Proceedings of the 5th International Symposium on Future Active Safety Technology toward Zero Accidents
(FAST-zero '19)
September 9-11, 2019, Blacksburg, VA, USA

Efficient Traffic Light Detection for Autonomous Vehicles

Wael Karkoub & Srikanth Saripalli

Department of Mechanical Engineering
Texas A&M University

College Station, Texas, 77843, USA
E-mail: wael.karkoub@tamu.edu
Co-author's E-mail: ssaripalli@tamu.edu

Keyword(s): Blob Detection, XGBoost, HOG, HSV

Abstract

Autonomous vehicles are quickly becoming a reality, expected to increase the safety and the efficiency of transportation infrastructure. One important factor of autonomous vehicles is their ability to detect traffic lights accurately at longer distances as they approach an intersection. We have developed an efficient algorithm based on XGBoost that is able to detect traffic lights as far as 45 m away from an intersection. We evaluated our algorithm on approximately 5000 frames, which were labeled according to their distances. The overall precision and recall were found to be 99.8% and 88.1%, respectively, and the algorithm can make the appropriate decision with 91% accuracy. The proposed algorithm can be executed on a CPU at 13 fps.

I. Introduction

Developing autonomous vehicles that can be safely deployed in urban environments is by no means an easy feat. There are many anomalies in the real-world that can not be simply replicated in a simulated environment to test out new solutions. One of the biggest challenges with developing a fully autonomous vehicle is detecting traffic lights of the approaching intersection. Failure to do so may result in catastrophic accidents. However, this problem has been tackled by other researchers utilizing different techniques and a representative few are listed below.

Advances in machine learning algorithms have aided in developing robust solutions for traffic light detection. Researchers have been using conventional deep neural networks and convolutional neural networks as their classifiers to detect traffic lights with great accuracy [1–4]. For instance, [3] proposed a traffic light detection system that utilized neural networks. In their system, the "You Only Look Once" (YOLO) neural network architecture was utilized to detect the traffic lights [5]. The architecture, however, was adjusted to detect smaller objects. A secondary convolutional neural network was also designed to detect the current state of the traffic light. Once the traffic light was detected, its position was tracked using the linear triangulation method. Their system did prove to be effective and robust according to their experimental results, however, neural networks are computationally intensive and expensive and will require specialized hardware for it be effective in the real-world environment. The second approach that researchers tend to explore more often are vision-based algorithms [6–8]. These processes tend to be similar to one another; the candidate regions are first extracted through either color thresholds in the RGB [6], HSV [9], or LAB [7] color spaces. The edges of the traffic lights are then found using either Sobel filters [10] or Canny edge [11]. Then to classify the candidate regions, template matching [12, 13], or in recent years, Support Vector Machine [8, 9] were used to detect the current state of the traffic lights.

We propose a traffic light detector that does not require any extra specialized hardware for it to be successful in a real-world environment and evaluation metrics to evaluate the effectiveness of the detector. We first describe the candidate region extraction in Section II, discuss the feature selection and classification in Section III, and then present the experimental results in Section IV.

A. Algorithm Architecture

The algorithm presented in this paper consists of two major stages. The first stage of the algorithm is the candidate region extraction stage, where image processing is used to locate the possible regions of traffic lights. These

estimated regions are then classified using machine learning algorithms to determine whether the highlighted regions are red, green, yellow traffic lights or if it is not a traffic light at all.

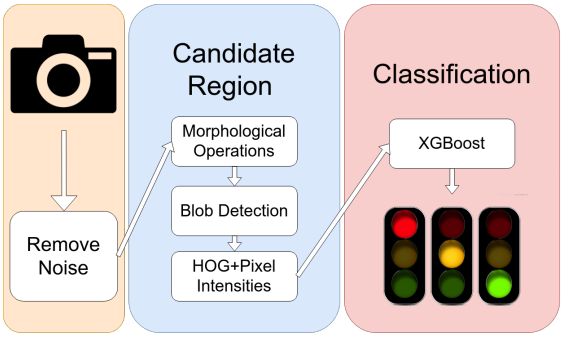

Figure 1: Overall flow chart of the algorithm architecture

II. Image Processing

The challenge behind this algorithm is to develop a real-time traffic light detector while keeping it CPU-bound. Thus, the suggested processes are selected in order to minimize computational speeds while maximizing the performance of the detector. The image processing conducted in this algorithm consists of de-noising the image, changing the color space to HSV, applying morphological operations, and detecting blobs using a method proposed by Suzuki and Abe [14]. The proposed algorithm was coded in Python using OpenCV [15] for image processing.

A. Image De-noising

The first step is to reduce the amount of noise picked up by the camera sensors using de-noising kernel techniques. Preserving the edges while de-noising the image was an important consideration due to the fact that the majority of traffic lights encountered are circular in shape. Therefore, two de-noising techniques were experimented; Bilateral Filtering and Median Filtering. As opposed to Gaussian and Mean filtering, these filtering techniques are typically used to reduce the noise level in an image while maintaining the present geometries in the image. For this algorithm, the median filter was selected over the bilateral filter due to its significantly faster execution and minimal performance difference [16]. Figure 2 Illustrates the effects of median filtering on a noisy image.

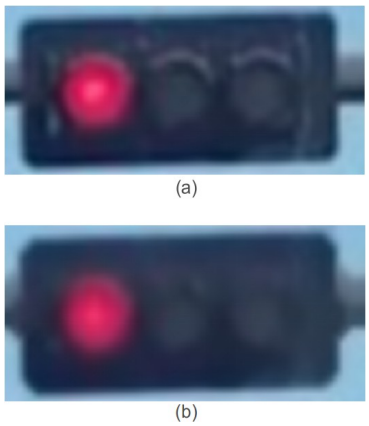

Figure 2: (a) Original image obtained from the camera. (b) De-noised image using median filter.

B. Color Space Conversion

Two critical pieces of information can be obtained from traffic lights that can be used for region proposals and for classification purposes. The first is the color information that can be extracted from the images, while

the second is the illumination of the colors in the image. Therefore, to develop a robust system that can behave consistently under different lighting and weather conditions, these two variables must be controlled independently of each another. The RGB color space does not provide necessary information to provide us with such control, since it does not separate the color information (chroma) from the luminosity information (luma). The HSV color space, on the other hand, separates the chroma and the luma of each pixel in the image. Where H is the hue of the pixel, S is the saturation of the color, and V is the luminosity of the color. Figure 3 illustrates the difference between the gray scale of traffic lights in RGB and the gray scale in HSV color space.

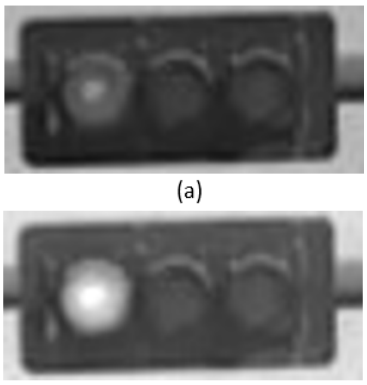

Figure 3: (a) Gray scale of a red traffic light in the RGB color space. (b) Gray scale of a red traffic light in the HSV color space.

Regions of interests are extracted by color and pixel intensity segmentation. After converting the image to HSV color space, thresholds for H, S, V were determined to transform the image into a binary image [9]. Figure 4 illustrates the results after segmentation of the image.

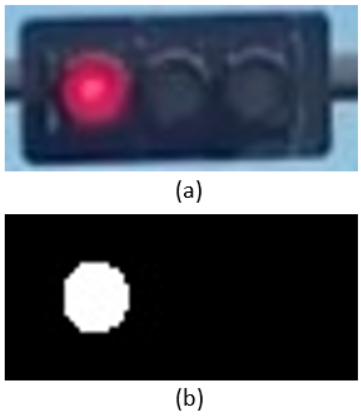

Figure 4: (a) Image captured from the camera. (b) Color segmentation applied to the image.

C. Morphological Operations

In real-world environments, traffic lights rarely emit constant red or green colors, and some of them may even be broken, creating an inconsistent binary image of the traffic lights. To mitigate this concern, morphological operations are applied to the frames. Multiple morphological techniques were tested and compared for speed execution, geometry preservation, and gap filling of traffic lights. It was found that applying the morphological closing, a process of applying dilation followed by erosion, yielded the best performance which met all the aforementioned criteria. One of the parameters that needed to be tuned to maximize the performance of the algorithm is the kernel size and geometry known as the structural elements. A kernel is an nxn size matrix used for image blurring, sharpening, or any other image processing. Since traffic lights are mainly circular in shape, the selected structural element is circular with an element size of 5. Figure 5 is an illustration of the effects of morphological closing.

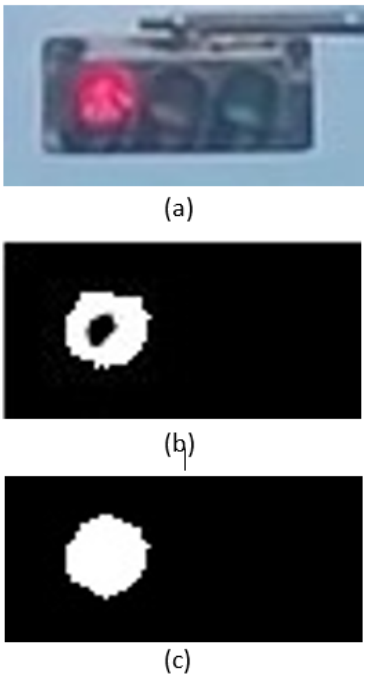

Figure 5: (a) Red traffic light captured by the camera. (b) Color segmentation of the image. (c) Morphological closing operation was applied to the image in (b).

Color segmentation of the image and binarization may result in unwanted "specks" that will increase the number of proposals fed to the classification algorithms. This in turn slows down the algorithm's speed. To alleviate the issue, applying morphological opening, a process of applying erosion followed by dilation, proved to reduce the number of unwanted proposals. The selected structural element is circular with an element size of 3.

D. Blob Detection

In this proposed algorithm, blob detectors are perfect the tools to detect emitted traffic lights. There are multiple techniques that are currently being deployed for blob detection, most prominently Laplacian of Gaussian (LoG), Difference of Gaussian (DoG), Determinant of Hessian (DoH), and Circle Hough Transform. After experimenting with these algorithms, it was found that while the aforementioned algorithms performed well when detecting the traffic lights, their implementation was too slow to be considered real-time. Suzuki and Abe presented in their paper a different approach for detecting blobs that is computationally efficient [14]. Their proposed approach performed poorly for small blobs (traffic lights at far distances), however, performed equally well for closer distances while speeding up the algorithm by at least five folds. This trade-off is justifiable as the classifiers used cannot classify the traffic lights correctly from this distance. Figure 7 shows the blob detection with a bounding box.

Figure 6: Detected blob by the algorithm.

After obtaining the bounding boxes for all the blobs detected in the image, the traffic light bounding box can be estimated through calibration. From the blob's bounding boxes, the radius and center point of the blob can be estimated. The blobs are then categorized into two different color groups, green or red. Separating the blobs into different groups, such as green and red traffic lights allows for generating the appropriate bounding boxes for

the traffic lights. This was the final step for image processing. The images that are encapsulated by the bounding boxes are then fed into the classification algorithm.

III. Classification

A. *Dataset*

The training data for this experiment was collected through a Samsung S8 camera, which is equipped with a 12 MP image sensor. The resulting image has a resolution of 2560 x 1440 pixels. The phone was placed on the dashboard a moving car at an angle approximately parallel to the ground. The recording rate was set to 25 fps while utilizing the optical image stabilizer. All video sequences recorded were in the city of College Station, Texas. The video sequences were recorded in three different lighting conditions: early in the morning, middle of the day when strong light is present, and at dusk. For the frames that were deemed to be useful and meaningful to be used for classification, a python application called LabelImg was used to draw a bounding box and label each traffic light manually [17]. There are 1200 green, 661 red, 200 yellow traffic lights, and 1000 negative backgrounds labeled for training. Due to the infrastructure in College Station, only horizontal traffic lights were collected and used for this experiment.

B. *Feature Selection*

Traffic lights can be described with two characteristics: color and their geometry. The color aspect of the traffic light is too computationally intensive and expensive to perform in real time. Three data points must be used for each pixel. To tackle this problem, the intensity of each pixel is used in the calculations. Intuitively, green traffic lights will have higher intensity pixels in the shape of a circle or an arrow in the left most one-third of an image. The yellow traffic light will have higher intensity pixels in the center of the image. Lastly, red traffic lights will have higher intensity pixels in the right most one-third of an image. By doing so, the number of features were reduced by a factor of three. Histogram of Oriented Gradients (HOG) has proven to be an excellent tool to capture the geometry of the traffic lights. Researchers have used HOG as a feature descriptor to detect pedestrians and traffic lights [18, 19].

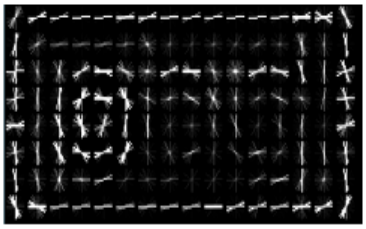

Figure 7: Histogram of Oriented Gradients applied to a red traffic light.

Using the intensities and HOG together has yielded better results than if a classifier used just one of the feature sets independent of the other.

C. *Classifier*

Execution speed and accuracy are the two main criteria that this algorithm aims achieve. Many of the world's leading companies in autonomous vehicles development, such as Waymo, use Deep Learning techniques for traffic light detection [4]. More notably, the Convolutional Neural Network (CNN) are considered state-of-the-art classifiers with extremely high accuracy and generalization ability. However, for what these techniques make up in accuracy, they lack in execution speed. Faster-RCNN was used for experimental purposes; the algorithm on average was executed at 0.2 fps on a CPU. The second main downfall of CNNs is that they require large amounts of data to be effective. Due to the lack of resources, this was not feasible. With that being said, the classifier of choice was XGBoost [20]. It is a classifier based on the concept of Gradient Tree Boosting algorithms. This classifier has many advantages including: negligible computation cost, it requires low number of training samples to be effective compared to other methods such as CNN to be effective, and it has better generalizability than other classical machine learning approaches such as Support Vector Machine. The power of XGBoost has been proven by many of the data science competition winners such as Kaggle [21]. As with many machine learning algorithms, parameters must be optimized to attain the best possible results. Genetic algorithms were used for this purpose.

IV. Results

For this experiment, a Toshiba Satellite laptop with a 4-core Intel I7 at 2.6 GHz was used to run the algorithm. 4967 frames were used to assess this algorithm. Many researchers claim real-time traffic light detection but did not provide analyses of the algorithms regarding its effectiveness in long and short distances [1, 4, 6, 12, 19]. It is important to provide metrics that will aid in the understanding of whether the algorithm is effective in real-world scenarios. For this reason, this paper proposes evaluation metrics to assess traffic light and road sign detection algorithms. The first metric is to analyze the frames based on their distances, the second is the ability for the algorithm to make a decision, and the third metric is the distance the vehicle traveled before the next frame is analyzed. The frames are then categorized based on the distances between the vehicle and the stopping point of the intersection: 967 frames for distances between 45 m and 30 m, 858 frames for distances between 30 m and 15 m, and 3140 frames for distances between 15 m and the stopping point. Table 1 shows the precision and recall calculated for each category. Table 1 also tabulates the second proposed metric. The criteria for the second metric is that there must be at least one traffic light visible to the camera, no false positives, and at least one true positive. These criteria can determine the ability of the algorithm to make a decision.

Table 1: Analyzed performance of the algorithm

Distance	Number of Frames	Precision	Recall	Able to make a decision
45 m - 30 m	967	0.995	0.884	0.720
30 m - 15 m	858	0.992	0.946	0.744
15 m - 0 m	3140	0.999	0.873	0.990
Overall	4977	0.998	0.881	0.911

It is evident that as the distance increases the ability for the algorithm to make an appropriate decision gets worse. This can be attributed to the low quality images captured by the camera. The frames were analyzed at 1920 x 1080 pixels while the vehicle was in motion. As long as there is motion, there is going be motion blur to a certain extent. This is believed to be the main source of error. The algorithm on average can be executed at 13 fps or 81 ms per cycle; however, this information alone is not enough. Table 2 illustrates the distances the car is expected to travel between each cycle.

Table 2: The distance the car travels per calculation cycle

Car Speed	Distance (m)
35 mph (15.65 m/s)	1.26
45 mph (20.12 m/s)	1.62
55 mph (24.59 m/s)	1.98

In some instances, the algorithm misclassified the red traffic light with a green light. This is due to the fact that the U-Turn sign was in the candidate region and was much brighter than the red light. Tail lights of the vehicle are sometimes classified as a red traffic light. This problem was expected to happen as many classification algorithms fail to eliminate this issue. One of the ways to combat this challenge is to train the machine learning algorithm with more negative data. Other situations that the algorithm misclassified is when light sources are reflected from the reflective surfaces. The color segmentation's thresholds for the red color is close to that of the yellow color, resulting in the algorithm tending to fail differentiating between the two traffic lights. There are some situations in which the candidate region is even difficult for humans to classify. For example, the pedestrian cross-walk lights were classified as red traffic lights. With a higher resolution camera, the HOG will clearly be able to differentiate between the red traffic light and the cross-walk lights. And lastly, the candidate regions of the traffic lights cannot be appropriately determined due to the sky color being closer to red.

V. CONCLUSIONS

A new traffic light detection algorithm for autonomous vehicles based on XGBoost is proposed here. The proposed algorithm aimed to reduce the computational cost of detecting traffic lights while maximizing the range and reliability. A median filter is applied to images to reduce the noise present in them. Converting the color space from RGB to HSV has proven to be beneficial since the light intensity and the color information are separated. A morphological closing followed by morphological opening is recommended to remove any unwanted specks while maintaining the integrity of the traffic lights. To detect blobs, the methods proposed by Suzuki and Abe has

proven to be computationally efficient and reliable. The feature sets that were selected for classifications are the pixel intensities and the HOG feature descriptor. XGBoost has shown the best results when compared to other classical machine learning techniques. The results show that the proposed algorithm is computationally efficient and accurate. It is worth noting that, the proposed algorithm can be improved in different ways. Humans do not only depend on vision for detection, depth perception and context play a big role in decision making as well. To remedy these issues, information retrieved from the camera can be cross-calibrated with LiDAR sensors to filter out the false positives. Google Maps API can also be used to give the algorithm context. For example, if the vehicle is on the highway or if the intersection is too far away, the algorithm is not expected to detect any traffic light. And finally, as with any machine learning algorithm, more data will help improve the detection accuracy [22].

References

[1] G. G. Lee and B. K. Park, "Traffic light recognition using deep neural networks," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 277–278, Jan 2017.

[2] V. John, K. Yoneda, Z. Liu, and S. Mita, "Saliency map generation by the convolutional neural network for real-time traffic light detection using template matching," *IEEE Transactions on Computational Imaging*, vol. 1, pp. 159–173, Sept 2015.

[3] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1370–1377, May 2017.

[4] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan, "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 606–611, June 2017.

[5] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[6] M. Omachi and S. Omachi, "Traffic light detection with color and edge information," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 284–287, Aug 2009.

[7] A. F. Said, M. K. Hazrati, and F. Akhbari, "Real-time detection and classification of traffic light signals," in *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–5, Oct 2016.

[8] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 690–700, March 2016.

[9] Z. Chen and X. Huang, "Accurate and reliable detection of traffic lights using multiclass learning and multi-object tracking," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, pp. 28–42, winter 2016.

[10] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 358–367, Apr 1988.

[11] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679–698, Nov 1986.

[12] C. Wang, T. Jin, M. YANG, and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," vol. 4, 12 2011.

[13] G. Siogkas, E. Skodras, and E. Dermatas, "Traffic lights detection in adverse conditions using color, symmetry and spatiotemporal information," in *VISAPP*, 2012.

[14] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, p. 396, 1985.

[15] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[16] T. Sun and Y. Neuvo, "Detail-preserving median based filters in image processing," *Pattern Recognition Letters*, vol. 15, no. 4, p. 341–347, 1994.

[17] Tzutalin, "Labelimg."

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, June 2005.

[19] X. Zhou, J. Yuan, and H. Liu, "Real-time traffic light recognition based on c-hog features," *Computing and Informatics*, vol. 36, no. 4, p. 793–814, 2017.

[20] T. Chen and C. Guestrin, "Xgboost," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*, 2016.

[21] Kaggle, "kaggle: your home for data science," 2018.

[22] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, p. 78, Jan 2012.