ENPM673 Project 6: Traffic Sign Recognition

Hae Lee Kim

Zhengliang Liu

Table of Contents

Objective

For this project, a sequence of images taken from a car's dash is provided. Detection, recognition and fine tuning of the traffic signs in the images is the goal of this project. Out of all the traffic signs seen in the images, only 8 of the signs are to be recognized.

Traffic Sign Detection

Color Segmentation

For the detection aspect of traffic sign recognition, color segmentation and thresholding were done to detect regions of blue and red (since all signs have blue or red features). The sequence of images was first converted to the HSV color space then a blue and red values were experimented with to obtain the ranges for the thresholds. A mask is then applied to the image frames to only display regions that are within the color thresholds. The threshold values are as follows:

1. Blue (HSV lower/HSV upper bounds): [6,75,20], [20,255,255]
2. Red (HSV lower/HSV upper bounds): [112,50,60], [200,150,255]

An example of a frame with the blue mask is shown in Fig. 1. Two parking signs show up in the frame since they are blue in RGB color space.

Fig. 1: Blue mask frame with two parking signs.

Region of Interest

After applying the blue and red color masks, the region of interest needs to be cropped out in order for the SVM model (discussed later) to receive an input image that only includes the traffic sign. Two methods were investigated in order to crop the desired region.

First, blob detection, cv2.SimpleBlobDetectorcreate(), method was used to where the area size, circularity, and convexity were adjusted to recognize the traffic sign region. The blob detection was able to detect most signs; however, the detection was not consistent between frames and would only detect the sign for only a couple frames even though there exists many consecutive frames with the same traffic sign. It also was not able to detect more than one sign in a frame at a time.

Instead, contour feature detector functions were used which gave much better results. cv2.approxPolyDP() was used to find all contour shapes in the frame, the area was found using cv2.contourArea(contours). An area threshold was then applied to be able to identify only the traffic signs. Then cv2.boundingRect(contours) was used to create a bounding box around the region and a region slightly larger was cropped to use as input into the recognition model. Fig. 2 shows some sample cropped images from the detection. After cropping the sign, all cropped images were resized to be 64 x 64.

Fig. 2: Traffic sign detection cropped images

Traffic Sign Classification

Now that the detection is complete, there are two parts for the classification – Histogram of Oriented Gradients (HOG) to find feature descriptors and Support Vector Machines (SVM) as the classification algorithm.

HOG Feature Descriptor

A feature descriptor is an image representation or a patch of an image that simplifies an image by extracting useful information from the image and ignoring the rest [1]. HOG is a feature descriptor that uses histograms of the directions of gradients as the features. The OpenCV function cv2.HOGDescriptor() is used. Function arguments include winSize, blockSize, blockStride, cellSize, and nbins.

1. winSize = (64,64) – size of image
2. cellSize = (8,8) – size of the sub-image that captures the feature
3. blockSize = (16,16) – typically 2x cellSize
4. blockStride = (8,8) – overlap between neighboring blocks and controls the degree of contrast normalization
5. nbins = 9 – # of histogram bins

Support Vector Machines (SVM) Model

SVM is a discriminative classifier defined by a separating hyperplane. Given labeled training data which is supervised learning, the algorithm outputs an optimal hyperplane to categorize the test data [2].

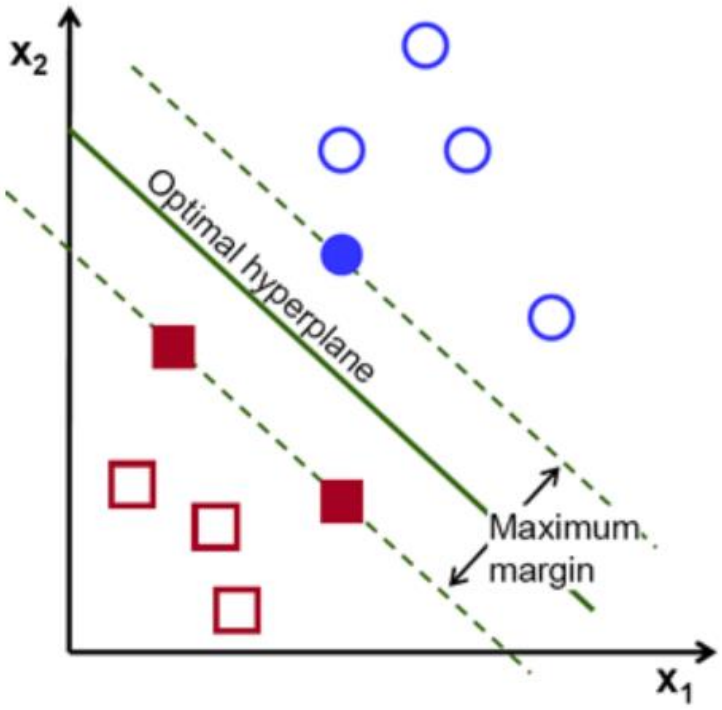

Fig. 3: Optimal hyperplane that maximizes margin given set of data [2].

A series of cropped images of various traffic signs have been provided to use as training and testing data for the classification model. The training set which consisted of 61 folders of images. The label for each folder of images can be identified from the folder number. For each image, the HOG descriptor was computed, then the descriptor was appended to a list for the training data. Its label was appended to a separate list. Then, OpenCV functions for creating an SVM model were used. The following functions/steps are used:

1. Create SVM: svm = cv2.ml.SVM_create()
2. Set SVM type: svm.setType(cv2.ml.SVM_C_SVC)
3. Set SVM kernel to linear: svm.setKernel(cv2.ml.SVM_LINEAR)
4. Set C parameter (does not really matter since kernel is linear): svm.setC(2)
5. Set gamm parameter (also does not matter): svm.setGamma(1)
6. Train model with training data: svm.train(train_data, cv2.ml.ROW_SAMPLE,label_train)
7. Save SVM model: svm.save("svm_model")

For every crop the detection part of the algorithm makes, the HOG descriptors will be extracted, then the function svm.predict(test_data) was used to classify the traffic sign. After

classifying the sign, a sample image of the traffic sign is placed next to the input video's traffic sign.

Fine Tuning

To achieve the best possible results, many parameters had to be tuned from both detection and classification steps. Most of the process was trial and error to see which thresholds would give the best detection and classification combination. The following are some parameters that were tuned and other steps that were taken to reduce false positives and negatives.

- Blue lower HSV bound
- Blue upper HSV bound
- Red lower HSV bound
- Red upper HSV bound
- Different color spaces
- Blob detection parameters
- Contours features parameters
- Image crop area
- Min area threshold
- Max area threshold
- Additional training dataset
- Bounding box width/height ratio
- SVM kernel type

Results

Samples of the 8 signs detected are shown in Fig. 4-11.

Fig. 4: Parking sign

Fig. 5: Stop sign

Fig. 6: Lane Merge Sign

Fig. 7: Direction and bicycle signs

Fig. 8: Narrow lane sign

Fig. 9: Yield sign

Fig. 10: Speed bump

References

[1] https://www.learnopencv.com/histogram-of-oriented-gradients/

[2] https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html