a system based on DSP-platform, which is capable of performing both license plate detection and license plate recognition.

In [8], the authors used a multi-stage approach for license plate detection, the edge images are first extracted, then histogram analysis together with compact factor measurement is used on the extracted edge images to determine the potential candidate license plate. Finally, some morphological operators are used for the detection.

In [9], the author presented a new license plate detection method based on *Lab* space. Their proposed method first categorize the candidate region into blue plate or yellow plate (two major categories of license plates) in order to select the corresponding threshold matrix for optimal parametrization. Then based on certain prior knowledge, the *Lab* space is used instead of gray level to better locate a potential license plate region.

Through this paper, we have contributed the followings

Firstly, we have proposed a novel training-free license plate detection method which aims at detecting license plate from natural scene images. The novelty of our method, however, is not the training-free nature of the algorithm, but those assumptions which is related to the particular practical problem of license plate detection. Those assumptions can be considered as some prior knowledge, and the connections between this method and other supervised learning methods is that in supervised learning methods, those prior knowledge is obtained from training examples through machine learning algorithms in an (usually) iterative manner, whilst for our work, the prior knowledge is well known and manually fitted into the framework.

Secondly, although our approach aims at detecting license plates, it is however, not restricted to the problem of license plate detection only, because object symmetry is a widely known effect for most of artificial (human made) objects, such as vehicles and buildings. Moreover, many natural object, such as butterflies, leaves, and flowers also exhibit strong symmetry under certain viewing angles. Hence it is a reasonable approach to explore object symmetry and propose better object detection algorithms. From another point of view, we have well formulated a higher level semantic information, *i.e.* object symmetry into an object detection framework which originally can only relies on a strong learning algorithm (*e.g.* AdaBoost algorithm) to perform selection of low level image features for object detection. Now with this high level semantic information(latent feature derived from object symmetry), we can discard the need for a well trained classifier and simplify the computational cost.

Thirdly, we proposed a novel modification to the conventional Edgelet feature extraction algorithm. Instead of predefine a set of edgelets manually, we use a (left to right) flipped part of the input image to form an edgelet. Such edgelet is compared with another part of the image to calculate an affinity value, which is later used to determine the degree of symmetry of the input image over the compared pair of image regions (parts).

The rest of this paper will be organized as follows, Sec. II illustrates the framework of the proposed license plate detection algorithm. Sec. III provides the experimental results and some discussions. Sec. IV concludes this paper.

II. Proposed Framework

As mentioned earlier, our motivation comes from two observations: The first observation is that the rear views of most vehicles are horizontally symmetrical, this means we should be able to find a symmetric axis along the vertical direction for most rear view vehicle images, given the assumption that those vehicle rear view images are well aligned uprightly. The second observation is that most of the license plates are located at a predictable location of the vehicle, that is, most license plates are located at the centre or lower centre part of a vehicle.

The first observation indicates the pixel location of the license plate along the horizontal axis and the second observation indicates the pixel location of the license plate along the vertical axis. Together, the pixel coordinate of the license plate can be determined. In our framework, we set the top left corner of an image to be the origin, all pixel coordinates are positive integers.

Given a (uprightly) well aligned rear view vehicle image, the simple method one can imagine is to perform subtraction, which is similar to fold up a paper when one tries to find the central axis of it. However, in our case, it is not guaranteed that those horizontal edges of a vehicle is exactly along the horizontal axis of the input image. Moreover, different input images may have different levels of subtle left-right rotation due to the nature of the dataset. As a result, using a simple subtraction technique may fail to detect the symmetric axis.

In our approach, we use the low level feature (Edgelet) proposed by [1] for symmetry detection. Although the Edgelet feature was initially proposed for human detection, we found that it can serve our purpose well. In [1], different types of Edgelet templates (line, curve and hyperbola) were manually defined with prior knowledge of the shape of different human body parts, a supervised learning method is used for human detection. In this paper, however, we do not manually define any Edgelet template, because the pattern of vehicles' rear views are largely unpredictable. As a non-learning based method, our framework is both simple and powerful.

In order to proceed, we introduce some symbols and definitions.

The index i_{max} for the symmetric axis $\mathbf{V}$ which is parallel to the vertical direction can be represented by

$$i_{max} = \arg\max_{i} f(\mathbf{I}(:, i-k:i), \mathbf{I}(:, i:i+k)). \quad (1)$$

In (1), i represents the index (pixel location) for the symmetric axis, which is parallel to the vertical (Y) axis. k represents the width of the 'folded' image region, which should be identical to the other part (image region which lays flat on the desk with width k). Intuitively, the value of k should be large enough so that at least the left half of the vehicle can be 'flipped' to match against the right half for the affinity value calculation. However, if k is too large, some of the background images will also get 'flipped' and causing undesirable effects. Moreover, some vehicles may be too close to the image boundary for a larger k, in this case, it is likely that the algorithm can not work well. In our experiments, k is determined empirically. The function $f(\cdot, \cdot)$ in (1) calculates the affinity value between two image regions. Please refer to [1] for the definition of affinity value. $\mathbf{I}(:, i-k:i)$ represents a sub image of image $\mathbf{I}$, with all the rows and column $i-k$ to column i of $\mathbf{I}$. $\mathbf{I}(:, i:i+k)$ represents a sub image of image $\mathbf{I}$, with all the rows and column i to column $i+k$ of $\mathbf{I}$. This definition is consistent with the Matlab syntax.

For notation simplicity, define

$$\mathbf{I_1} = \mathbf{I}(:, i - \frac{m}{2} : i). \quad (2)$$

and

$$\mathbf{I_2} = \mathbf{I}(:, i : i + \frac{m}{2}). \quad (3)$$

In (2) and (3), m represents the width of the candidate plate region, the left part of such image region is represented by $\mathbf{I_1}$ and its right part is represented by $\mathbf{I_2}$.

When $\mathbf{V}$ is detected, the next step is to determine potential license plate region(s) among all those image regions along $\mathbf{V}$ with a width m. In our approach, we use a slightly modified version of the Vertical Edge Density Variance (VEDV) feature described in [3] for this task. Instead of considering the density variance, in our modification, we sum up all the density values for different sub-blocks and use the total value as a feature, for notation simplicity, we term it Summation of Vertical Edge

Density (VED^{Σ}). The image region which gives maximum response for VED^{Σ} feature will be considered as a candidate license plate region. Please refer to [3] for more details about the Vertical Edge Density Variance feature and the definition of sub-block.

The Vertical Edge Density Variance is a weak feature (simply used as a filter in [3]). The proposed VED^{Σ} feature is also a weak feature, in our experiments, we found that the VED^{Σ} feature is sensitive to plant textures. In order to obtain a better detection performance, we again make use of higher level semantic information, in particular, in the Caltech Car markus dataset, all vehicle rear views are complete, which implies that the license plate should be located at the middle or lower part of the image (top part of the image are either vehicle rear window or cluttered backgrounds). In order to make use of this information, we need to introduce one more parameter u, which represents the smallest possible index value for a potential license plate (recall that we are using the top left corner as the origin point). Larger u can remove more potential false positive regions, however, if u is too large, there will be missing plates. On the other hand, smaller u may not be sufficient to reduce the sensitivity of the VED^{Σ} feature. We demonstrate through our experiments that, introducing u can improve the detector's performance. Which again proved the validity of using prior knowledge to complement low level image features for a better detector performance.

Similar to (1), define j_{max} as

$$j_{max} = \arg\max_{j} VED^{\Sigma}(\mathbf{I}_1(j:j+l,:), \mathbf{I}_2(j:j+l,:)). \quad (4)$$

j_{max} is the index of the horizontal line $\mathbf{H}$, which overlaps with the top boundary of a potential license plate.

In (4), j represents the pixel location on Y axis where the maximum VED^{Σ} value is identified. l determines the height of the image patch which can be a potential license plate. In our experiments, similar to k, l and m are also empirically determined. $\mathbf{I}_1(j:j+l,:)$ is a sub image of $\mathbf{I_1}$ with row j to row $j+l$ and all the columns. $\mathbf{I}_2(j:j+l,:)$ is a sub image of $\mathbf{I_2}$ with row j to row $j+l$ and all the columns.

Finally, given i_{max} and j_{max}, which represents the horizontal and vertical indices of a potential license plate, the pixel coordinates of the license plate's top middle point is represented by

$$Intersection(\mathbf{V}, \mathbf{H}) = (i_{max}, j_{max}). \quad (5)$$

Details of our symmetry detection algorithm with Edgelet feature is illustrated in Fig. 1, note that the algorithm in Fig. 1 only describes how to detect the symmetric vertical axis in an image(example).

In Fig. 1, function $flip_{(left\ to\ right)}(\mathbf{I})$ flips an image $\mathbf{I}$ horizontally. Function $max(\mathbf{A})$ gives the exact value and index of the maximum element of matrix $\mathbf{A}$. In the algorithm, the exact value is represented by $maxval_A$ and the index is represented by $maxind_A$. The function $f(I_2, I_1')$ calculates an affinity value that describes how similar two image patches of the same size I_2 and I_1' are. Detailed algorithm for function $f(\cdot,\cdot)$ is given in Fig. 2.

1: $input : k, \mathbf{I}$.
2: $output : \mathbf{A} = [a_1, a_2, ...a_N], maxval_A, maxind_A$.
3: $[Height, Width] \leftarrow size(\mathbf{I})$
4: $N \leftarrow Width - 2 * k$
5: **for** $i = k+1 \rightarrow Width - k$ **do**
6: $\quad \mathbf{I_1} \leftarrow \mathbf{I}(:, i-k:i)$
7: $\quad \mathbf{I_2} \leftarrow \mathbf{I}(:, i:i+k)$
8: $\quad \mathbf{I_1'} \leftarrow flip_{(left\ to\ right)}(\mathbf{I_1})$
9: $\quad \mathbf{A}(i) \leftarrow f(\mathbf{I_2}, \mathbf{I_1'})$
10: **end for**
11: $[maxval_A, maxind_A] \leftarrow max(\mathbf{A})$
12: Return $maxind_A$ as the index for the symmetric axis

Fig. 1: Algorithm for Symmetry Axis Detection

1: $input : \mathbf{I}, \mathbf{J}$.
2: $output : a$.
3: $[Height_I, Width_I] \leftarrow size(\mathbf{I})$
4: $[Height_J, Width_J] \leftarrow size(\mathbf{J})$
5: **if** $(Height_I == Height_J)$ AND $(Width_I == Width_J)$ **then**
6: $\quad [\mathbf{Gx_I}, \mathbf{Gy_I}] \leftarrow gradient(\mathbf{I})$
7: $\quad [\mathbf{Gx_J}, \mathbf{Gy_J}] \leftarrow gradient(\mathbf{J})$
8: $\quad \mathbf{E_I} \leftarrow edge(\mathbf{I})$
9: $\quad mx \leftarrow$ max value of $\mathbf{E_I}$
10: $\quad \mathbf{E_I'} \leftarrow \frac{\mathbf{E_I}}{mx}$
11: $\quad \mathbf{C_I} \leftarrow arctan(\frac{\mathbf{Gy_I}}{\mathbf{Gx_I}})$
12: $\quad \mathbf{M_J} \leftarrow \sqrt{\mathbf{Gx_J} * \mathbf{Gx_J} + \mathbf{Gy_J} * \mathbf{Gy_J}}$
13: $\quad \mathbf{C_J} \leftarrow arctan(\frac{\mathbf{Gy_J}}{\mathbf{Gx_J}})$
14: $\quad \mathbf{aff} \leftarrow \mathbf{E_I'} * \mathbf{M_J} * cos(\mathbf{C_I} - \mathbf{C_J})$
15: $\quad a \leftarrow \frac{sum\ of\ all\ elements\ in\ \mathbf{aff}}{sum\ of\ all\ elements\ in\ \mathbf{E_I'}}$
16: **else**
17: $\quad$ error: $\mathbf{I}$ and $\mathbf{J}$ must be equally sized
18: **end if**
19: Return a as the affinity value.

Fig. 2: Algorithm for the Affinity Value Calculation

The $'*'$ symbol in Fig. 2 represents matrix convolution operation.

Our license plate detection result is indicated by the intersection of two lines. Also, in our current framework, each failed detection (missed example) is corresponding to one false positive.

III. Experimental Results

We test our framework on the Caltech Car markus dataset, which is composed of 126 images and 124 license plates. Among those 126 images, there are 124 images, of which each contains exactly one license plate, 2 images have name plate instead of license plate. But for our method, those 2 images can be considered the same as other 124 images as well, because we do not perform License Plate Recognition (LPR) at this stage. All vehicles images are from the rear view. In addition

Name	k	l	m	u
Value	300	40	100	100

TABLE I: Parameters

to wild illumination changes and different size of those license plates, the background of the vehicle images are very complex natural images in severely cluttered environments, from plants and sky to building and road surfaces of various texture.

Table I lists the detailed parameters.

In Tab. I, k represents the width of the image region which will be compared with another image region of the same size using Edgelet affinity values to measure their similarity. l and m represents the height and width of an image region which will be measured by the VED^{Σ} features to determine if it is a potential plate region. u represents the lower bound for the index of a potential plate region along the vertical direction.

In our experiments, the algorithm in Fig. 1 can successfully detect the symmetric axis for 89.68% of the images (113 out of 126). For all the test images, if we scan the symmetric axis from the top to the bottom, the proposed algorithm achieves a detection rate of 80.16% (101 out of 126) with 25 false positive regions. Under the assumption that all plates are having a vertical index value of greater than or equal to 100 (*i.e.* $u = 100$), the proposed algorithm can achieve a detection rate of 84.92% (107 out of 126) with 19 false positve regions.

For those failed examples, From Fig. 3, we can see that the background of those vehicles are either buildings which also exhibits very high symmetric properties or the vehicle itself is carrying a spare tyre which severely disturbs the detector. However, From Fig. 3l, we can also see that our algorithm can detect potential license plate even when the characters of the license plate is hardly visible. Fig. 3l also indicates that the proposed method have resistance to disturbances caused by non-object of interest (spare tyre mounted at the rear of the jeep).

Fig. 4 gives the detailed Edgelet affinity value and VED^{Σ} value for examples (l), (j), (f), and (e) in Fig. 3. In Fig. 4, the red vertical line indicate the actual pixel locations, and the data labels indicate the detection result given by the algorithm. Note that in the Edgelet Affinity value plot (the second column), the index needs to be increased by 300 in order to give the correct license plate coordinate, because in our experiment, we empirically set $k = 300$ and $l = 40$. Also, in Fig. 4l, the horizontal line **H** coincide with the ground truth because there is characters along the detected vertical line, however, the detected vertical line **V** is not the ground truth.

One may argue that this method assumes that the vehicle itself is symmetric in the captured images, also, it assumes that the license plate is located along the vertical axis by which vehicle is symmetric. Indeed, in reality, it is not always the case. In the Caltech Car markus dataset, we found that 123 out of 126 (*i.e.* 97.62%) of the vehicles are abiding by the rules of symmetry (both vehicle symmetry and license plate location symmetry), it is not practical to figure out exactly how many vehicles are not symmetric and how many license plates are located at asymmetric locations. Nevertheless, we can conclude that in the Caltech Car markus dataset, our algorithm works for 97.62% of the vehicles. This also implies that, given a sample set of 126, we can predicate under certain accuracy (probability), that in the Caltech Car markus data acquisization environment, 96.72% of vehicles are symmetric (both vehicle symmetry and license plate location symmetry). The practical applicability of our approach can be assessed once we have corresponding census data available. Such census data can be collected from highway entries or road authorities.

IV. Conclusion

In conclusion, we have proposed a training-free novel license plate detection framework based on higher level semantic information, which can be interpreted by two symmetry assumptions. Those assumptions are, firstly, most of the vehicles are symmetric, secondly, license plate for most vehicle are located along the vertical symmetric axis.

Given those two assumptions, our aim is to first extract the Edgelet feature, then find the maximum affinity value (Edgelet feature value as defined by [1]) in order to locate the vertical symmetry axis. After that, a modified Vertical Edge Density Variance feature, VED^{Σ}, is used to determine the coordinate of a potential license plate.

We tested our method on a benchmark datasets and the experimental results indicate that our training-free method is reasonably good for a challenging public dataset with complex background in a cluttered environment.

References

[1] B.Wu and R.Nevatia. Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[2] W.Jia, H.Zhang, X.He, and M.Piccardi. Mean Shift for Accurate License Plate Localization. In:*Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pp. 566–571, 2005.

[3] H.Zhang, W.Jia, X.He, and Q.Wu. Learning-Based License Plate Detection Using Global and Local Features. In:*Proc. ICPR*, pp. 1102–1105, 2006.

[4] H.Zhang, W.Jia, X.He, and Q.Wu. A Fast Algorithm for License Plate Detection in Various Conditions. In:*Proc. SMC*, pp. 2420–2425, 2006.

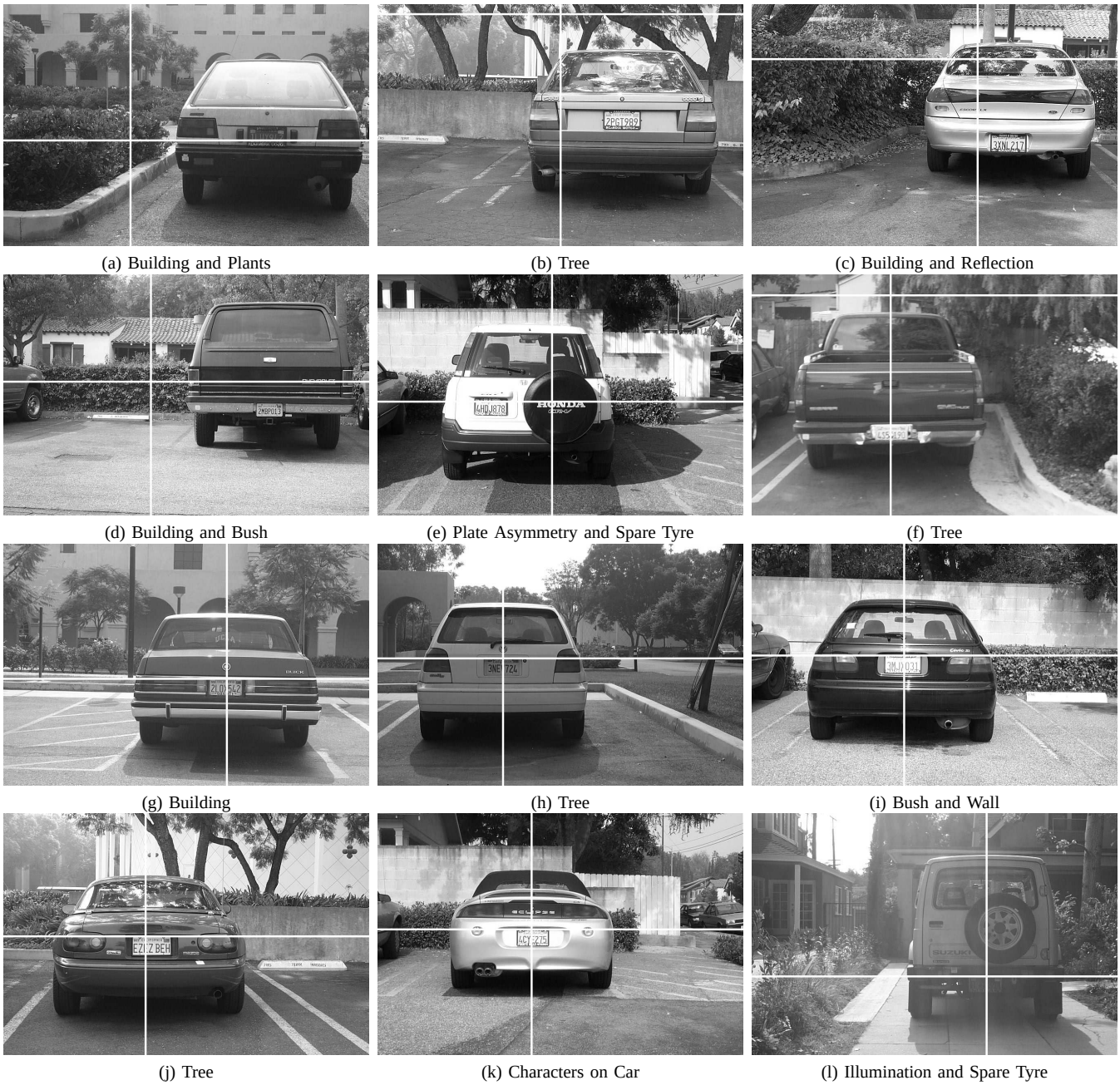
[5] J.Liu, J.Liu, and W.Jin. Robust Vehicle License Plate Localization Using TO-MACH Filter. In:*Proceedings of the 5th International Conference on Computer Science and Education*, pp. 1106–1109, 2010.

[6] W.Wang, Q.Jiang, X.Zhou, and W.Wan. Car License Plate Detection Based on MSER. In:*Proceedings of the 2011 International Conference on Consumer Electronics, Communications and Networks*, pp. 3973–3976, 2011.

[7] C.Arth, F.Limberger, and H.Bischof. Real-Time License Plate Recognition on an Embedded DSP-Platform. In:*Proc. CVPR*, pp. 1–8, 2007.

[8] F.Faradji, A.H.Rezaie, M.Ziaratban. A Morphological-Based License Plate Location. In:*Proc. ICIP*, pp. I-57–I-60, 2007.

[9] L.Xu. A New Method for License Plate Detection Based on Color and Edge Information of Lab Space. In:*Proceedings of the 2011 International Conference on Multimedia and Signal Processing*, pp. 99–102, 2011.



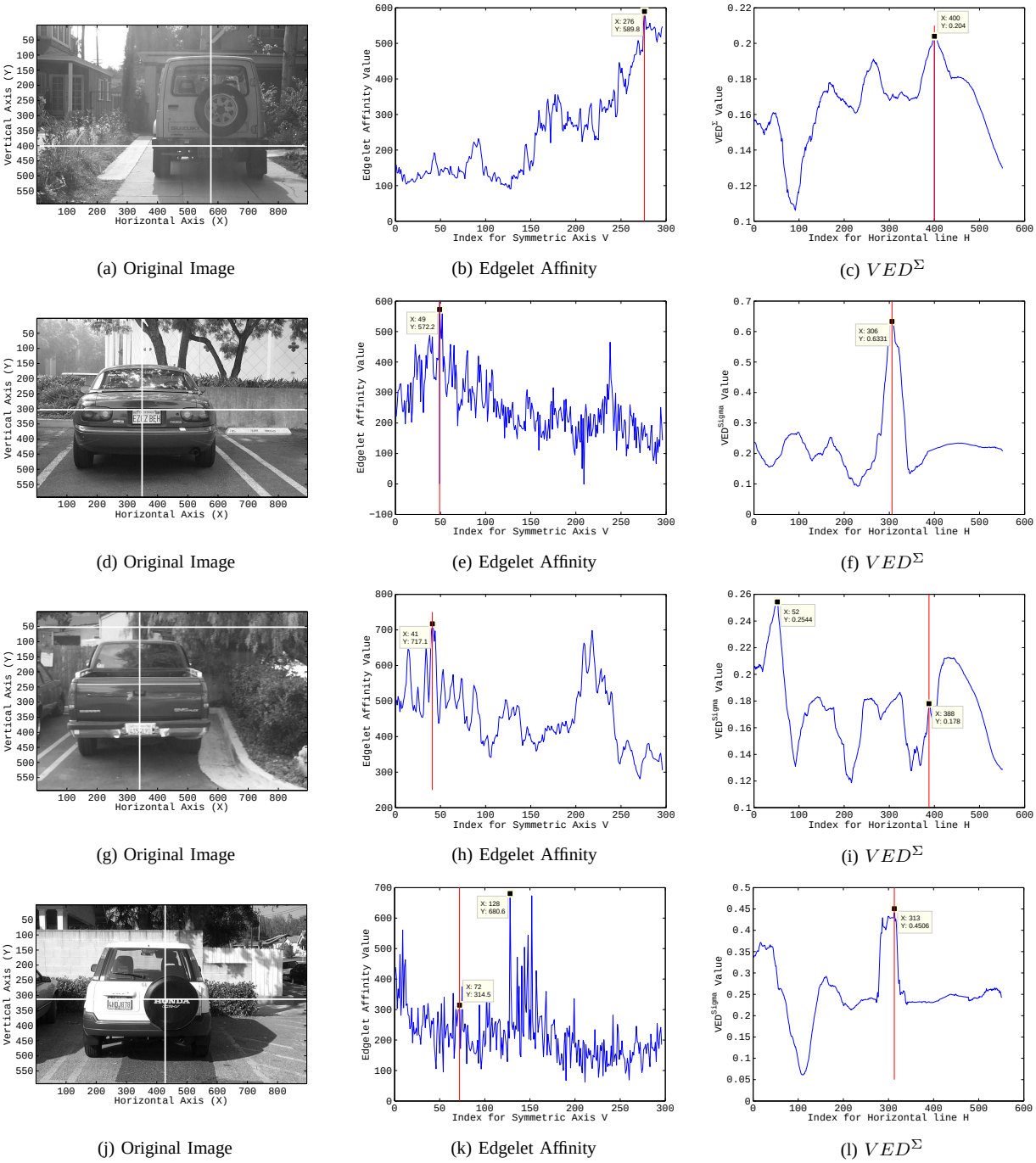
(a) Building and Plants (b) Tree (c) Building and Reflection

(d) Building and Bush (e) Plate Asymmetry and Spare Tyre (f) Tree

(g) Building (h) Tree (i) Bush and Wall

(j) Tree (k) Characters on Car (l) Illumination and Spare Tyre

Fig. 3: Some Examples (Sub-figure (a) to (f) are failed examples, the caption in each sub-figure indicates the object or factor which disturbs the detector. Sub-figure (g) to (l) are successful detections, the caption in each sub-figure indicates the challenging objects or factors posed to the detector).



(a) Original Image (b) Edgelet Affinity (c) VED^{Σ}

(d) Original Image (e) Edgelet Affinity (f) VED^{Σ}

(g) Original Image (h) Edgelet Affinity (i) VED^{Σ}

(j) Original Image (k) Edgelet Affinity (l) VED^{Σ}

Fig. 4: Detailed Edgelet affinity value and VED^{Σ} value for examples in Fig. 3(row 1 for example (l), row 2 for example (j), row 3 for example (f), and row 4 for example (e)).