**ECE 276A – Stop Sign Detection Project**

A little info about your project and/ or overview that explains **what** the project is about.

This project is intended to take as an input an image, determine if there are any stop signs in the image, and return the location all stop signs, in a Python environment. To accomplish this, the program uses logistic regression to train a classifier to identify the red in stop signs, then various OpenCV functions to determine if red regions fit the characteristics of a stop sign.

**Motivation**

This project exists to satisfy the requirement of UC San Diego class ECE 276A, Project 1. The purpose of this project and the class as a whole is for the student to learn and get familiar with applying the classification algorithms of Logistic Regression, Single Gaussian Generative Modeling, and Gaussian Mixture Generative Modeling.

**Build status**

This project has been completed and tested, though it passed only 11 out of 15 test cases in the ECE 276A Autograder. The reasons behind this, as well as details about which test cases the program handles well and which it does not can be found in the project report.

**Code style**

The goal of the student was to code this project in the Python Pep8 style, according to python.org, however, as this is project is the first time the student has coded in Python, there are some styling errors in the code that the student will look to resolve in later versions.

**Tech/framework used**

The Red/notRed classification part of this project was achieved by through logistic regression, as detailed in ECE 276A lecture on Supervised Learning.

The OpenCV library was used to do all image processing (for example: importing an image, overlaying the bounding boxes on stop signs, etc.), as well as polygon fitting on a classified image (in order to filter out stop signs from other red objects in the image).

**Features**

This project features the ability to start from a set of training images and re-train the color classifier if the user so desires (for example, having more data points, or wishing to use their own data points). Therefore, if the user desires, this project can go all the way from the user’s own inputted images as training data to finding the location of stop signs in new images.

If the user prefers to simply find the stop signs in a large set of images, without retraining or altering the program that is also possible, with the user simply needing to select the appropriate directory when importing the images.

**Code Example**

This is an example of how the project can be used, with the following images showing the correct implementation for calling “get\_bounding\_box” function and printing out the results of the bounding box for the image shown. Clearly it can be seen that these coordinates represent, in cartesian coordinates, the two opposing corners of the bounding box around the stop sign.

A screenshot of a cell phone

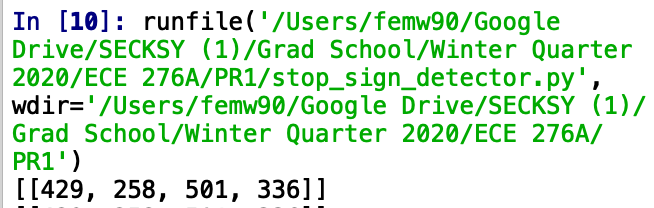
Description automatically generated

The code representing the correct way to call the “get\_bounding\_box” function and handle the results it returns.

A red stop sign sitting on the side of a road

Description automatically generated

The image given to the code above, placed in the “TestFolder” directory seen in the second line

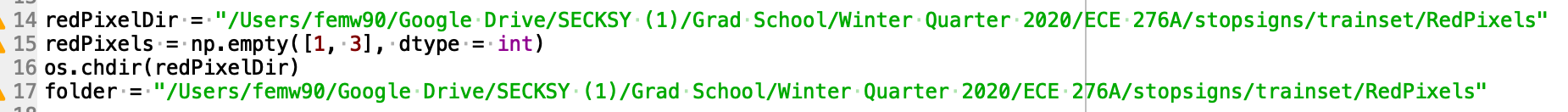


The output of the code above. Clearly it can be seen that the corners of the required bounding box are given as the last line of the output.

**Installation**

To install, please make sure the user is running Python 3.7.0 or later and has OpenCV cv2 installed.

If the user does not wish to re-train the model, the user can skip this step. Place the training images (if desired) in the same directory as the code included for this project, in folders named ‘﻿RedPixels’ for training images consisting solely of red pixels, and ‘﻿notRedPixels’ for training images with no red pixels or all red pixels blacked out. Adjust the directory in lines 14, 17, 31, and 34 of “ECE 276A – Find Red and NotRed.py” to reflect the current directory of the training images (seen in the image below).



Finally, the user must run “ECE 276A – Find Red and NotRed.py” and “ECE 276A – PR1 – Parameter Finder.py” in that order to obtain the new parameters. Finally, set the new parameters obtained to the wOptimized variable in line 40 of “stop\_sign\_detector.py” (seen below).



**Pseudo Code**

**Code File Name: ECE\_276A\_Find\_Red\_and\_NotRed.py**

Import necessary libraries

redPixelDir = Directory containing images with red training pixels

redPixels = empty array to store red pixels in

set directory

folder = Directory containing images with red training pixels

for all files in the folder

if the file is .jpg

read the image

remove pixels without sufficient red (all non-red pixels are black)

add these to the list of pixels

keep only the first row of these

get the length of the red pixels

create an array for the red labels of same length

make all labels 1

notRedPixelDir = Directory containing images with not red training pixels

notRedPixels = empty array to store not red pixels in

set directory

folder = Directory containing images with not red training pixels

for all files in the folder

if the file is .jpg

read the image

remove pixels that are red

add these to the list of pixels

keep only the first row of these

get the length of the not red pixels

create an array for the not red labels of same length

make all labels -1

make TrainingData containing all the pixels

make TrainingDataLabels containing the corresponding labels for each pixel

save TrainingData

save TrainingDataLabels

**Code File Name: ECE\_276A\_Parameter\_Finder.py**

Import necessary libraries

Define sigmoid

If z is less than -20 (to prevent overflow)

Return 0

Else

Return value of sigmoid

Main function

Load training data

Load training data labels

Create new omega array

Create current omega array

Set the value of alpha

Set the number of iterations

Create an array to store the iterations (to see if gradient descent is progressing)

For the number of iterations

Set result of sum equal 0

For iterating through all the training pixels

Define j (ends up being trivial)

Set y value equal to the class value of current pixel

Set x equal to current pixel

Find x transpose

Calculate z

Calculate value in the summation for the current pixel

Add the result to the total sum

Set the new omega equal to the current plus alpha times summation result

Store the iteration value

**Code File Name: stop\_sign\_detector.py**

Import necessary libraries as given in project spec

Create class stopSignDetector

Define initialization function

Define segment\_image

Get the size values for the image passed in

Set image height

Set image width

Create numpy array for reshaping image

Reshape the image

Define the optimized omega values

Create a vector to store the mask in

Iterate through each pixel

Set xT value to current pixel

Calculate the test value

If test value is greater than 0

Set current pixel on the vector mask to 1

Reshape vector mask into array

Transpose to get correct orientation

Return the mask

Define get\_bounding\_box

Set mask\_img equal to output of segment\_image

Blur the mask to elimate jagged edges

Threshold the blur to black and white

Convert blur to array of type int

Turn blurred\_img into black and white rgb

Get the contours of the mask

Elimate all but the largest 3 contours

Get the size of the original image

Set height

Set width

Create array to store bounding boxes for stop signs in

Iterate through all contours

Calculate epsilon value for approxPolyDP

Get a polygon around current contour

Get polygon length

If length is between 6 and 14 segments

Fit an ellipse

Get the axes of the elipse

Find major axis

Find minor axis

Calculate ratio of major to minor axis

If ratio is close enough to a square

Get a bounding rectangle for the polygon

If the rectangle is large enough

Get the corners of the rectangle

Add the corners to the lost of boxes

Sort the list as required in the spec

Return the list of lists of box corners for the image

Define sigmoid

If z is less than -20 (to prevent overflow)

Return 0

Else

Return value of sigmoid

Main function

Define the folder the test image is in

Create detector object

Iterate through all files in the folder

Get the current image

Define boxes to store bounding boxes

Get bounding boxes with get\_bounding\_boxes

Print bounding boxes

**Credits**

ECE 276A Course Website: <https://natanaso.github.io/ece276a/>

ECE 276A Piazza: [https://piazza.com/class/k4tbsww5fuu4y0?cid=34#](https://piazza.com/class/k4tbsww5fuu4y0?cid=34)

Study Group: Roumen Guha, Maria Fatima