School Bus Stop Sign Detection

1st Tessa Mitchell
College of Engineering,
University of Missouri
Columbia, United States
tamdcz@umsystem.edu

***Abstract*—Autonomous vehicles currently do not stop for school buses when they have their stop signs extended and lights flashing. This is not only illegal but also a major safety concern. To fix this, I trained a model using an AdaBoost Cascade to first detect school bus stop signs in still images, and then to detect whether the lights on the school bus are illuminated by comparing their intensity and shape to the rest of the image. The trained model was successful in detecting the stop sign in almost all cases, however, the light detection had a very low accuracy rate due to multiple factors.**

I. Introduction

A. Problem

In the field of autonomous vehicles, there has been very little research on the detection of stop signs on school buses, a problem that has become evident in recent years, with the Dawn Project identifying this issue with Tesla Full Self Driving software as early as 2022 [1], and Waymo recalling their software in early December 2025 for illegally passing stopped school buses [2]. Not only is passing stopped school buses illegal in all 50 states, it is a huge safety risk, with at least one child having been hospitalized by being hit with a self-driving Tesla [1]. Standard stop sign detection does not work for school buses because the main difference between school bus stop signs and standard stop signs is that the have two lights that will typically flash in an alternating pattern when extended (perpendicular to the school bus), so detecting a standard stop sign won't work. Training a model to just detect the school bus stop sign could work, however, there are cases where a car might be at an intersection and see a school bus stop sign and try to stop even though the lights are not flashing and the stop sign is not extended (see Fig. 1). Because of this, it is important to either take into account either the positional information of the school bus, or the flashing lights on the stop sign (and potentially the lights on the rest of the school bus), which in this case I chose to take the flashing lights into account.

B. Existing Literature

There is plenty of research on stop sign detection, with methods ranging from traditional image processing to more complex computer vision methods. One traditional image processing technique commonly used was using an AdaBoost Cascade of Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), and Haar-like features [3] [4]. These take advantage of the distinct octagonal shape of a stop sign, as well as features like the text on it.

When it came to detecting the lights on the stop sign, I looked at methodologies used in traffic light detection. In traffic light detection, there are three main lights that are being detected (red, yellow, and green). Because of this, models often start by looking at color, and then at shape, using methods like the Hough Transform, to extract the circular shape of the light [5]. Other methods expand on this by using positional data of the traffic light in relation to the rest of the image to better prune the results [6].

C. Overview

This project relied on only still images and used combined stop sign detection methodology with some traffic light detection methods. Specifically, it used a combination of HOG and LBP features to train an AdaBoost Classifier. The detections that contained stop signs were then run through a light detection that aimed to see if the lights were illuminated using a Hough Transform to detect shape, then verified using intensity and positional information.

Fig. 1. A school bus at an intersection where the stop sign would be detected but it is not loading or unloading. Source: [15]

II. Method

There were two main sections to my method, stop sign detection, then light detection. As part of this, I first had to train the model for stop sign detection, the process of which is represented in Fig. 2. Then there was a separate process for actual taking an image and detecting whether it is of a stop sign and whether the lights on that stop sign are illuminated or not, which is represented in Fig. 3.

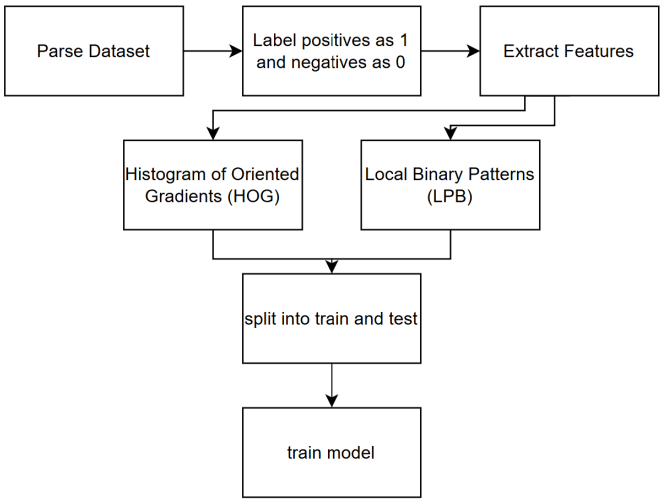

Fig. 2. Flowchart representing the process for training the Adaboost model.

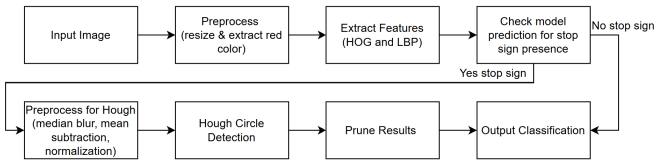

Fig. 3. Flowchart representing the process for detecting stop signs and lights in an image.

A. Input

I created the dataset myself using a collection of images downloaded from the internet. I then uploaded these images to Roboflow to annotate them with positive and negative windows (negatives were random windows of the images that did not contain stop signs). From there, the images were processed to fit into a 1024x1024 image with black edges to keep image's original aspect ratio. I then used Roboflow's augmentation to randomly apply rotation, shear, saturation, brightness, exposure, and noise to the original images, tripling the size of the dataset. I used ChatGPT to figure out what format of Roboflow annotations to export for my use case, and how to parse the XML files with the annotations [7].

B. Preprocessing

The images were extracted from the dataset, separating the positive and negative windows into two collections. Each window was resized to 64x64 (using OpenCV's resize method [8]) and the red color channel was extracted, with the assumption that all stop signs will be primarily red.

C. Feature extraction

The positive and negative windows then went under feature extraction. The first feature used was HOG, to get the shape of the objects in the window, as the octoganal shape of the stop sign is very important in their detection. This was implemented using Scikit-Image's Feature library that has a built-in method to extract HOG features [9]. The other feature used was LBP, which is helpful in analyzing textures of objects in an image. This was also implemented using a method from Scikit-Image's Feature library [10].

D. Model Training

The training method used was Scikit-Learn's AdaBoost-Classifier [11], using the default weak learner of a Decision-TreeClassifier, 50 weak learners to train iteratively, with the weights of those learners being 1 (all the default/standard values). The model was trained off of 70% of the input features, with the other 30% used to test the model accuracy (96.5%).

E. Regions of Interest

Due to issues with processing time, rather than analyzing whole images, I prioritized taking an isolated image of either a stop sign or something random and being able to identify whether it had an illuminated light or not. In this case, those isolated images were the same ones in the dataset. For this, I extracted the features of each image and ran those through the model. If the results were positive, the image then went through the light detection process.

F. Light Detection Preprocessing

The images at this stage had already been preprocessed for the stop sign detection (resized and red channel only). To preprocess the images for the Hough Transform, each image was smoothed using OpenCV's medianBlur method [12] with a kernel size of 7. The images then went through mean subtraction, with any negative values becoming 0. This was done to highlight brighter regions, which were ideally the lights on the stop sign if they were illuminated in the image.

G. Circle Detection

The preprocessed image was then passed through OpenCV's built-in HoughCircles [13] method with the maximum accepted radius one-fourth of the image size and the minumum accepted radius one-tenth of the image size. These values were decided because if the image perfectly fit the stop sign, then these should be the approximate size of the lights on the stop sign, controlling for variations of different sign models or how they might be illuminated. The minimum distance between two circle centers was 30 pixels (roughly half the image size). The other input values for the method were decided based on tweaking the results for one image or using the recommended values.

H. Result Pruning

The circles found in the circle detection were then pruned using positional information. The circles had to either be entirely located in the top or bottom half of the image and be in the center half of the image (left-most point had to be greater than one-fourth of the image width, and the right-most point had to be less than three-fourths of the image width). Only if these conditions were met were the circles considered to have been a positive detection for the lights.

I. Output

When testing my methods, the images ran through the process were sorted into three arrays: no stop sign, stop sign, and stop sign and light (with the detected light(s) in the image circled using OpenCV's draw circle method [14]).

III. Experimental Results

Fig. 4. A sample of some of the correct positive stop sign detections when 1449 image windows were passed through the program.

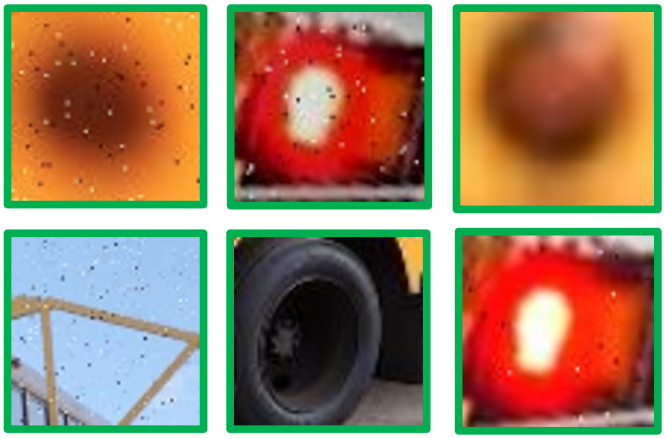

Fig. 5. Six of the nine false positive stop sign detections when 1449 images were passed through the program.

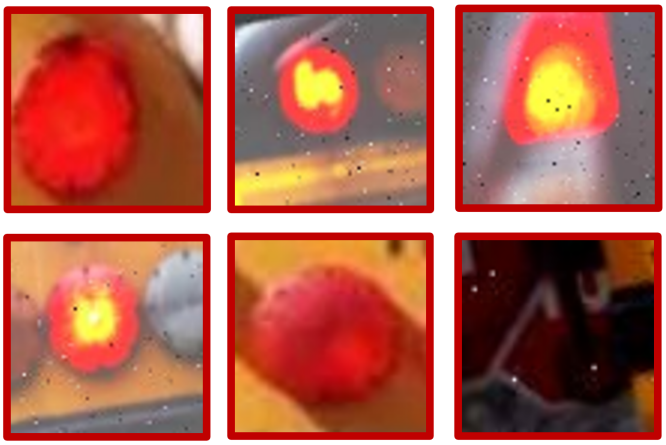

Fig. 6. A sample of some of the correct negative stop sign detections when 1449 image windows were passed through the program.

Fig. 7. All false negative stop sign detections when 1449 images were passed through the program.

Fig. 8. A sample of some of the images were illuminated lights were detected in images that were positive for containing a stop sign.

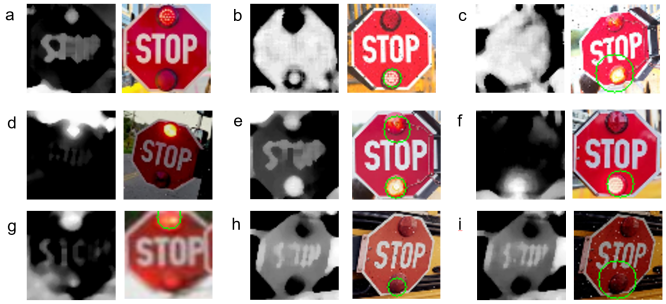

Fig. 9. A sample of the outputs of light detection with the preprocessed images used for the Hough Transform.

TABLE I
DETECTION ACCURACY

	Stop Sign Detected	Light Detected
Expected	231	134
Actual	234	39
Accuracy	98.96%	29%

IV. DISCUSSION

A. Interpretations

The model was very successful at detecting school bus stop signs. When the 1449 windows in the dataset were ran through the model, there were only 15 false detections (6 false negatives and 9 false positives), leading to a 98.96% accuracy rate. The reasons for these incorrect detections are varying. In Fig. 5, it can be seen that most of the false positives have a circular shape, particularly in relation to the red channel, and due to the scaled size, they ended up pixelated enough that the shape (HOG features) might have read more octagonal. If you compare this to Fig. 6, which contained correct negatives, most of the images of similar objects (the lights on the school bus), were correctly labeled, at least those of higher resolution. The false positives were lower resolution at the processed scale, which might have impacted how the program interpreted their features. However, there were outliers with less obvious reasons, like the bottom right image in Fig. 5, where the angular shape might have resulted in the positive detection, especially because the model was trained on a few mostly occluded stop signs. The tire also has less clear reasons as to what led to the false positive. However, whatever the reason, the extracted features managed to look very similar to those with positive detections, leading the model to mark it as a stop sign. There were only six false negatives, all of which can be seen in Fig. 7. There are common themes as to why these images were incorrectly detected. Most of them were occluded in some way, either by other objects or by shadows/intensity changes in the image. Others had higher amounts of noise or had very low image resolution, impacting the features extracted from the image. Overall, though, the model accurately detected school bus stop signs in a majority of the cases.

In contrast, the light detection was very inaccurate. Only 29% of expected lights were detected, with a smaller portion of these being accurate detections (actually lit up). One reason for this is that some of the logic I used for my preprocessing was incorrect. I applied mean subtraction to the window before applying the HoughCircles method, however I forgot to make sure that the pixels within the circle were brighter than the rest of the image, because in cases where the lights were darker than the mean, they were black in the preprocessed image, and it was this black circle that was found in multiple cases of the light detection (as seen in Fig. 9(h)). I tried using Otsu thresholding to fix this, however that often captured area outside of the illuminated circle, due to the fact that while the light was brighter than the rest of the image, it also illuminated the area around the light, resulting in non-circular outputs. Part of this was also due to how the light was often right on the edge of the stop sign, so these processes would easily be affected by the intensities in the background of the image (as seen in Fig. 9(d)). This also brings up the issue that the area behind the stop sign would affect the mean intensity of the image, so if the background was dark, then the mean was lower and the resulting image still had most of the stop sign as non-zero (seen in Fig. 9(h)). There were also instances where the light was not circular enough, sometimes because it was only a reflection that was being captured, other times because of differences in how the illumination was captured in the image.

B. Improvements

The current form of the application only works for still images, but it would be great if it could work for video or at least a collection of frames. For instance, the light on a school bus stop sign is not continuously on, it is typically flashing in an alternating pattern with the other one on the stop sign. Therefore, the light could better be detected by analyzing a collection of frames and seeing if the highest intensity areas switch locations over time. This still might be affected by areas of shadow and lighting changes, however it could potentially remove the need for the detection of the circular shape, which would be especially beneficial as I found instances where the lights on the stop sign were actually square shaped.

C. Advantages

The main advantage of this program is that the stop sign detection works with a high accuracy level. The light detection is more subjective to a variety of factors, but as long as a car can detect and stop for a school bus stop sign, the objective of this project has been reached.

D. Disadvantages

In the long run, there seemed to be more disadvantages than advantages for this implementation, for instance, the stop sign detection did not work efficiently on whole images, only pre-processed windows. Efficiency is important in the field of autonomous vehicles because they need to be able to make split second decisions for the safety of the passengers as

well as surrounding vehicles and pedestrians. Additionally, the light detection does not work accurately which limits the applications of the detection. In most cases, simply detecting the stop sign is enough, but like mentioned previously, a school bus stop sign might be in clear view at an intersection (seen in Fig. 1), but that doesn't mean that a vehicle should stop for it, which is why the light detection is necessary.

V. Conclusion

The implementation of this project did the core part of what it set out to do. It effectively detected stop signs on school buses, however failed to consistently detect the lights on the stop sign, due to incorrect logic as well as general issues with the methods chosen.

VI. References

References

[1] The Dawn Project, "The Dawn Project and Tesla Takedown Conduct Live Demonstration of Tesla FSD's Critical Safety Defects in Austin - The Dawn Project," The Dawn Project, Jun. 13, 2025. https://dawnproject.com/the-dawn-project-and-tesla-takedowns-live-safety-tests-of-tesla-full-self-driving-in-austin/

[2] "Waymo will recall software after its self-driving cars passed stopped school buses," NPR, Dec. 06, 2025. https://www.npr.org/2025/12/06/nx-s1-5635614/waymo-school-buses-recall

[3] A. Arunmozhi, S. Gotadki, J. Park, and U. Gosavi, "Stop Sign and Stop Line Detection and Distance Calculation for Autonomous Vehicle Control," 2018 IEEE International Conference on Electro/Information Technology (EIT), pp. 0356–0361, May 2018, doi: https://doi.org/10.1109/eit.2018.8500268.

[4] T. P. Cao and G. Deng, "Real-Time Vision-Based Stop Sign Detection System on FPGA," Digital Image Computing: Techniques and Applications, pp. 465–471, Jan. 2008, doi: https://doi.org/10.1109/dicta.2008.37.

[5] X.-H. Wu, R. Hu, and Y.-Q. Bao, "Parallelism Optimized Architecture on FPGA for Real-Time Traffic Light Detection," IEEE Access, vol. 7, pp. 178167–178176, 2019, doi: https://doi.org/10.1109/access.2019.2959084.

[6] H.-Y. Lin, S.-Y. Lin, and K.-C. Tu, "Traffic Light Detection and Recognition using a Two-Stage Framework from Individual Signal Bulb Identification," IEEE Access, vol. 12, pp. 132279–132289, Jan. 2024, doi: https://doi.org/10.1109/access.2024.3446277.

[7] ChatGPT. (GPT-5). OpenAI. Accessed: November. 25, 2025. [Online]. Available: https://chatgpt.com/share/692615ef-8958-8010-adfd-24ddd028c3e9

[8] "OpenCV: Geometric Image Transformations," docs.opencv.org. https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d

[9] "skimage.feature — skimage 0.25.2 documentation," Scikit-image.org, 2020. https://scikit-image.org/docs/0.25.x/api/skimage.feature.html#skimage.feature.hog

[10] "skimage.feature — skimage 0.25.2 documentation," Scikit-image.org, 2020. https://scikit-image.org/docs/0.25.x/api/skimage.feature.html#skimage.feature.local_binary_pattern

[11] "sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.22.1 documentation," scikit-learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[12] "OpenCV: Smoothing Images," docs.opencv.org. https://docs.opencv.org/4.x/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html

[13] "OpenCV: Feature Detection," docs.opencv.org. https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d

[14] "OpenCV: Drawing Functions in OpenCV," docs.opencv.org. https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html

[15] Scenelabpro, "Yellow School Bus Stopped At Intersection In Roslyn Long Island On A Sunny Day Video." Accessed: Dec. 15, 2025. [Online]. Available: https://www.pond5.com/stock-footage/item/112853369-yellow-school-bus-stopped-intersection-roslyn-long-island-su