Computer Vision based Traffic Sign Recognition

Motivation:

The motivation for the algorithm is to develop a **TRAFFIC SIGN RECOGNITION SYSTEM** that is compatible with **RASPBERRY PI.**

IMPLEMENTATION:

The Implementation of the algorithm is divided into 2 parts.

1. Image pre-processing
2. Image Recognition

TOOLS USED:

The following tools are for applying the algorithm in Python

1. Python
2. Visual Studio Code (Editor for many types of software like Python, C, Java etc.,)
3. List of libraries in python (Installation commands)
    a. OS (default library installed while installing python)
        i. This library is used to access files and folders available in the system (windows)
    b. OpenCV (pip install opencv)
        i. This library is used to apply image pre-processing methods on images
    c. Pandas (pip install pandas)
        i. This library is used for reading files like excel, csv etc.,
    d. Numpy (pip install numpy)
        i. This library is used to perform matrix operations easily on data
    e. Sklearn (pip install sklearn)
        i. This library is used for implementing Machine Learning algorithms

PRE-PROCESSING:

There are several steps in this process

1. Applying Histogram Equalization on the image
2. Converting Image to Gray Scale

Histogram-Equalization:

This is a image processing technique that helps to equalize the pixel Intensity throughout the image. This method is helpful to recognize images in low lighting conditions.

Gray Scale:

Grayscaling is a method where the image is converted from Color to Gray. This reduces the Image size

In a Color Image there are 3 different matrices.

1. Red intensity matrix
2. Green intensity matrix
3. Blue intensity matrix

But a gray image has only 1 matrix.

Hence we can decrease large amount of data by converting image to greyscale. This decrease in size helps algorithm to run faster.

RECOGNITION:

In this phase the pre-processed images are sent into a ML algorithm for recognition. The ML methods used for recognition are:

1. K-Nearest Neighbour(KNN)
2. Support Vector Machine Classifier(svc)
3. Decision Tree Classifier(dtree)
4. Random Forest Classifier(rf)
5. Neural Network (nn)

METHOD FOLLOWED:

To implement the algorithm we have used the GERMAN TRAFFIC SIGN DATASET (https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign). The Dataset was divided into train and test folders. A total of 43 classes of different Traffic signs are taken for training. Along with images 2 different csv files are given for Train and test data in which the region of interest of each image is specified.

We have imported all the training images into a array using the OS Library specified above. The Region of Interest specified in the excel file is used to extract the traffic sign from the image. This extracted image is then resized to 30x30. This resizing helps in training the model fast. The resized image is pre-processed as mentioned in the PRE-PROCESSING.

The pre-processed training images are then sent into a Machine learning model in order to train the model. Here we have used various models mentioned in RECOGNNITION. The **ACCURACY** of each model is as follows:

Model	Accuracy(%)
K-Nearest Neighbour	43
Support Vector Machine (linear kernel)	88.45
Naïve Bayes	40.88
Decision Tree	68.53
Random Forest	82.75

Improvements:

1. The models were considered with an assumption “The model should not affect the Raspberry Pi”. Hence we can still go into Deep Learning for models with much better accuracy.
2. To implement a method to find region of Interest of real-time images.