**CS 482: Final Project Paper/Tutorial**

Cody Querubin, Misha Burnayev, Alexander Walker

Github: [CodyQue/CS482-Traffic-Sign-Classifier (github.com)](CodyQue/CS482-Traffic-Sign-Classifier (github.com))

Pre-Recording: [https://youtu.be/cR-u7iA2Gtg](https://youtu.be/cR-u7iA2Gtg)


# Traffic Sign Classifier Using Viola-Jones Algorithm

***NOTE***: Most of this paper explains the logistics behind how the program/algorithm functions. To learn how to run the program, skip to the ***Tutorial/How To Run The Program*** section.

## Problem/Motivation

Implementing machine vision in a vehicular setting has been rapidly growing and increasing, especially with technologies such as self-driving cars and traffic cameras. Although there are many training models for machine visions, such as YOLOS, SSDs, CNNs, R-CNNs, and many more, this project's objective is to explore a different, and potentially even faster, method for classifying data in computer visions and compare the results to these training models. We wanted to pick an algorithm that is uncommonly used to detect traffic signs; therefore, the main algorithm used in this project to identify traffic signs is the Viola-Jones algorithm. This algorithm is mainly used for face detection; however, this project will alter the training to be able to classify traffic signs, instead of faces.

Aside from using the Viola-Jones algorithm, other techniques such as feature selection and cosine similarity were also used to help detect and train images to detect traffic signs. The library, OpenCV, was used for selecting features in images and selecting shapes. The library, sklearn, was also used for classifying test images.

# The Viola-Jones Algorithm

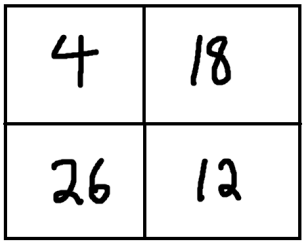
## Algorithm:

Proposed in 2001 by Paul Viola and Michael Jones, the Viola-Jones algorithm is a machine-learning algorithm used for object detection in images and videos. The most common practice of using this algorithm is for face detection. This section explains how the Viola-Jone algorithm works, and how this algorithm is altered to detect and classify traffic signs. The procedure is as follows: it first computes the Integral Image; then using the Integral Image, it computes the Haar Features of the image, and the classification of signs depends on the outcome of these Haar Features.

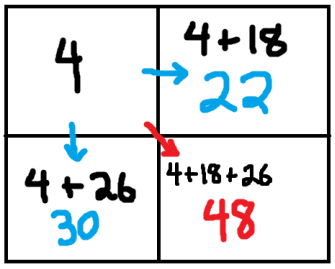
## The Integral Image

The algorithm first converts the image into a grayscale image. This is to make sure each pixel value is between 0 and 255; it would be very complicated if the pixel values were in RGB values. The algorithm's second step is the integral image, the sum of all pixels at a specific image row and column. Below is an example of how the integral image is computed using a 4x4 image:

**Original Image (In Grayscale Value):**




**Integral Image (In Grayscale Value)**

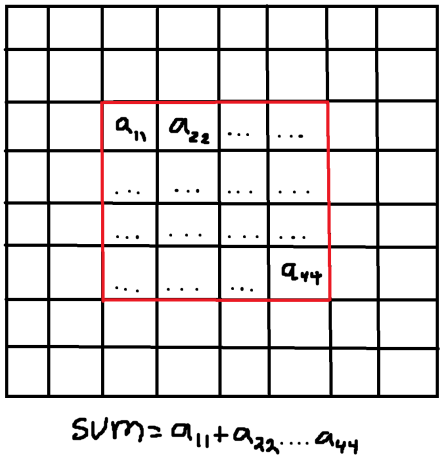



Once the entire integral image is computed, the algorithm loops through every possible feature in the image. It first finds the total number of pixels of, and around, the feature. An example below demonstrates how the total number of pixels is determined.

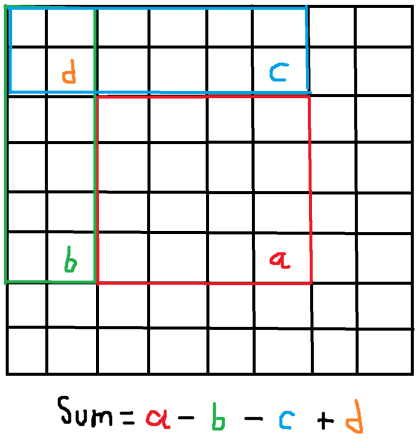
**Efficiency:** The reason this machine-learning algorithm works well and fast is because of the utilization of the Integral Image, which saves a lot of time when computing the total number of pixels of a specific area. Rather than taking the pixel sum by looping through each pixel of a specific area, which takes O(n^2) time complexity, it determines the sum of all pixels of a specific feature by 1) Getting the sum of all pixels of a specific point, 2) Subtracts the total number of sums of the bottom left, 3) Subtracts the

total number of sums of the upper right, 4) Adds the total number of pixels of the upper left. This alone is a time complexity of O(1).

**Total Number Of Pixels By Adding All Pixels In Feature (Number Of Operation: N)**



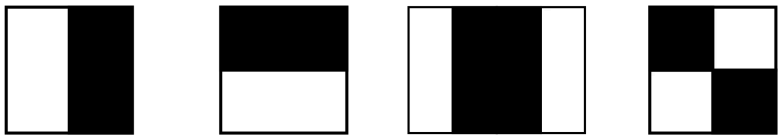

**Total Number Of Pixels Using Integral Image (Number Of Operation: 4)**




## Haar Features

Haar Features are essential to the Viola-Jones algorithm since these are features to determine object recognition. These features are measured by taking the difference between two different areas of a selected feature in the image. However, what makes Haar Features reliable are the different types of features and the different ways of computing these Haar Features.

Although there are many different Haar Features, this project utilizes the four main different Haar Features: the x-axis edge feature, the y-axis edge feature, the line feature, and the four-rectangle feature. These are what the different Haar Features look like, respectively.



(1) (2) (3) (4)

**Formula:** The formula is as follows: it is the difference between the pixel intensities of the white area and the pixel intensities of the black area. By doing this, it obtains a value, for each feature, and this is used for classification. The image below is the mathematical equation to calculate these Haar Features.

$$\sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i,j)1_{P(i,j) \text{ is white}} - \sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i,j)1_{P(i,j) \text{ is black}}.$$

## Implementation

The entire Viola-Jones algorithm, calculating the integral image and computing the Haar Features, used for this project was developed in the violajones.py. The algorithm was all developed, with the help of a few sources to understand how the algorithm worked.

## Existing Work Relating To Viola-Jones Algorithm

Below is some existing research studying the Viola-Jones algorithm to be used to detect objects in images. Most of the research provided uses Viola-Jones for face detection.

**An Analysis of the Viola-Jones Face Detection Algorithm- Yi-Qing Wang**
article.pdf (ipol.im)

**Facial Parts Detection Using Viola-Jones Algorithm**
Facial parts detection using Viola Jones algorithm | IEEE Conference Publication | IEEE Xplore

**Facial Emotion Recognition Based On Viola-Jones Algorithm in the Learning Environment**
Facial Emotion Recognition Based on Viola-Jones Algorithm in the Learning Environment | IEEE Conference Publication | IEEE Xplore

**Literature Study of Face Recognition Using The Viola-Jones Algorithm**
Literature Study of Face Recognition using The Viola-Jones Algorithm | IEEE Conference Publication | IEEE Xplore

**Spliced Images Detection By Using Viola-Jones Algorithms Method**
Spliced images detection by using Viola-Jones algorithms method - ScienceDirect

# Utilizing OpenCV For Feature Selection And Shape Detection

Traditionally, the Haar Features are determined by looping through the entire image to measure pixel intensity patterns. However, for this project, the program uses the OpenCV library to extract major features from the image.

Additionally, shapes were also detected using the OpenCV library. This is done by finding the Contours and looping every Coutours to find ‘big shapes’. These big shapes are assumed to be traffic signs. This is important.

Both feature and shape selection are essential for this project. Combining these two algorithms would enable the selection of features inside of signs. It is essential to make sure it selects actual sign features instead of selecting a random feature, like a feature from a tree or a house since these extra features could contribute to the wrong classification.

# Training/Preprocessing Data

The training is accomplished through three separate programs, testSigns.py, testRedSigns.py, and testYellowSigns.py. This is used to train signs based on sign color; this will be explained in the ***Classifying Signs In Images*** section as to why this is relevant. Although they train different signs, the algorithm to train is the same. The program, *training.py* is used to call the *trainSigns.py, trainRedSigns.py, and trainYellowSigns.py* and train and classify signs.

The first thing done is to open the .txt file (signs.txt, redsigns.txt, and yellowsigns.txt depending on the color classification). The .txt file(s) contain the name of the file name and their classification number. The training program loops through every row of the .txt file and opens the image file.

It then detects the ‘shape’ of the sign and extracts features inside it using the OpenCV library. Once it gathers all of the features inside of the sign, it computes the Viola-Jones algorithm on every feature in the sign, which returns values of each Haar Feature. These Haar Features are stored in their respective .csv file (the CSV files with the name train____.csv); this will be opened when classifying test images. The classifications are also stored in their respective CSV files (files with the name trainClassifier____.csv).

**Data Storage:** A separate directory, named *input*, is used to store the training set of all of the signs. When running the training program, it selects the sign images inside of the directory. To **add to the training set**, add the image to this directory, and add the sign to the .txt file, depending on what color the sign is.

**Exceptions:** Some signs do not have a training set since they are uniquely shaped. For example, stop signs are the only octagon-shaped signs, and pedestrian/school crossing signs are the only pentagon-shaped signs. Therefore, there are no training sets for these signs, and instead automatically classify octagons as stop signs and pentagons as pedestrian/school crossing signs.

# Classifying Signs In Images

The user inputs images, and the program will classify the main signs inside the image. Just like how the data is trained, the program detects the ‘’shape’ of the sign and extracts features inside of the sign; then it computes the Viola-Jones algorithm on every feature in the sign.

**Color Classification:** Once each Haar Feature is calculated in the image, it moves on to classifying the sign. It first determines the color of the sign; this is where the sign color, from the ***Training Data,*** comes in. If the test sign is red, then it only considers red signs when classifying. The same logic applies to yellow and black/white signs. This is to make sure it does not loop through every single sign feature and instead considers signs of the same color.

**Sign Classification Using Cosine Similarity:** Classifying signs in images utilizes the Cosine Similarity algorithm, from the sklearn library, to find the closest and most similar train feature. Once it finds the closest feature, the feature where the cosine similarity value is greater than 0.95, it assigns that sign classification to the test feature. After performing cosine similarity and classifying all of the test features, it picks the classification with the greatest count, and that sign is classified.

# Experimental Results

## Outcome

Here are the results when classifying test images using the Viola-Jones algorithm. The program does a really good job detecting shapes, detecting sign features inside of the shapes, and classifying signs based on the shape, color, and Haar Features.

One surprising thing was the detection of speed limits and detecting them. Because there are many different speed limit variants (25 MPH, 35 MPH, 45 MPH, etc), and because the signs only contain words, it was feared that the program would not be able to classify these signs

## Problems And Future Improvements

Although the program works well to classify signs in images, there are still problems/issues encountered when testing the program.

### Amount Of Signs That Exist

One small issue is the large number of signs that exist today. The program classifies the few, common, important signs of the United States, such as stop signs, yield signs, no right turn, no turn or red, windy roads, speed limits, and one-way. However, there are still a lot more signs that do exist.

Aside from the signs in the United States, there are also signs to consider from other countries around the world, and these signs differ from signs in the United States depending on sign color, pixel intensity patterns, and many more.

### Blurry/Low-Quality Images Containing Traffic Signs

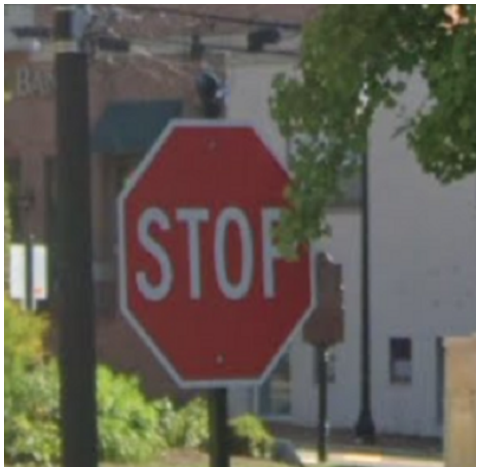
There is one major issue when running the program. Although the program works with images clearly showing the sign, it does not work for blurry/distorted images containing signs. However, it is not the program developed that is causing this issue. The reason is because of OpenCV's shape detection; this tool makes it difficult to detect shapes on low-quality images.

Below is a low-quality stop sign image. Since it is low-quality, the OpenCV's shape detection cannot identify the big octagon in the middle, so therefore, it results in finding the next 'big shape', which is the middle of the sign.


There are possible solutions to fix this; the easy solution is the use blob detection, instead of shape detection and select the 'big blobs' of the image, assuming these big blobs contain the signs. One solution, if wanting to continue with shape detection, is using interpolation to approximate signs in images. The issue using this is the implementation complexity; it would need to predict the color of the sign. This would be complicated to implement but would work well to make objects look clearer.

### Objects Blocking Signs

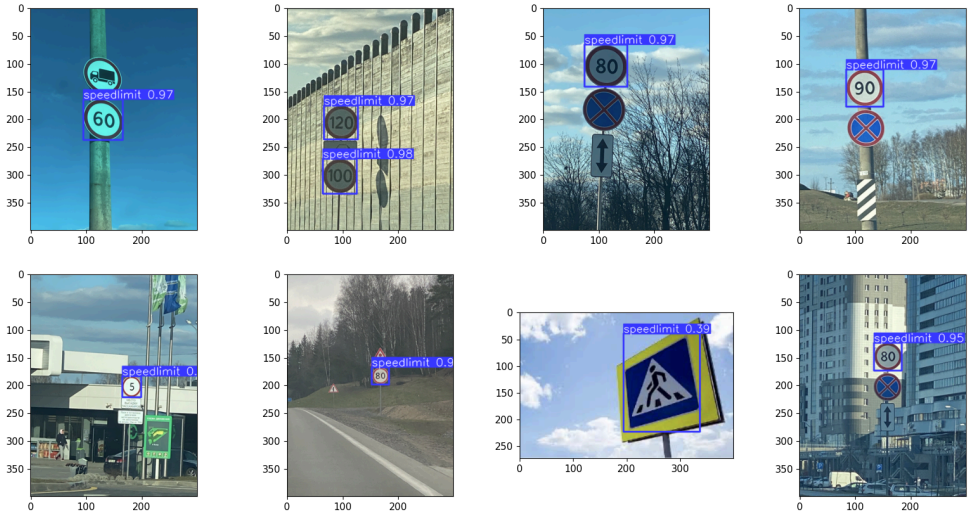
Another objective the OpenCV's shape detection does not work well on is detecting shapes with objects in the way of it. The bottom image of the stop sign demonstrates this. Because the tree branch is blocking the stop sign, the OpenCV's shape detection cannot detect the shape of the sign being an octagon.

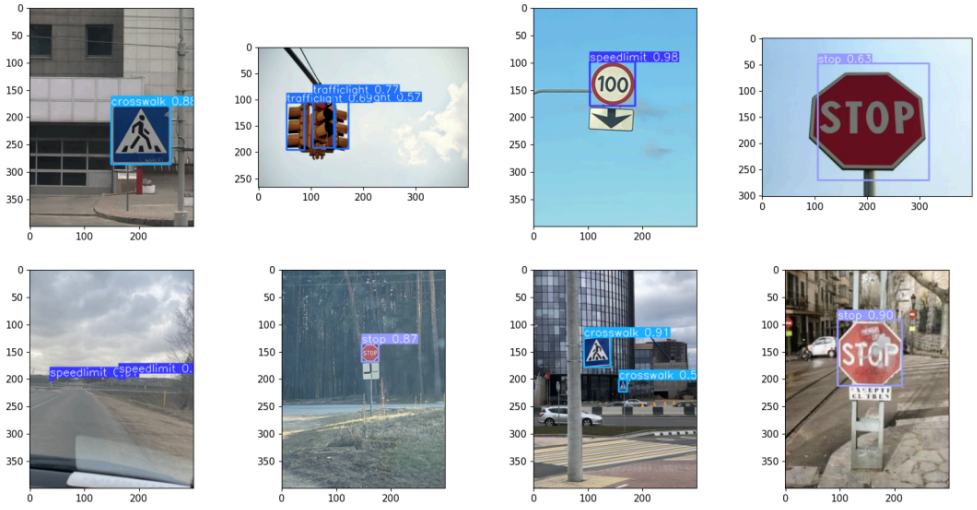



## Comparing Viola-Jones With Other Machine Learning Models

There are obviously imperfections with this program dealing with low-quality images and signs with objects in front of it, making the classification difficult. However, comparing this algorithm with other models, the algorithm does a impressive job with training data and classifying signs that are clearly shown in the image.

**YOLO:** Aside from using the Viola-Jones algorithm, the YOLO model was also used to classify signs in images. The YOLO fixes the issues getting from our Viola-Jones algorithm by being able to detect shapes regardless of image quality. It is able to detect signs up-close or really far. It can also classify different variants of the same sign (for example, the different variants of speed limit signs). It can also classify signs regardless of objects in the way too. For the comparison, the YOLO v5 Model was used to train and classify traffic signs in images. Below shows the traffic sign classification using the YOLO model:






Although this fixes the issues contained in our Viola-Jones algorithm, there is still a downfall for using this algorithm. The preprocessing and model training takes a very long time to compute. The YOLO model took approximately 3 ½ hours to train, compared to the training using our Viola-Jones algorithm which took less than a minute to compute. However, because of the long training time, this is probably why the YOLO model can classify any sign, regardless of how close or far they are, what angle they are located, and what objects are blocking the signs.

# Tutorial/How To Run The Program

Before running the program, these dependencies need to be installed:

- Cv2
- Pandas
- NumPy
- sklearn

**Running Train Data:** The directory already contains CSV files containing the Haar Features and the classification file. However, if wanting to add signs to the training set, add the image to the *signs* directory and edit the .txt file, depending on the color of the sign. Run the *training.py* program to run all of the training programs (trainSigns.py, trainRedSigns.py, and trainYellowSigns.py).

**Running Classifying Program:** The first thing to do is put images in the *input* directory; the program loops through every image in the directory. Run the *main.py* program to start classifying signs in the image. Most of the computations, such as color and sign classification and shape detection, are done through the *main.py* program. Additionally, other programs, such as the *violajones.py* and *featureselector.py* programs are also called upon from *main.py*.

# Citations


Computerphile. (2018, October 19). *Detecting faces (Viola Jones algorithm) - computerphile*. YouTube. https://www.youtube.com/watch?v=uEJ71VlUmMQ&ab_channel=Computerphile

Hirzi, M. Fakhrul, Syahril Efendi, and Rahmat Widia Sembiring. "Literature study of face recognition using the viola-jones algorithm." *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*. IEEE, 2021.

Jeyasrisenthil. “Road Sign Detection: YOLOV5.” Kaggle, Kaggle, 8 July 2023, www.kaggle.com/code/jeyasrisenthil/road-sign-detection-yolov5/notebook.

Kirana, Kartika Candra, Slamet Wibawanto, and Heru Wahyu Herwanto. "Facial emotion recognition based on viola-jones algorithm in the learning environment." *2018 International seminar on application for technology of information and communication*. IEEE, 2018.

OpenAI. (2024). *ChatGPT* (3.5) [Large language model]. https://chat.openai.com

Prasanna, GV Sai, K. Pavani, and Mahesh Kumar Singh. "Spliced images detection by using Viola-Jones algorithms method." *Materials Today: Proceedings* 51 (2022): 924-927.

Shakrina, Y. (2020, April 22). *The viola-jones algorithm*. YouTube. https://www.youtube.com/watch?v=p9vq90NYHMs&ab_channel=YoussefShakrina

Vikram, K., and S. Padmavathi. "Facial parts detection using Viola Jones algorithm." *2017 4th international conference on advanced computing and communication systems (ICACCS)*. IEEE, 2017.

Wang, Yi-Qing. "An analysis of the Viola-Jones face detection algorithm." *Image Processing On Line* 4 (2014): 128-148.

Wikimedia Foundation. (2023, November 3). *Haar-like feature*. Wikipedia. https://en.wikipedia.org/wiki/Haar-like_feature

YouTube. (2021, March 3). *Haar features for face detection | face detection*. YouTube. https://www.youtube.com/watch?v=ZSqg-fZJ9tQ&ab_channel=FirstPrinciplesofComputerVision