TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 1, February 2021, pp. 173∼181
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i1.16281

Traffic sign detection optimization using color and shape segmentation as pre-processing system

Handoko, Jehoshua Hanky Pratama, Banu Wirawan Yohanes
Department of Electronic and Computer Engineering, Universitas Kristen Satya Wacana, Indonesia

Article Info

Article history:

Received Apr 8, 2020
Revised Jun 13, 2020
Accepted Jul 6, 2020

Keywords:

Color segmentation
Optimization
Shape segmentation
Traffic sign detection

ABSTRACT

One of performance indicators in an autonomous vehicle (AV) is its ability to accommodate rapid environment changing; and performance of traffic sign detection (TSD) system is one of them. A low frame rate of TSD impacts to late decision making and may cause to a fatal accident. Meanwhile, adding any GPU to TSD will significantly increases its cost and make it unaffordable. This paper proposed a pre-processing system for TSD which implement a color and a shape segmentation to increase the system speed. These segmentation systems filter input frames such that the number of frames sent to artificial intelligence (AI) system is reduced. As a result, workload of AI system is decreased and its frame rate increases. HSV threshold is used in color segmentation to filter frames with no desired color. This algorithm ignores the saturation when performing color detection. Further, an edge detection feature is employed in shape segmentation to count the total contours of an object. Using German traffic sign recognition benchmark dataset as model, the pre-processing system filters 97% of frames with no traffic sign objects and has an accuracy of 88%. TSD system proposed allows a frame rate improvement up to 32 frame per second (FPS) when You Only Look Once (YOLO) algorithm is used.

Corresponding Author:

Handoko
Department of Electronic and Computer Engineering
Universitas Kristen Satya Wacana
Diponegoro 52-60, Salatiga, Central Java, Indonesia
Email: handoko@uksw.edu

1. INTRODUCTION

In industrial revolution 4.0 era, research in autonomous vehicle (AV) are extensively encouraged to build a modern and safer transportation. An AV which is equipped with an enormous number of sensors (infrared, camera, proximity, etc) allows sensitive sense for the change of its surrounding to provide maximum safety for human being. Since it is controlled by computational devices without any human as a driver, AV is claimed to lower down the accident rate by 90% [1].

One of the crucial parts in an AV is its detection system, which has to be very responsive to any changes of its surrounding, including traffic signs. The speed of processing in the detection system, in term of frame rate number, will define correct or incorrect decisions. The low frame rate indicates slow response of the traffic sign detection (TSD) system to respond to the external changes including the existence of traffic signs. Since 2015, a number of accidents which involve AVs had been reported and the low frame rate was responsible for most of these accidents. Most of the accidents was caused by the system failure of the AV to

make a fast and accurate decision [2, 3]. Therefore, a good TSD system must provide a frame rate of minimum 25 frame per second (FPS) in order to react instantly but still preserve its accuracy.

A high frame rate TSD can be obtained in two ways either using additional graphics processing unit (GPU) or by improvement of its algorithm. An additional GPU will instantly boost the frame rate of TSD since the detection process will be performed using additional processors and memories. However, GPU extension are costly and as the consequence the vehicle price will not affordable. In contrast, improvement of algorithm for the current system may produces a higher speed TSD with a lower cost.

As one of the detection systems, the TSD system requires an extensive computation resources at the server and produces a poor frame rate. Meanwhile, not all frames sent for processing at the detection system contains traffic sign objects. Therefore, application of a preliminary system which is able to determine frames to pass for processing at the artificial intelligence (AI) system and which ones should be denied, will significantly increase TSD frame rate. Since the traffic signs have a specific characteristic in colors and shapes, the segmentation of color and shape systems are good option to perform frame filtering in the pre-processing system and allows a lower workload for artificial intelligence system [3].

2. PREVIOUS WORK

Traffic sign detection is typically classified into three fundamental categories: color-based, shape-based and learning based methods [4]. Meanwhile, Liu [5] add two mode categories: color and shape based methods and light detection and ranging (LIDAR) based methods. Among these methods, detection system based on learning is the best because it achieves a detection rate to more than 90%. Using a deep learning method such as convolutional neural network (CNN) even increases the detection rate to more than 98% [6]. However, the existing algorithms are still far from a real time classified system [4]. Balado [7] proposed an algorithm to achieve a better detection rate, but still could not reach a real time processing frame rate. Research on traffic sign detection based on CNN proposed by Wang [8] produced a detection rate of 5 FPS which is far from the real time processing standard. A multi-scale cascaded R-CNN is proposed by Zhang [9] and Liu [10] to overcome the problem of false detection when traffic signs are in small sizes. Moreover, a real time object detection named YOLO and its modification are often used to tackle the real time TSD problems [9, 11, 12]. Sphurti More in [13], found that YOLO showed a better accuracy than support vector machine (SVM) based algorithm, although it run slower.

On the other hand, classification using color-based and shape-based algorithms are usually fast but they perform a lower detection rate which is not suitable for AV system. Color-based segmentation algorithms can be applied for TSD since the traffic signs have specific characteristic colors: red, blue and yellow [14]. The red sign is used for signs which represent a danger or prohibition, yellow sign shows construction signs and blue sign is for information [14]. Color thresholding, region growing, color indexing, dynamic pixel aggregation are most popular color-based detection methods [15]. However, the variation of color due to the age of signs, variation of light makes color-based segmentation algorithms produce low detection rates [4]. Some researches on color segmentation algorithms show that using HSV as the color space produces better results although it requires a longer processing time [16, 17, 18].

Meanwhile, shape-based segmentation algorithms are commonly used to identify objects. In the TSD system, there are four specific shapes to identify since traffic signs can be classified into rectangular, circular, octagonal and triangular signs. Wali [15] listed a number of shape-based methods such as: Hough transformation, similarity detection, distance transform matching, edge detection features and Haar-like features. The combination of an adaptive color threshold segmentation method and a shape symmetry detection algorithm was proposed in [19] to get higher detection rate in TSD.

3. PROPOSED SYSTEM

Based on the performance of color-based, shape-based and learning-based method discused earlier, we proposed a system to increase the detection rate of learning-based TSD by application of a frame filtering system. In the real implementation of TSD, the number of frames without any traffic sign object typically more than the ones with traffic signs. Therefore, processing only the frames which contain traffic signs will significantly increase its overall detection rate. The proposed system consists of two important processes: pre-processing and AI system. Figure 1 shows the diagram of proposed TSD system. The pre-processing part applies a color segmentation and a shape segmentation algorithm. They are used to pass only the candidate

frames which contain any traffic sign to the AI system. Then, the system will continue the processing to detect whether the candidate frame is actually having traffic sign in it by application of learning-based algorithm (AI).

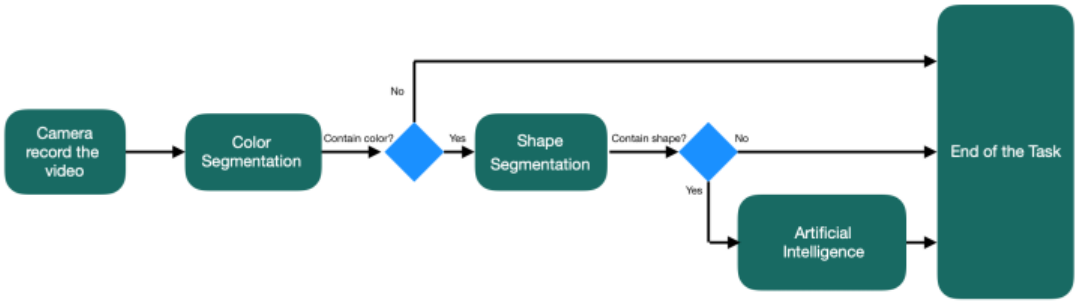

Figure 1. The proposed AV system diagram

Input from camera is resized using a bilinear interpolation algorithm which performs linear interpolation algorithm twice in different directions. This method uses four nearest point to calculate the point in the center of four points chosen [20, 8]. Resizing is used to reduce the number of iteration when performing convolution in a frame. Bilinear interpolation is chosen because it produces a smoother interpolation compares to the other resizing methods [21, 22]. Since traffic signs have specific characteristic of shapes and colors, we limit the detection of shapes for square, triangle, circle and octagon; and the detection of colors for red, blue and yellow. A frame which has none of these shapes and colors can be classified as a frame without any traffic sign, then it can be excluded from further processing.

3.1. Color segmentation

In this work, we use HSV threshold for color segmentation to identify the desired color available in a frame. Before being processed by HSV threshold, the particular frame is filtered using Gaussian blur to smoothen and enhance the frame's color feature. It removes unwanted salt noises of the images and makes the color detection easier and more accurate. Gaussian blur uses the kernel window to perform blurring of an image. The blur level can be adjusted by modification of kernel size. The bigger kernel size will produce a higher blur of image which is unlikely can be detected correctly [23]. Figure 2. shows the 5x5 Gaussian blur kernel window.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figure 2. Gaussian blur kernel window [24]

HSV color space is used because the color can be detected according to its hue and value without considering the saturation of the color, and according to the previous works, using the HSV color space showed a better performance than RGB. The HSV threshold is set to the color desired such as red, blue and yellow. The upper and lower bound of the thresholds are chosen based on the experiment done by Adrian from PyImageSearch [25]. The threshold [110, 50, 50] to [130, 255, 255] is used for blue color, [20, 100, 100] to [30, 255, 255] is for yellow color. Meanwhile, the threshold [160, 20, 70] to [190, 255, 255] is for red color [25]. Algorithm 1 shows an implementation of color segmentation method.

3.2. Shape segmentation

In this work, we use an edge detection feature method for shape segmentation of the frames. A morphological filter is employed to reduce the number of noises in a frame and to add pixels between important

features. This technique uses a nonlinear operation [26]. There are five morphological filter operations: *dilation*, *erosion*, *opening*, *closing* and *hit or miss* transform. *Dilation* and *erosion* are the main operations in the morphological filter. Meanwhile, the *opening*, *closing* and *hit or miss* transforms are formed by the process of combination between *dilation* and *erosion* [27].

Dilation is an operation which can thicken the structure of object in a frame by adding extra pixels between important features. The 1 is used for dilation operation [28]. In addition, the *erosion* is an operation which in contrast to the *dilation* operation. In this operation, the structure of an object will be reduced by elimination of pixels around the object. The 2 shows the equation of *erosion* operation [28].

$$\oplus H \equiv (p+q) | \text{for every } p \in I, q \in H \tag{1}$$

$$\ominus H \equiv p \in \mathbb{Z}^2 | (p+q) \in I, \text{for every } q \in H \tag{2}$$

These operations are used to enhance the features of an object inside the frame. Morphological filter allows an easier detection of the objects in an image. After the frame being processed by morphological filter, the contour of object in the frame is detected and counted. The Douglass Peucker algorithm is used to count the contour of object. It reduces the number of curve in an image and transform it to a straight line between two points. As a result, the contour of object can be identified easier [29]. If the number of contours is 4, then the object is classified as a rectangle object. The total contours of 3, 15 and 8 represent a triangle, circle and octagon object repectively. The frame which is classified having one of the desired shapes will be processed by the AI system. Algorithm 2 shows an implementation of the shape segmentation method.

Algorithm 1 Color segmentation

```
Set threshold for red, yellow and blue colors
while !(end of frame) do
    Get a frame from input camera
    Apply Resizing for the current frame
    Apply a Gaussian Blur filter to the frame
    if (the frame contains at least one color desired) then
        Send the current frame to the shape segmentation system
    else
        get the next frame
    end if
end while
```

Algorithm 2 Shape segmentation

```
Set desired contour values
while !(end of frame) do
    Get a frame from color segmentation system
    Apply Morphological filter to the current frame
    Apply a Douglass Peucker algorithm to the frame
    if (the number of contour macth one of the desired shape) then
        Send the current frame to the AI system
    else
        Get the next frame
    end if
end while
```

3.3. Artificial intelligence (AI) system

A frame which is successfully pass pre-processing system is processed by the AI system for *detection* and *classification*. In AI system, the frame is resized to a smaller size to reduce the number of neural network processing. In this work, You Only Look Once (YOLO) algorithm is employed as the processing of the AI system. YOLO is an open source system which has several versions: YOLO, YOLOv2, YOLOv3 and Tiny YOLO. Furthermore, YOLOv2 is used because it is faster and accurate enough for a TSD system [30]. YOLO

has a feature network with 24 convolutional layers and 2 fully connected layers [31]. Convolutional layer is used to extract the important feature in a frame by application of a convolution using 5x5 mask. This mask will keep moving through the available position in a frame to get its important features [32]. The output of this process is a smaller matrix with important feature pixels. Figure 3 describes the process in a convolution layer.

The output of this process is a list of neurons which includes all the important values from the layer. This list will be connected in a fully connected layer and become one final feature map. The final feature map will be compared to the feature map from the model which has been obtained from the training phase. If the feature map of the frame matches to one of the feature maps in the model, then the system is success to detect and classify the particular object [33].

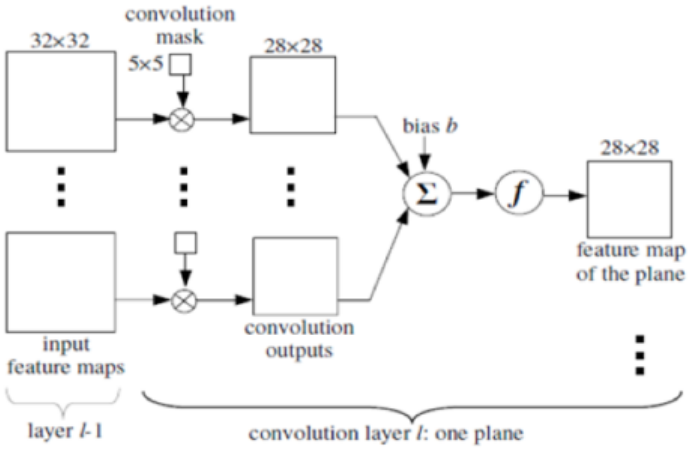

Figure 3. The convolution process [32]

Max pooling will be performed after finishing the convolution process. Max pooling will take the biggest value in the matrix which is assumed as the feature which being searched. Similar to the convolutional layer, max pooling employs a matrix which will move throughout the frame and compare the values in the matrix [34, 35]. Max pooling is chosen because of the condition of the dataset where most of the images have a bright color.

4. RESULT AND DISCUSSION

In this work, we borrow the German Traffic Sign Recognition Benchmark (GTSRB) dataset to construct our model. Figure 4 shows the sample images of the dataset. We divide our experiments into two parts, the pre-processing part and the full system part. The pre-processing experiments are done with 5 different configurations. Meanwhile, the full system is used to measure the frame rate of entire system with and without pre-processing.

Figure 4. Sample of dataset images

4.1. Pre-processing

The first part of the experiments involves 43 traffic signs of the GTSRB dataset. This experiment run with different combination of parameter settings including the *size of open kernel*, *size of close kernel* and *size of Gaussian blur*. In this experiment the still images from the dataset are used as input of the system. Figure 5 shows an example of pre-processing of an image where the color segmentation system catch a yellow object and

the shape segmentation identifies a rectangular sign. Table 1 shows the percentage of the successful detection and filtering of the experiment using 43 traffic signs with various parameter values. The successful detection value represents how good the system can recognize traffic signs by using the shape and color of the signs. Meanwhile, the successful filter value shows the total frame successfully pass to the next processing compare to the total of input frames. As we can see from the result, changes on parameters do effect the accuracy of detection and filtering of the frames.

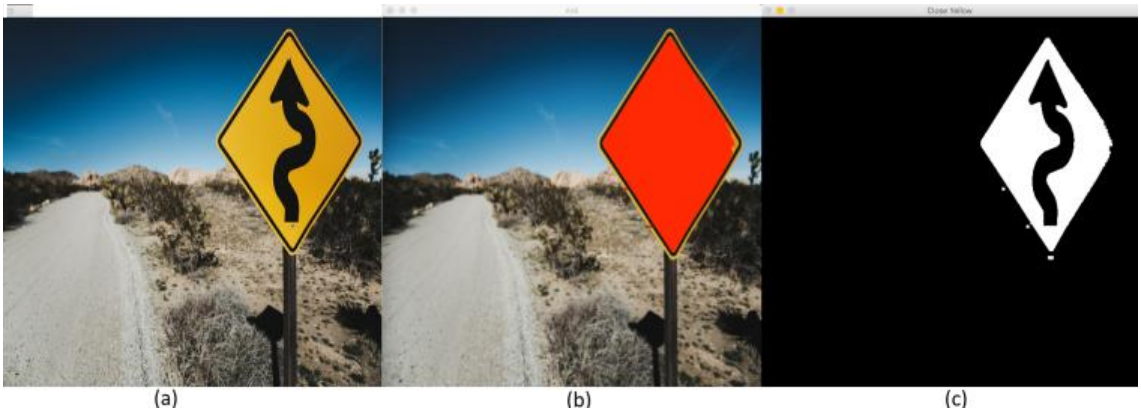

Figure 5. Pre-processing of a frame;
(a) original image, (b) shape and color segmentation result, (c) detection result

Table 1. Pre-processing experiment results

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Open Kernel size	5 x 5 px	11 x 11 px	5 x 5 px	5 x 5 px	7 x 7 px
Close Kernel size	5 x 5 px	5 x 5 px	11 x 11 px	5 x 5 px	5 x 5 px
Gaussian Blur size	3 x 3 px	3 x 3 px	3 x 3 px	5 x 5 px	3 x 3 px
Frame Total	1835 frames	1835 frames	1835 frames	1835 frames	1835 frames
True Positive	47 frames	40 frames	46 frames	53 frames	53 frames
True Negative	9 frames	7 frames	9 frames	11 frames	16 frames
False Positive	1772 frames	1774 frames	1772 frames	1765 frames	1760 frames
False Negative	7 frames	14 frames	8 frames	1 frames	1 frames
Successful Detection Percentage	84%	77%	81%	88%	88%
Successful Filter Percentage	97,05%	97,34%	97,08%	97,18%	96,53%

In experiment 1, the open and close kernel were set to 5x5 and Gaussian blur was set to 3x3. In this configuration, there were a lot of salt noises found in the frame and will produce inefficient frame filtering. In experiment 2, the open kernel size were set to 11x11 and the Gaussian blur was set to 3x3. Using this configuration, we obtain a better result than the first experiment since the open kernel can filter more frames. However the detection accuracy in experiment 2 decreased to 77%. This degradation was caused by the size of the open kernel which is too big and therefore will eliminate some of the important features in the frame.

In experiment 3, the close kernel was set to 11x11 and open kernel to 5x5. The result when using this configuration is slightly higher than the experiment 2 with the detection accuracy up to 81%. It is better because the close kernel capable to close the pixel gaps in the frame which will enhance the feature of traffic sign in a frame. In the next experiment, the Gaussian blur size, kernel open and kernel close were set to 5x5. Using these parameter values enable the system to get a higher percentage in detection accuracy and filtering. The bigger size of Gaussian blur makes the feature in a frame can be easily detected. In the last experiment, the size of open kernel was increased to 7x7 where the close kernel and the Gaussian blur were maintained to 5x5 and 3x3. The result of using these parameter values was not as expected where the performance of filtering frame were lower than experiment 4. Since the optimal results are reached in experiment 4, then these parameters are used for further experiments. Table 2 shows the average time needed to process a single frame in the pre-processing system.

Table 2. Pre-processing time elapse for a single frame

Processing	Average processing time for 1 frame)
Color Segmentation	11.3 ms
Shape Segmentation	0.4 ms
Total pre-processing time	11.7 ms

4.2. Traffic sign detection (TSD) system

In the next experiment, we use some videos as inputs of the TSD system. Figures 6 (a and c) shows an example of processing a video which is taken from a car dashboard. The overall processing time for every processing step and the total frame processed are displayed on the Figures 6 (b and d).

Figure 6. (a, c) TSD with pre-processing system,
(b, d) Number of frames processed and the time processing

In the next step, we measure average time needed to process a single frame. The time processing includes pre-processing system, transfer file, and artificial intelligence system, and can be shown in Table 3. It shows that pre-processing system takes 1.012 seconds in average to process a single frame. Meanwhile, without the pre-processing system takes on average 1.0003 second which includes the transfer file. This means application of pre-processing system increases the time processing of the TSD system up to 1.17% when any candidate traffic sign object is identified. On the other hand, when no candidate traffic sign object is identified, the system takes 0.0117 second which is the pre-processing time only since the transfer file and AI processing are not required. Using 10 frame videos with 10% of the frames having a traffic sign object, we summaries the time processing of the system as shown in Table 4. The processing time is reduced from 10.003 to 1.12 second

(88%) and as the consequences, the frame rate will increase. The result on Figure 6 shows that the FPS values vary from 6.8 FPS to 105.64 FPS. The low frame rate indicates the corresponding frame has any candidate traffic sign object. On the other hand, system will gain a high frame rate when no candidate traffic sign object is detected. The experiment results shows that proposed algorithm with pre-processing system can boost up the speed of TSD system to the average of 32 FPS.

Table 3. Processing time of full system for a single frame

	Average time for processing 1 frame (second)
Pre-processing system	0.0117 s
Transfer file via TCP	0.0003 s
AI System	1.0000 s
Total Time	**1.0120 s**

Table 4. Time processing for 10 frame videos with 10% contains any traffic sign object

	Pre-processing (s)	Transfer File (s)	AI (s)	Total (s)
Without Pre-processing	0	10*0.0003	10*1.0000	10.0030
With Pre-processing	10*0.0117	10*0.0003	1*1.0000	1.1200

5. CONCLUSION AND FUTURE WORK

Using color and shape segmentation as the pre-processing system allows a frame rate improvement of the traffic sign detection system. Using open kernel, close kernel and Gaussian blur matrix size of 5x5 pixels, this system is able to filter up to 97.18% of the frames and pass less than 3% for processing at the AI system. On average, the filtering system reaches 88% detection accuracy. As the number of frames sent to the AI system is reduced, the workload of the artificial intelligence system is decreased and as a result, the frame rate of this detection system increases up to 32 FPS without using any graphics processing unit (GPU). In the future, refinement of color and shape segmentation algorithms can be used to increase the accuracy of pre-processing and therefore improve accuracy of the whole system.

REFERENCES

[1] K. Sergeant, "Autonomous vehicles: Risk management issues and concern," *Lockton Companies*, March 2017.
[2] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: Disengagements, accidents and reaction times," *PLOS ONE*, vol. 11, no. 12, pp. 1–14, 2016.
[3] J. H. Pratama, Handoko, and B. W. Yohanes, "Frame rate optimization in traffic sign detection," in *International Conference on Cybernatics and Intelligent System*, 2019.
[4] Y. Saadna and A. Behloul, "An overview of traffic sign detection and classification methods," *International Journal of Multimedia Information Retrieval*, vol. 6, pp. 1–18, 2017.
[5] C. Liu, S. Li, F. Chang, and Y. Wang, "Machine vision based traffic sign detection methods: Review, analyses and perspectives," *IEEE Access*, vol. 7, pp. 86 578–86 596, 2019.
[6] J. Jiang, X. Feng, F. Liu, Y. Xu, and H. Huang, "Multi-spectral rgb-nir image classification using double-channel cnn," *IEEE Access*, vol. 7, pp. 20 607–20 613, 2019.
[7] J. Balado Frias, E. González, P. Arias, and D. Castro, "Novel approach to automatic traffic sign inventory based on mobile mapping system data and deep learning," *Remote Sensing*, vol. 12, no. 3, pp. 442–457, 2020.
[8] Z. Wang and H. Guo, "Research on traffic sign detection based on convolutional neural network," in *Proc. of the 12th Int. Symposium on Visual Information Communication and Interaction*, 2019.
[9] J. Zhang, Z. Xie, J. Sun, X. Zou, and J. Wang, "A cascaded r-cnn with multiscale attention and imbalanced samples for traffic sign detection," *IEEE Access*, vol. 8, pp. 29 742–29 754, 2020.
[10] Z. Liu, D. Li, S. Ge, and *et al.*, "Small traffic sign detection from large image," *Applied Intelligence*, vol. 50, pp. 1–13, 2019.
[11] X. Liu and F. Xiong, "A real-time traffic sign detection model based on improved yolov3," *IOP Conference Series: Materials Science and Engineering*, vol. 787, p. 012034, 2020.
[12] Q. Xu, R. Lin, H. Yue, H. Huang, Y. Yang, and Z. Yao, "Research on small target detection in driving scenarios based on improved yolo network," *IEEE Access*, vol. 8, pp. 27 574–27 583, 2020.
[13] S. More and J. Bos, "Comparing machine learning and neural network-based approaches for sign detection and classification in autonomous vehicles," in *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2020*, M. C. Dudzik and S. M. Jameson, Eds., vol. 11415, International Society for Optics and Photonics. SPIE, 2020, pp. 129–134. [Online]. Available: https://doi.org/10.1117/12.2558966

[14] N. Mohd Ali, S. Karis, A. F. Zainal Abidin, B. Bakri, E. F. Shair, and N. Razif, "Traffic sign detection and recognition: Review and analysis," *Jurnal Teknologi*, vol. 77, no. 20, pp. 107–113, 2015.
[15] S. Wali, M. Abdullah, M. A. Hannan, A. Hussain, S. Samad, P. J. Ker, and M. Mansor, "Vision-based traffic sign detection and recognition systems: Current trends and challenges," *Sensors*, vol. 19, pp. 2093–2120, 2019.
[16] X. Wang, R. Hänsch, L. Ma, and O. Hellwich, "Comparison of different color spaces for image segmentation using graph-cut," *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 1, pp. 301–308, 2014.
[17] M. Hassan, R. Ema, and T. Islam, "Color image segmentation using automated k-means clustering with rgb and hsv color spaces," *Global Jpurnal Inc. (USA)*, vol. 17, no. 2, pp. 33–41, 2017.
[18] S. Sural, Gang Qian, and S. Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," *Proceedings. International Conference on Image Processing*, vol. 2, pp. II–II, 2002.
[19] X. Xu, J. Jin, S. Zhang, L. Zhang, S. Pu, and Z. Chen, "Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry," *Future Generation Computer Systems*, vol. 94, pp. 381–391, 2018.
[20] P. A. Dilip, K. Rameshbabu, K. P. Ashok, and S. A. Shivdas, "Bilinear interpolation image scaling processor for vlsi bilinear interpolation image scaling processor for vlsi architecure," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 3, no. 3, pp. 104–113, November 2014.
[21] A. C. Bovik, "Chapter 4 - basic binary image processing," pp. 69 – 96, 2009.
[22] X. Wang, X. Jia, W. Zhou, X. Qin, and H. Guo, "Correction for color artifacts using the rgb intersection and the weighted bilinear interpolation," *Appl. Opt.*, vol. 58, no. 29, pp. 8083–8091, Oct 2019.
[23] E. Gedraite and M. Hadad, "Investigation on the effect of a gaussian blur in image filtering and segmentation," pp. 393–396, 2011.
[24] A. Pathmanabhan and S. Dinesh, "The effect of gaussian blurring on the extraction of peaks and pits from digital elevation models," *Discrete Dynamics in Nature and Society*, vol. 2007, pp. 1–12, November 2007.
[25] A. Rosebrock, "Opencv and python color detection," 2014. [Online]. Available: https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/
[26] R. K. Pandey and S. S. Mathurkar, "A review on morphological filter and its implementation," *Internatinal Journal of Science and Research*, vol. 6, no. 1, pp. 69–71, 2015.
[27] A. M. Raid, W. M. Khedr, M. A. El-dosuky, and M. Aoud, "Image restoration based on morphological operations," *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, vol. 4, no. 3, pp. 9–21, June 2014.
[28] R. S and A. M. Khan, "Morphological operations for image processing : Understanding and its applications," *National Conference on VLSI, Signal processing Communications*, pp. 17–19, December 2013.
[29] S.-T. Wu, A. C. G. d. Silva, and M. R. G. Marquez, "The Douglas-peucker algorithm: sufficiency conditions for non-self-intersections," *Journal of the Brazilian Computer Society*, vol. 9, no. 3, pp. 67 – 84, 2004.
[30] J. Zhang, M. Huang, and X. L. Xiaokang Jin, "A real-time chinese traffic sign detection algorithm based on modified yolov2," *Algorithms 2017*, vol. 10, no. 4, pp. 1–13, 2017.
[31] C. Y. Wang and R. C. Yue, "Traffic sign detection using you only look once framework," Stanford University, Tech. Rep., 2016.
[32] J. Deepika, "Image classification using convolutional neural networks," *International Journal of Advancements in Research and Technology*, vol. 6, no. 3, pp. 22–26, 2014.
[33] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *CoRR*, vol. abs/1807.08596, 2018. [Online]. Available: http://arxiv.org/abs/1807.08596
[34] K. Yue, F. Xu, and J. Yu, "Shallow and wide fractional max-pooling network for image classification," *Neural Computing and Applications*, vol. 31, no. 5, p. 409–419, 2017.
[35] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," *The 9th International Conference on Rough Sets and Knowledge Technology (RSKT'14)*, pp. 364–375, 2014.