Traffic Lights Detection Based on Deep Learning Feature

Changhao Wang, GuanWen Zhang(✉), Wei Zhou, Yukun Rao, and Yu Lv

Northwestern Polytechnical University, Xi'an 710072, China
{guanwen.zh,zhouwei}@nwpu.edu.cn

Abstract. Traffic lights detection is an important task for intelligent vehicles. It is non-trivial due to variance backgrounds and illumination conditions. Therefore, a traffic lights detection system that can apply to different scenes is necessary. In this paper, we research the traffic lights detection based on deep learning, which can extract features with representation and robustness from input image automatically and avoid using artificial features. The approach of traffic lights detection proposed in this paper includes two stages: (1) region proposal and (2) classification of traffic lights. Firstly, we propose a region proposal method based on intensity, color, and geometric information of traffic lights. Secondly, convolutional neural network (CNN) was introduced for the traffic lights classification, obtaining 99.6% average accuracy. For detection, we evaluate our system on 6804 images of different scenes, the recall and accuracy of detection achieve 99.2% and 98.5% respectively.

Keywords: Traffic lights detection · Deep learning · Region proposal · Classification

1 Introduction

Nowadays, the number of vehicles increases dramatically, which makes the traffic condition more complicated. Under this circumstance, the techniques of self-driving can be used to avoid accidents introduced by human emotion, physical condition, and qualification. The information of lanes' direction and the vehicles' or pedestrians' number and conditions nearby are crucial for self-driving cars to understand the environment so that they can make some adjustment when travelling. Therefore, the self-driving techniques include lane, vehicle and pedestrian detection. Besides, the navigating information of traffic lights also play an important role when vehicles travelling at the intersection, so traffic lights detection is one of the most important tasks for intelligent vehicles travelling safely.

Sponsored by the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University, zz2019140 and National College Students' innovation and entrepreneurship training program, 201810699167.

B. Li et al. (Eds.): IoTaaS 2019, LNICST 316, pp. 382–396, 2020.
https://doi.org/10.1007/978-3-030-44751-9_32

In this paper, we research traffic lights detection based on deep learning, which can extract features with representation and robustness from input image automatically, avoid using artificial features. Our approach can achieve high recall and accuracy with high processing speed in different scenes.

The approach of traffic lights detection proposed in this paper includes two stages: (1) region proposal and (2) classification of traffic lights. The layout of our approach is illustrated in Fig. 1.

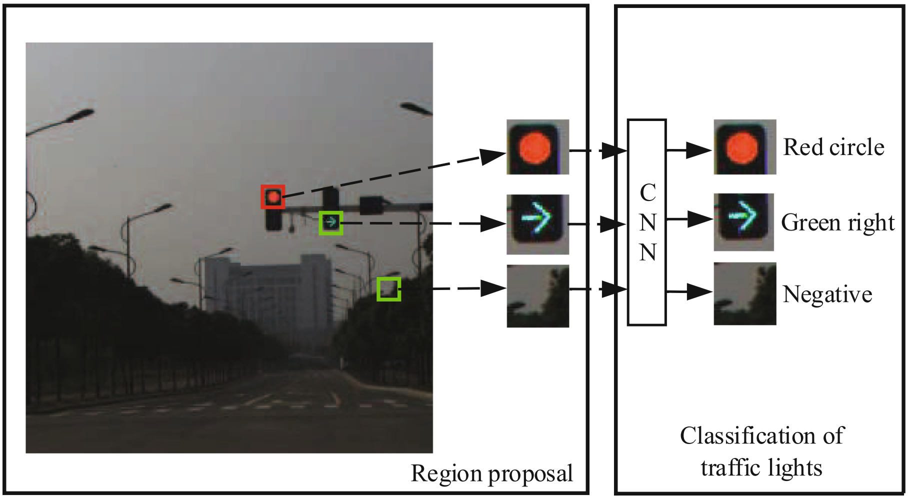


Fig. 1. Framework of the traffic lights detection

2 Related Work

Current traffic lights detection algorithm are mainly consist of two steps: extract features from images and apply a high-quality classifier or match a template for classification [12,13,28]. The most distinct features of traffic lights are color and intensity which can be used to propose the region of traffic lights from images. Omachi et al. [20] achieved traffic lights detection by using color and edge information. Their approach normalizes the RGB color space of input image, and some regions are selected as candidates of traffic lights. Then a method based on Hough transform is applied to obtain target regions. Yi et al. [17] adopted morphology filtration and statistical classification to detect traffic lights. In their method, original image is converted to a binary image by top-hat transform and threshold segmentation to obtain brighter regions firstly. Then the candidate regions that unsatisfying filtration conditions are removed by morphology and geometry features. Furthermore, a novel recognition method is carried out based on statistical analysis with amount of traffic lights image samples. It performs the color feature extracted by the Hue component in HSV color space for classifying traffic lights. In [24], a vision-based traffic lights detection method is proposed, which contains the candidates extraction and recognition. On the candidates extraction stage, they highlight the traffic lights candidate regions by perform an adaptive background suppression algorithm while suppressing the undesired backgrounds. Then, each candidate region is verified and further classified into different traffic light semantic classes. [2] uses a spot detection algorithm to detect traffic lights, and classify the detected spots with Adaptive Templates Matcher, which can avoid motion blur and illumination variations.

All those traditional detection methods use artificial features for classification, which can achieve traffic lights detection ideally in some scenes. However, those artificial features used for classification are only suitable for some fixed surroundings because of limited represent-ability. Therefore, we have to adjust those approaches for applying in different scenes. In recent years, deep learning has turned out to be excellent for discovering intricate structures in high-dimensional data, and it performs well in image analysis, speech recognition, and computer vision etc. Besides, deep learning has been applied to intelligent vehicle techniques successfully, mainly in vehicle and pedestrian detection [5,6,14,21].

Object detection based on deep learning mainly has two methods: (1) detection based on regions, e.g. Region-CNN (R-CNN) [8], Fast R-CNN [7], and (2) the method of end-to-end, e.g. You Only Look Once (YOLO) [22], Single Shot MultiBox Detector (SSD) [18]. At present, the method of end-to-end is very popular, because it can achieve high-accuracy detection with high processing speed. However, it will have some problems while applying to traffic lights detection. The traffic lights are small compared to other objects in a scene image. The input of CNN for end-to-end method is whole scene image, and the feature extracting process will downsample input image because of convolution and pooling operations. Finally, the region of traffic lights in the feature maps becomes too small to recognize. Intuitively, traffic lights has some distinct features, such as color, intensity, and shape, which can be used to distinguish traffic lights from surroundings [26,28]. Therefore, we can adopt the method of detection based on regions: proposing candidate regions according to those features firstly [3,10,11,27], and then introducing a CNN to classify candidate regions.

3 Region Proposal

We propose a region proposal method based on intensity, color, and geometric informations of traffic lights. Firstly, we employ Gaussian filtration and perform gray processing and Top-hat transform to process the intensity information of image. Secondly, we convert the image from RGB to HSI color space and filter the image according to the hue value of traffic light regions. Lastly, we restrict the regions' geometrical information to generate candidates. We can optimize the threshold of region proposal conditions to reduce the number of candidate regions, which is crucial to achieve high processing speed.

3.1 Intensity Filtration

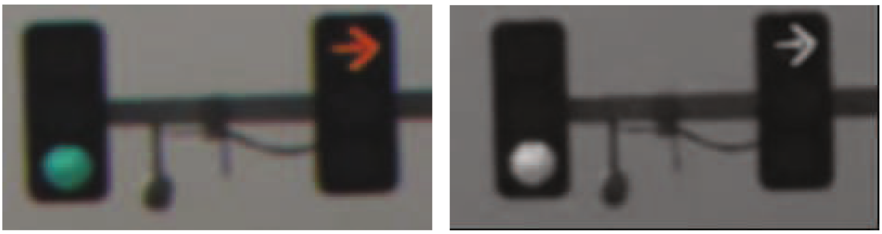
Traffic lights are kinds of illuminant body with high intensity, this can help us to tell them from surroundings. The intensity of traffic lights varies with lighting conditions caused by weather, surroundings and other factors. In order to eliminate the effect of some small regions with high luminance, we employ Gaussian filtration. At the same time, Gaussian noise can be filtered. We achieve Gaussian filtration by using Gaussian kernel, and the kernel's values distribute according to (1).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{1}$$

Where x, y means relative coordinates of the points in Gaussian kernel to the central point.

After eliminating the difference of intensity, we convert the RGB image to gray level image to facilitate intensity information processing. For traffic lights detection task, we have to highlight the regions of red and green in images, which means the weights of R and G in the gray processing formula should be increased. The gray processing formula we proposed is shown as (2). Figure 2(a) shows an image after Gaussian filtration, and Fig. 2(b) is the result of gray processing.

$$Y = \max(0.9R - 0.1G - 0.3B, 0.9G + 0.1B - 0.5R) \tag{2}$$



(a) Image after Gaussian filtering (b) Result of gray processing filtration

Fig. 2. Gray processing

We use (2) for processing the image after Gaussian filtration to highlight the regions in red or green, and then we need to distinguish them from background. The Top-hat transform [9,25], demonstrated in Fig. 3, was used to solve this problem, which is effective to obtain the bright aggregate of pixies from dark background. The top-hat transform was described as (3) in [24].

$$Tophat(f) = f - (f \circ b) = f - ((f \odot b) \oplus b) \tag{3}$$

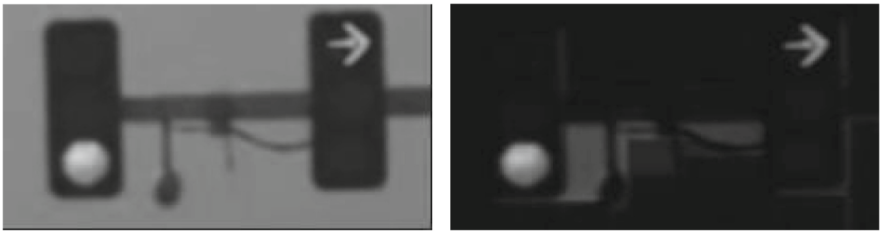
In (3), f is an input image; b is a kernel element; $\circ$ is opening operation, and defined as (4).

$$f \circ b = (f \odot b) \oplus b \tag{4}$$

Where operator $\odot$ and $\oplus$ is defined as (5) and (6) respectively.

$$f \odot b \,|\, (x, y) = \max \{ f(x - x', y - y') - b(x', y') \,|(x', y') \in D_b \} \tag{5}$$

$$f \oplus b \,|(x, y) = \max \{ f(x - x', y - y') + b(x', y') \,|(x', y') \in D_b \} \tag{6}$$



(a) Image after gray processing (b) Result of Top-hat transform

Fig. 3. Top-hat transform

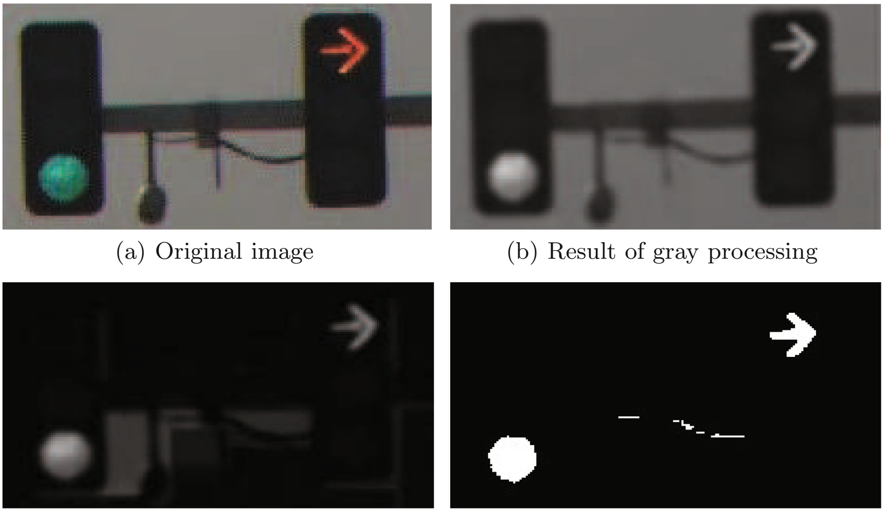
Next, we convert the image after Top-hat transform to binary image with a threshold to obtain the result of intensity filtration, the transformation uses (7).

$$Binary(x, y) = \begin{cases} 255, v(x, y) \geq T \\ 0, v(x, y) < T \end{cases} \tag{7}$$

In (7), $v(x, y)$ is the value of pixel in the image after Top-hat transform. Figure 4 shows the result of intensity filtration.

3.2 Color Segmentation

Compared with backgrounds, the color of traffic lights is also a distinctive features [1,26]. Therefore, we perform color segmentation after intensity filtration to remove background as much as possible. Since the intensity is not a concern and processing color information of an image in HSI color space is convenient, the input image is converted from RGB to HIS color space [4,23]. HSI stands for hue (H), saturation (S), and intensity (I), and H can be used to restrict the image for achieving color segmentation. To reduce computations, only the pixels of input images corresponding to the reserved regions of the result of intensity filtration are converted. The transformation from RGB to HIS uses (8), (9) and (10).



(a) Original image (b) Result of gray processing

(c) Result of Top-hat transform (d) Result of intensity filtration

Fig. 4. Intensity filtration

$$H = \begin{cases} \arccos\left(\frac{(r-g)+(r-b)}{2\sqrt{(r-g)^2+(r-b)(g-b)}}\right), b \leq g \\ 360 - \arccos\left(\frac{(r-g)+(r-b)}{2\sqrt{(r-g)^2+(r-b)(g-b)}}\right), b \geq g \end{cases} \tag{8}$$

$$S = 1 - \frac{3\min(r,g,b)}{r+g+b} \tag{9}$$

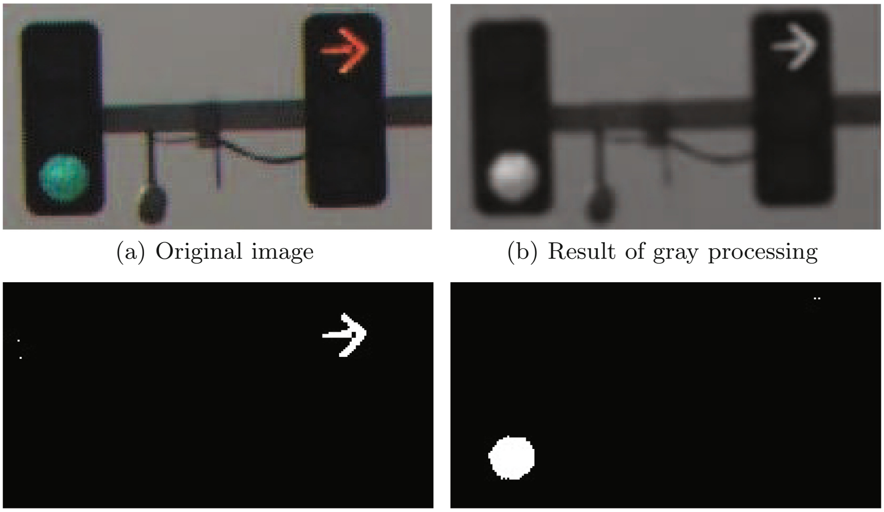
$$I = \frac{r+g+b}{3} \tag{10}$$

Where r, g and b are the normalized values of R, G and B in RGB color space.

The value of H varies from 0 to 360, starting at the red primary at 0, passing through the green primary at 120 and the blue primary at 240, and then wrapping back to red at 360. According to the statistics, the condition of color segmentation is defined as (11), (12). Figure 5 shows the result of color segmentation.

$$\text{Red region}: 0 \leq H \leq 70, 340 \leq H < 360 \tag{11}$$

$$\text{Green region}: 110 \leq H \leq 250 \tag{12}$$



(a) Original image (b) Result of gray processing

(c) Regions satisfied formula (12) (d) Regions satisfied formula (13)

Fig. 5. Result of color segmentation

3.3 Geometry Filtration

The majority backgrounds of an input image can be removed after performing intensity filtration and color segmentation. However, in a scene image some objects always exist, of which intensity and color are similar to traffic lights. To solve this problem, we can utilize the geometrical information to restrict the result of color segmentation, and obtain the candidate regions. It is known that the pixels of traffic light regions should be conterminous. So, we can detect the conterminous regions and draw circumscribed rectangles of those regions firstly, and then restrict these rectangles with geometrical conditions. However, some "black holes" in the target regions will make them un-conterminous and undetectable due to the previous processing. Therefore, before we performing geometry filtration, a closing operation is employed. Closing operation can pad "black holes" to make regions conterminous. It is defined as (13). Figure 6 shows the function of closing operation.

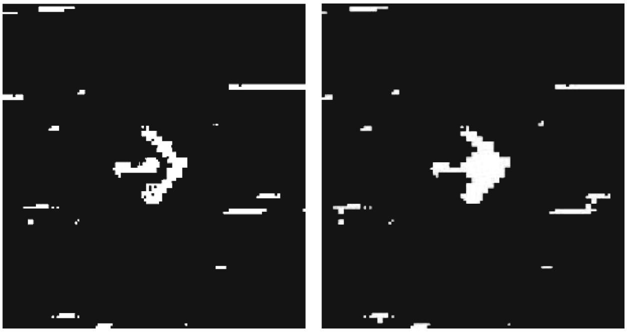
$$f \bullet b = (f \oplus b) \odot b \tag{13}$$

Where operator $\odot$ and $\oplus$ is defined as (5) and (6) respectively.

From Fig. 6, it is clearly that the "black holes" among the arrow light are padded. Next, we can draw circumscribed rectangles of conterminous regions, with restriction of the height-width ratio ($Ratio$) and area (S), the restrict conditions are shown as (14).

$$\begin{cases} S_{\min} \leq S \leq S_{\max} \\ 1 \leq Ratio \leq \frac{\max(width, height)}{\min(width, height)} \end{cases} \tag{14}$$

Where $S_{\min}$ is the minimum area of the circumscribed rectangles; $S_{\max}$ is the maximum area of the circumscribed rectangles; *width* is the width of the circumscribed rectangles; *height* is the height of the circumscribed rectangles. Figure 7 shows the result of region proposal.



(a) Before closing operation (b) After closing operation

Fig. 6. Closing operation

Fig. 7. Result of region proposal

4 Classification of Traffic Lights

During this stage, we use a CNN for traffic lights classification [8,15]. We can obtain a set C of candidate regions after performing region proposal. C contains the location $c_i(x_i, y_i, width_i, height_i)$ of each candidate region. In C, (x_i, y_i) is the coordinate of the top left vertex of candidate regions, $width_i$ and $height_i$ are the width and height of candidate regions respectively. According to the locations, we can cut out corresponding regions from original images as inputs of the CNN for classification. The results of classification $(label_i)$ will be returned to set C and obtain set D which contains the location and label

($d_i(x_i, y_i, width_i, height_i, label_i)$) of each candidate region. The CNN for classification only has 6 convolution layers, which can ensure high classification accuracy and fast processing speed. The data augmentation [16,19] is performed for training the CNN.

4.1 CNN Model

The CNN can perform feature learning on the basis of input data, and it can extract multiple feature maps by convolution and pooling operation. In the output layer of CNN, we usually utilize the softmax classifier for classification. The structure of the CNN model is shown as Table 1.

Table 1. CNN model

Type	Structure
input	size: 40 × 40 × 3
conv1	kernel size: 3 × 3, output number: 32, stride: 1, pad: 1
conv2	kernel size: 3 × 3, output number: 32, stride: 1, pad: 1
pooling1	max pooling, kernel size: 3 × 3, stride: 2
conv3	kernel size: 3 × 3, output number: 32, stride: 1, pad: 1
conv4	kernel size: 3 × 3, output number: 32, stride: 1, pad: 1
pooling2	max pooling, kernel size: 33, stride: 2
conv5	kernel size: 3 × 3, output number: 64, stride: 1, pad: 1
conv6	kernel size: 3 × 3, output number: 64, stride: 1, pad: 1
pooling3	max pooling, kernel size: 3 × 3, stride: 2
fc1	output number: 128
fc2	output number: 128
output	softmax, output number: 10

On the basis of region proposal, we resize the candidate regions to 40 × 40 as inputs of the CNN. The input images has 3 channels (R, G, B), and they are directly feed into the CNN.

The CNN model used for classification totally has six convolution layers. The first convolution layer (conv1) has 32 kernels of size 3 × 3 with a stride 1 pixel (for each convolution layer, we pad the border of input with 1 pixel of zero). The outputs of first convolution layer are 32 feature maps of size 40 × 40. The second convolution layer (conv2) is same with conv1. Then, we perform max pooling with a kernel of size 3 × 3 and 2-pixel stride to subsample the input feature maps, and all of the pooling layers are same. So, the output size of the first pooling layer (pooling1) is 20 × 20. The third and the forth convolution layers (conv3 and conv4) all have 32 kernels of size 3 × 3. Next, the second pooling layer (pooling2) is performed, and the size of the output data is 10 × 10. In the fifth and sixth convolution layers (conv5 and conv6), they both have 64 kernels of size 3×3. The last pooling layer (pooling3) reduce the size of input to 5×5. Then we introduce two fully connected layers (fc1 and fc2) with a output size of 128×1. In the output layer (classifier), we use the softmax function to predict classes.

4.2 Training

We train the CNN on a dataset made from scene images by using labeled region proposals. The dataset contains 10 categories, i.e., red circle (0), green circle (1), red left (2), green left (3), red forward (4), green forward (5), red right (6), green right (7), red negative (8), green negative (9).

During training phase, we use "step" learning rate policy, and the initial learning rate is 0.01. We carry out total 80 epochs for training and the learning rate is reduced by a factor of 10 every 20 epoch. The batch size is 100. Besides, we adopt dropout after all fully connected layers, which keeps each neuron in CNN activating with probability p to prevent overfitting.

Since the negative examples varying tremendously, the number of negative examples should be lager than positive examples in dataset. We increase the number of negative examples gradually while training the CNN to achieve high classification accuracy.

5 Experimental Results

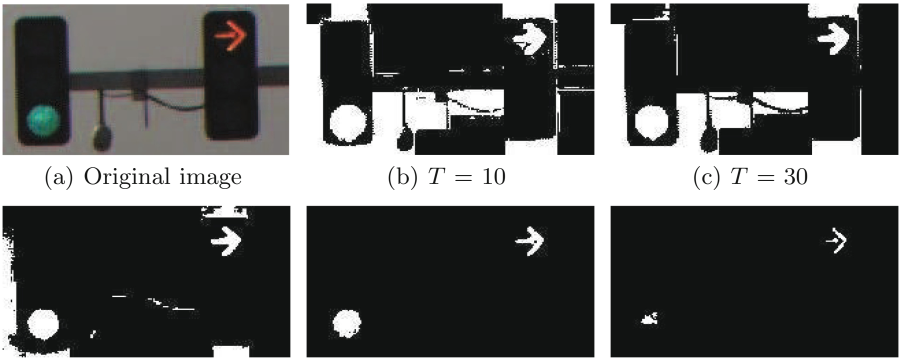
In order to confirm the effect of proposed method, we carried out experiments of classification by CNN model and detecting traffic lights with whole system from scene images separately. We also tested the processing speed of whole system.

5.1 Region Proposal and Classification

During the region proposal stage, we convert the result of Top-hat transform into a binary image with a threshold T according formula (7). We process the image with different threshold value, and the results are shown as Fig. 8.

From Fig. 8 we can see that the backgrounds are reducing with the increasing of threshold. However, the traffic light regions will be removed partially when the threshold is overrange, so that influences the result of region proposal. According to experiments, 50 is the bast value of threshold to obtain the ideal results. For geometry filtration, the interval of S and $Ratio$ are $[144, 2000]$ and $[1, 1.5]$ respectively according to the statistics.

We used 17852 images of traffic lights and negative examples with 10 categories to test the CNN model, and the result of experiment is shown in Table 2.



(a) Original image (b) $T = 10$ (c) $T = 30$

(d) $T = 50$ (e) $T = 70$ (f) $T = 100$

Fig. 8. Result of intensity filtration with different threshold

Table 2. The result of classification

Label	Type	Test image (Frame)	Accuracy (%)
0	Red circle	2000	99.7
1	Green circle	1846	99.9
2	Red left	2000	100
3	Green left	2000	100
4	Red forward	1816	100
5	Green forward	1359	99.7
6	Red right	2000	99.6
7	Green right	561	99.5
8	Negative red	2000	98.8
9	Negative green	2000	98.8
Total	/	**17582**	**99.6**

5.2 Detection

The dataset for experiment of detection includes 6804 images of 10 different scenes. And we mainly consider two evaluating indicators: accuracy and recall, which are defined as (15) and (16).

$$recall = \frac{t}{s} \tag{15}$$

$$accuracy = \frac{t}{u} \tag{16}$$

Where s is the total number of traffic lights in scene images; t is the number of detected traffic lights. u is the number of all candidate regions; The result of detection is demonstrated in Fig. 9, the recall and accuracy are displayed in Table 3.

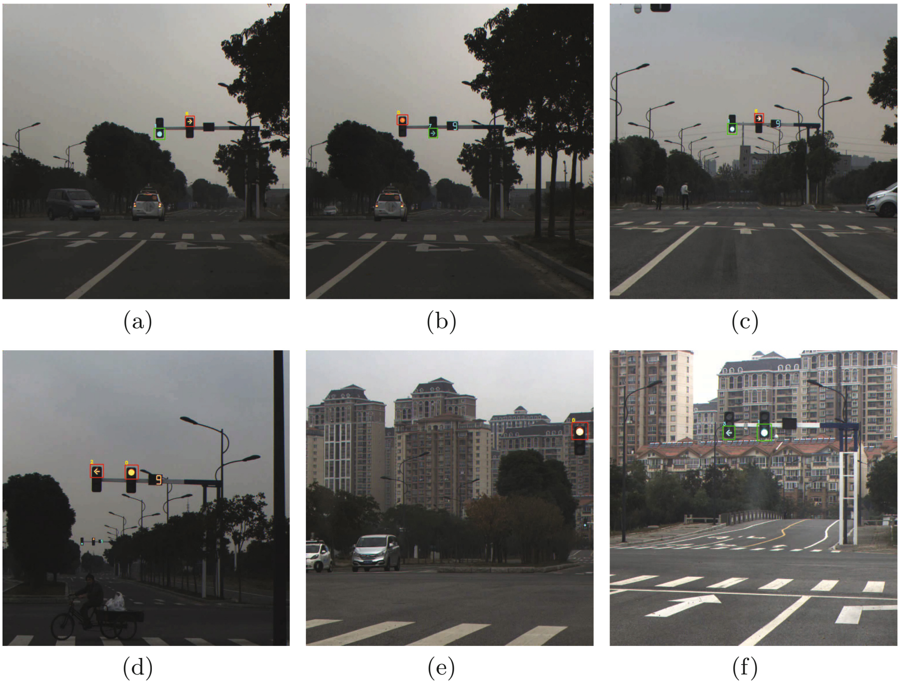


Fig. 9. Result of traffic lights detection

Table 3. The result of detection

Scene	Images	s	Recall (%)	Accuracy (%)
1	733	1344	99.9	99.3
2	427	782	100	99.9
3	636	1160	99.9	98.7
4	632	1181	99.5	99.7
5	202	287	99.0	98.6
6	876	1644	96.6	100
7	1935	2833	99.5	95.8
8	425	747	99.1	98.7
9	659	1153	99.7	99.8
10	279	279	97.5	100
Total	**6804**	**11406**	**99.2**	**98.5**

From Table 3, it is obvious that the recall and accuracy of detection achieve 99.2% and 98.5% respectively. In order to achieve high recall, we can relax some restrictions while performing region proposal. As a result, more negative regions are extracted from scene images. We can notice that the test accuracy of scene 7 is lower than others obviously. By analysing the detection results in scene 7, we found that more than one crossing are in the images, and traffic lights in the second crossing are also detected. However, traffic lights in the second crossing are too small to be classified correctly (Fig. 10(a)). Besides, the background in images are extracted as candidates regions too many so that more negative examples are classified as traffic lights (Fig. 10(b)). Therefore, the structure of the CNN can be more complicated and more training data are needed to improve the classification accuracy of negative regions.

(a) (b)

Fig. 10. Failure result of traffic lights detection

5.3 Time Efficiency

We test 6804 images of size 922×1000 on a 2.1 GHz CPU. The average processing time of two main stages is shown in Table 4.

Table 4. Processing time of traffic lights detection

Main stage	Duration (ms)
Region proposal	39.8
Classification	27.1
Total	**66.9**

Note that the region proposal stage is time-consuming, and the processing time of classification is related to the number of candidate regions.

6 Conclusions

In this paper, we proposed a method of traffic lights detection based on deep learning features. Firstly, we take advantage of the intensity, color, and geometric information of traffic lights to perform region proposal, and obtain candidate regions. In this stage, we employ Gaussian filtration and perform gray processing and Tophat operation to process the intensity information of image. Then we convert the image from RGB into HSI color space, and filter the image according to the hue of traffic light. Next, we restrict the regions' geometrical information to generate candidate regions. Secondly, we introduce a CNN to classify the candidate regions, and combine the location and the result of classification to achieve traffic lights detection.

From the results of experiment, we obtain 99.6% average accuracy of classification by performing the data augmentation for training the network. Besides, the recall and average accuracy of detection can achieve 99.2% and 98.5% respectively, with processing time about 66.9 ms per image. Furthermore, we will optimize the region proposal algorithm to improve the processing speed for real-time application.

References

1. Chen, Z., Yang, J., Kong, B.: A robust traffic sign recognition system for intelligent vehicles. In: 2011 Sixth International Conference on Image and Graphics, pp. 975–980, August 2011
2. de Charette, R., Nashashibi, F.: Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates. In: 2009 IEEE Intelligent Vehicles Symposium, pp. 358–363, June 2009
3. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2155–2162, June 2014. https://doi.org/10.1109/CVPR.2014.276
4. Eric Hague, G.: Color segmentation in the HSI color space using the k-means algorithm. In: Proceedings of the SPIE - The International Society for Optical Engineering, vol. 3026 (1997)
5. de la Escalera, A., Armingol, J., Mata, M.: Traffic sign recognition and analysis for intelligent vehicles. Image Vis. Comput. **21**, 247–258 (2003)
6. Espinosa, J.E., Velastin, S.A., Branch, J.W.: Vehicle detection using alex net and faster R-CNN deep learning models: a comparative study. In: Badioze Zaman, H., et al. (eds.) IVIC 2017. LNCS, vol. 10645, pp. 3–15. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70010-6_1
7. Girshick, R.: Fast R-CNN. arXiv e-prints arXiv:1504.08083, April 2015
8. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv e-prints arXiv:1311.2524 (Nov 2013)

9. Horak, R., Mattioli, J.: Target detection of very dim objects using gray-level morphologic tophat (1996)
10. Hosang, J., Benenson, R., Dollár, P., Schiele, B.: What makes for effective detection proposals? IEEE Trans. Pattern Anal. Mach. Intell. **38**(4), 814–830 (2016)
11. Hosang, J., Benenson, R., Schiele, B.: How good are detection proposals, really? (2014)
12. Jensen, M.B., Philipsen, M.P., Møgelmose, A., Moeslund, T.B., Trivedi, M.M.: Vision for looking at traffic lights: issues, survey, and perspectives. IEEE Trans. Intell. Transp. Syst. **17**(7), 1800–1815 (2016)
13. John, V., Yoneda, K., Liu, Z., Mita, S.: Saliency map generation by the convolutional neural network for real-time traffic light detection using template matching. IEEE Trans. Comput. Imaging **1**(3), 159–173 (2015)
14. John, V., Yoneda, K., Qi, B., Liu, Z., Mita, S.: Traffic light recognition in varying illumination using deep learning and saliency map. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 2286–2291, October 2014
15. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
16. Leng, B., Yu, K., Liu, Y., Jingyan, Q.: Data augmentation for unbalanced face recognition training sets. Neurocomputing **235**, 10–14 (2016)
17. Li, Y., Cai, Z.X., Gu, M.Q., Yan, Q.Y.: Traffic lights recognition based on morphology filtering and statistical classification, vol. 3, pp. 1700–1704 (2011)
18. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
19. Mas Montserrat, D., Lin, Q., Allebach, J., Delp, E.: Training object detection and recognition CNN models using data augmentation. Electron. Imaging **2017**, 27–36 (2017)
20. Omachi, M., Omachi, S.: Traffic light detection with color and edge information, pp. 284–287, January 2009
21. Ouyang, W., Wang, X.: Joint deep learning for pedestrian detection. In: 2013 IEEE International Conference on Computer Vision, pp. 2056–2063, December 2013
22. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. arXiv e-prints arXiv:1506.02640, June 2015
23. Rotaru, C., Graf, T., Zhang, J.: Color image segmentation in HSI space for automotive applications. J. Real-Time Image Process. **3**(4), 311–322 (2008)
24. Shi, Z., Zou, Z., Zhang, C.: Real-time traffic light detection with adaptive background suppression filter. IEEE Trans. Intell. Transp. Syst. **17**(3), 690–700 (2016)
25. Soille, P.: Morphological Image Analysis-Principles and Applications, vol. 49. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-662-05088-0
26. Tran, T.H., Pham, C.C., Nguyen, T.P., Duong, T.T., Jeon, J.W.: Real-time traffic light detection using color density. In: 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), pp. 1–4, October 2016
27. Uijlings, J.R.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. Int. J. Comput. Vis. **104**, 154–171 (2013)
28. Wang, C., Jin, T., Yang, M., Wang, B.: Robust and real-time traffic lights recognition in complex urban environments. Int. J. Comput. Intell. Syst. **4**, 1383–1390 (2011)