St. Thomas' College of Engineering & Technology

Traffic Light Detection

Prepared by

Suman Kalyan Das **University Reg. No.- 031558 (roll-12200219003)**

Varsha Kumari **University Reg. No.- 181220110123 (roll-12200218003)**

Under the guidance of

Aditi Bal
Assistant Professor, Information Technology

Project Report

Submitted in the partial fulfillment of the requirement for the degree of B.Tech in Information Technology

Department of Information Technology

Affiliated to

Maulana Abul Kalam Azad University of Technology, West Bengal

June, 2022

St. Thomas' College of Engineering and Technology

This is to certify that the work in preparing the project entitled Traffic Light Detection has been carried out by Varsha Kumari, Suman Kalyan Das undermy guidance during the session 2021-2022 and accepted in partial fulfillment of the requirement for the degree of Bachelor of Technology in Information Technology.

Dr. Arindam Chakravorty

Head, Department of Information Technology

Aditi Bal

Department of Information Technology

Acknowledgement

We are submitting the project Traffic Light Detection under the guidance of Aditi Bal who supported us at every stage of the report and guided us with everything with right and excellent knowledge. We would also like to thank our project review class teachers, they give us the instruction what to add in our project or what improvementsneed to be done.

Signature with date

SUMAN KALYAN DAS

Signature with date

VARSHA KUMARI

Vision & Mission (St. Thomas' College of Engineering & Technology)

Vision

To evolve itself into an industry oriented research based recognized hub of creative solutions in various fields of engineering by establishing progressive teaching-learning process with an ultimate objective of meeting technological challenges faced by the nation and the society.

Mission

- To create opportunities for students and faculty members in acquiring professional knowledge and developing social attitudes with ethical and moral values.
- To enhance the quality of engineering education through accessible, comprehensive, industry and research oriented teaching-learning process.
- To satisfy the ever-changing needs of the nation for evolution and absorption of sustainable and environment friendly technologies.

Vision & Mission (Department of Information Technology)

Vision

To promote the advancement of learning in Information Technology through research oriented dissemination of knowledge which will lead to innovative applications of information in Industry and Society.

Mission

- To incubate students grow into industry ready professionals, proficient research scholarsand enterprising entrepreneurs.
- To create a learner- centric environment that motivates the students in adopting emerging
  technologies of the rapidly changing information society.
- To promote social, environmental and technological responsiveness among the membersof the faculty and students.

Program Educational Objectives (PEO)

Graduates of Information Technology Program shall

PEO1: Exhibit the skills and knowledge required to design, develop and implement IT solutions for real life problems.

PEO2: Excel in professional career, higher education and research.

PEO3: Demonstrate professionalism, entrepreneurship, ethical behavior, communication skills and collaborative team work to adapt the emerging trends by engaging in lifelong learning.

PROGRAM OUTCOMES(POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Mapping with Program Outcomes

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
2	2	3	3	3	2	2	3	3	1	2	2

Enter correlation levels 1, 2 or 3 as defined below:

1: Slight (Low) **2: Moderate (Medium)** **3: Substantial (High)**

Justification :

In this project, we can apply our engineering knowledge, system components, used modern IT tools, including classification and prediction. By this project, we understand the professional engineering solutions in social and environmental contexts. Demonstrate knowledge and understanding of the engineering community and with society at large, such as being able to comprehend and design documentation make effective presentations and give clear instructions. Recognize the need for and have the preparation and ability to engage independent and life-long learning in the broadcast context of technological change.

Program Specific Outcomes (PSOs)

At the end of the program the students will be able to

PSO1 (Programming) : Apply the programming knowledge to build an efficient and effective solution of the problem with an error free , well documented and reusable code, user friendly interface and well organized database.

PSO2 (Multimedia Authoring) : Create a multimedia product using proper metaphor, designing effective navigation following human computer interface rules with proper interactivity, which will be useful for educational , social and business purpose.

PSO3 (Software Engineering) : Understand and analyze a big complex problem and decompose it into relatively smaller and independent modules either algorithmically or in an object oriented way choosing correct life cycle model and using effective test cases.

Project Mapping with Program Specific Outcomes

PSO1	PSO2	PS03
2	1	3

Enter correlation levels 1, 2 or 3 as defined below:

1: Slight (Low) **2: Moderate (Medium)** **3: Substantial (High)**

Justification:

Understand and analyze **Traffic Light Detection** problem and classify the image data using machine learning and deep learning.

Index

Introduction

Traffic light detection and recognition is essential for autonomous driving in urban environments. A camera based algorithm for real-time robust traffic light detection and recognition was proposed, and especially designed for autonomous vehicles.

Traffic light detection is a vital feature for an autonomous car and so is the proximity sensing for obstacles around it. As one transitions from manually driven cars to fully autonomous ones, recognition of traffic lights becomes a key part of Driver Assistance Systems (DAS). Research related to autonomous cars is the flavor of the season in the development of intelligent transportation systems. Autonomous cars aspire to fully replace the human driver by a computer system, with no compromise and eventually improving safety and efficiency. The human ability of sight (for example – seeing the roads, pedestrians, road signs and other vehicles) and associated behavior need to be carefully reproduced by the computer system. Autonomous cars must have the capability to perceive traffic lights and recognize their current states (red, amber, green). When some form of computer-controlled automation is involved in autonomous cars, safety and reliability are critical. Since ma-chines aren't 100% accurate, it is good to have redundancies in case the machine fails. The worst-case scenario would be when the autonomous car is made to determine that a red light is imminent when it is not the case. This distracts the driver, or leads to the driver applying emergency brakes. Current research focuses on day-time detection and recognition, which makes it much easier to reject false positives, for example - tail lamps, street lights, red-colored road signs and other reflections.

A camera mounted on an autonomous car will be able to capture the images of the traffic light in real-time. Which will be fed to the computer as input datasets. The Opencv2 module is being used primarily for color detection, Hough gradient method for reducing the loss function and Hough circles to draw a circle around the traffic light for recognition. The autonomous car will either come to halt if a red light is detected or continue driving - if a green light is detected and if an amber light is detected it will start its engine.

1.1 Problem Statement:

Traffic Light Detection for Autonomous vehicles

1.2 Problem Definition

In this fast-moving digital world autonomous cars will be a norm in the near future. One of the most critical features of any autonomous car is traffic light detection and recognition The problem related to outdoor perception is a key issue faced by driver-assisted and autonomous cars, as the machine, unless taught, cannot detect traffic lights, road signs, obstacles, etc. in the direction of motion. Autonomous cars need to be capable of detecting traffic signals and recognizing their current status whereas Human beings can easily identify the relevant traffic signals.

1.3 Objective

The Objective is to compare and contrast the feasibility and overall performances of the different approaches and thus figure out the best suited algorithm among the ones tested for satisfactorily precise
Traffic light Detection.

1.4 Tools and Platform

1.4.1 Python: [1]

The Python programming language is a general-purpose programming language that has carved its own notch into the list a most used programming languages. This language has many key takeaways that make it stand out for its uses and applications in the corporate world, as well as, many other areas of industry. This complex, yet simple language's history is fascinating. So, even though Python is still used all over the world in almost every industry you can imagine, it was conceived in the late 1980s. It was then implemented in 1989 by Guido van Rossum That is right, like a good amount of other programming languages,

Python has been around and has weather the test of time well. In fact, Python's popularity only continues to rise. This upward trend shows no signs in stopping, leaving Python as a force to be reckoned with. Guido van Rossum, often referred to as, Benevolent Dictator for Life (BDFL), is the principal author of the Python programming language. Guido van Rossum, has recently, however, stepped down as leader on July 12, 2018. Python itself is actually named after Monty Python's Flying Circus. The first version of the code published by Van Rossum was labeled version 0.9.0. Even in this early version of code there were classes with inheritance, exception handling, functions, and many other major datatypes. Python's userbase form comp.lang.python in 1994 was formed, leading to more growth and use of the language. In 1994, Guido was invited to the USA by NIST, the US National Institute for Standards and Technology.

NIST was interested in using Python for several standards-related projects and needed somebody to boost their Python skills. Obviously, the creator of language was a great choice.With NIST support, Guido was able to run workshops and participate in conferences, spreading Python and attracting key contributors that would later be very important for the future of the language. Starting from 2000, core developers started thinking about Python 3.0. They wanted to streamline the language, cutting unnecessary language constructs and functions that Python had accrued in its almost 20 years of existence.

1.4.2 Visual Studio Code: [2]

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

Its popularity is due to the growth of the web development field in these years and the need of the developers of having a lightweight well-done editor, with few features but less complex than the others available on the market.

Here is a list of 10 features every developer should know about Visual Studio Code:

1. Command Line-Visual Studio Code has a powerful command line interface that lets you control how you launch the editor. You can open different files, install extensions, and even change the display language at the startup.
2. Command Palette-VS Code is equally accessible from the keyboard. The most important key combination to know is Ctrl+Shift+P, which brings up the Command Palette. From here, you would have access to all of the functionality of VS Code, including keyboard shortcuts for the most common operations.
3. Git Integration-Visual Studio Code comes with Git integration that allows you to commit, pull, and push your code changes to a remote Git repository.
4. Change language mode-If you want to persist the new language mode for that file type, you can use the Configure File Association for command to associate the current file extension with an installed language.
   Customization-
5. Zen Mode-Zen Mode lets you focus on your code by hiding all UI except the editor (no Activity Bar, Status Bar, Side Bar and Panel), going to full screen and centering the editor layout. Zen mode can be toggled using the View menu, Command Palette, or by the shortcut Ctrl+K Z. Double Esc exits Zen Mode.

6. Split view-If you're good at multitasking and if you are working on two different files of the same project simultaneously, or need to check the difference between two files then go to the split view.
7. Status Bar-Users can see Errors and warnings from here.Keyboard Shortcut: Ctrl+Shift+M
8. Debugging-Open the Command Palette (Ctrl+Shift+P) and select Debug: Open launch.json, which will prompt you to select the environment that matches your project (Node.JS, Python, C++, etc). This will generate a launch.json file. Node.JS support is built-in and other environments require installing the appropriate language extensions. See the debugging documentation for more details.
9. Default Keyboard shortcuts-All of the commands are in the Command Palette with the associated key binding (if it exists). If you forget a keyboard shortcut, you can use the Command Palette to help yourself out. Download the keyboard shortcut reference sheet for your platform (macOS, Windows, and Linux).

1.4.3 Anaconda Navigator [3] - Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.Navigator is an easy, point-andclick way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- Jupyter
- Jupyter Notebook
- Spyder
- Pycharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anacondo Prompt(Windows only)
- Anaconda PowerShell(Windows only)
- Datalore
- IGB Watson Studio Cloud
- QtConsole
- PyCharm Professional

1.4.4 Cascade Trainer GUI [4] -

Cascade Trainer GUI is a program that can be used to train, test and improve cascade classifier models. It uses a graphical interface to set the parameters and make it easy to use OpenCV tools for training and testing classifiers.

When Cascade Trainer GUI is first started you will be presented with the following screen. This is the starting screen and it can be used for training classifiers. To train classifiers usually you need to provide the utility with thousands of positive and negative image samples, but there are cases when you can achieve the same with less samples.

To start the training, you need to create a folder for your classifier. Create two folders inside it. One should be "p" (for positive images) and the other should be "n" (for negative images).

For example, you should have a folder named "Car" which has the mentioned folders inside it.

Positive image samples are the images of the object you want to train your classifier and detect. For example, if you want to train and detect cars then you need to have many car images.

And you should also have many negative images. Negative images are anything but cars.

Important Note 1: Negative images must NEVER include any positive images. Not even partially.

Important Note 2: In theory, negative images can be any image that is not the positive image but in practice, negative images should be relevant to the positive images. For example, using sky images as negative images is a poor choice for training a good car classifier, even though it doesn't have a bad effect on the overall accuracy:

1.4.5 Libraries and packages

1.4.5.1 NumPy: [5]

NumPy is a Python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.Arrays are very frequently used in data science, where speed and resources are very important. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.This behavior is called locality of reference in computer science.This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.
Functions used in Numpy are:
1. numpy.array-It is used to create an array.
Parameters used in numpy.array are:
i. Object:Array-Like:

An array, any object exposing the array interface, an object whose __array__ method returns an array, or any (nested) sequence. If object is a scalar, a 0-dimensional array containing object is returned.

1.4.5.2 Pandas: [6]

Pandas is a Python library used for working with data sets.It has functions for analyzing, cleaning, exploring, and manipulating data.The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.Relevant data is very important in data science. Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Functions used in Pandas are:

1.Pandas.DataFrame()-Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns. Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects.

The parameters used in pandas.DataFrame are:
i. Data: ndarray (structured or homogeneous), Iterable, dict, or DataFrame- Dict can contain Series, arrays, constants, dataclass or list-like objects. If data is a dict, column order follows insertion-order. If a dict contains Series which have an index defined, it is aligned by its index.

ii. Columns : Index or array-like:
Column labels to use for resulting frame when data does not have them, defaulting to RangeIndex(0, 1, 2, …, n). If data contains column labels, will perform column selection instead.

1.4.5.3 OpenCV(cv2): [7]

cv2 is the module import name for opencv-python. OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

Some OpenCV library functions are:
1.cv2.imshow()-cv2.imshow() method is used to display an image in a window. The window automatically fits to the image size.
2.cv2.imread() method-It loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

1.5 Brief Discussion on Problem

Traffic light detection technology can help drivers to identify the status of traffic lights and make decisions quickly according to the identified status of traffic lights. This can reduce driver distraction and prevent the occurrence of non-standard driving and illegal behavior. Therefore, this research on a traffic light detection and recognition model with a high accuracy in real-time has practical significance and
broad development prospects for improving road traffic safety. Traffic light detection systems often use industrial cameras to collect road condition information. Due to the complexity and changeability of the traffic light image background in real traffic scenarios, the traffic light in an image occupies less pixels and its feature structure is sparse, which increases the difficulty of algorithm recognition. Therefore, it is very important to study a more effective small target detection algorithm for traffic signal light detection.

Traffic light detection has undergone several years of development, and many excellent detection methods have been reported. At present, these methods can mainly be divided into two categories : one is based on traditional algorithms using image processing and machine learning, while the other is based on deep learning.

Traffic light detection is a vital feature for an autonomous car.Autonomous cars aspire to fully replace the human driver by a computer system, with no compromise and eventually improving safety and efficiency.The human ability of sight (for example – seeing the roads, pedestrians, road signs and other vehicles) and associated behavior need to be carefully reproduced by the computer system.

Autonomous cars must have the capability to perceive traffic lights and recognize their current states (red, amber, green). When some form of computer-controlled automation is involved in autonomous cars, safety and reliability are critical. Since ma-chines aren‟t 100% accurate, it is good to have redundancies in case the machine fails. The worst-case scenario would be when the autonomous car is made to determine that a red light is imminent when it is not the case. This distracts the driver, or leads to the driver applying emergency brakes. Current research focuses on day-time detection and recognition, which makes it much easier to reject false positives, for example - tail lamps, street lights, red-coloured road signs and other reflections.
A camera mounted on an autonomous car will be able to capture the images of the traffic light in real-time. which will be fed to the computer as input datasets. The Opencv2 module is being used primarily for color detection, Hough gradient method for reducing the loss function and Hough circles to draw a circle around the traffic light for recognition. The autonomous car will either come to halt if a red light is detected or continue driving - if a green light is detected and if an amber light is detected it will start its engine.

1. The camera mounted on the windshield inside the autonomous car will capture the images of traffic lights which will be the datasets (input) to the open cv2 module.

2. The Python code uses an open cv2 module for color detection and the RGB (Red-Green-Blue) colors in the datasets will be converted into the equivalent HSV (Hue-Saturation-Value).
3. Color range values are specified using an array for the respective colors.
4. Masking of HSV is done to set the threshold values so that the traffic light detection becomes more accurate (The computer will not detect each and every red or green or amber object as traffic light).
5. Hough gradient method is used to reduce the loss function by drawing a line from the central coordinates of the circle to the frame of the image (the image is broken down into small frames by the code when detecting the traffic light), then a ratio of center of circle to the number of times the iteration has gone outside the bounds of the image (count) is taken; it is compared to the optimal epoch (the number of iterations) value, if the condition is satisfied the loss is significantly reduced.
   $(x - xcenter)^2 + (y - ycenter)^2 = r^2$

   Where, xcenter and ycenter are the central coordinates of the Hough circles, x and y are any point on the circumference of the circle and r is the radius of the circle.
6. Hough circles are used to draw circles around the detected color (output).
7. Uint16 data type is used to convert the corresponding color range values to pixels (dimensions).
8. Once the traffic light is detected and recognised a text is displayed on the output image as follows:
   - Red-Stop
   - Amber-Get ready
   - Green-Go

2. Concepts and Problem Analysis

In order to reduce accident at traffic intersections, the algorithm of traffic lights detection which is applied in a vehicle driver assistance system is designed by using the image processing technology. The system of traffic light detection includes three parts: a CCD camera, an image acquisition card, and a PC. Based on RGB color space, the algorithm extracts red, green, and yellow objects in the image firstly; For the purpose of eliminating disturbance in the environment, the features of traffic lights are used to verify the object identity, and then the types of traffic signals are judged.. The results of experiments show that the algorithm is stable and reliable.

How will the driverless car detect the traffic lights?

Well, as humans, we have an incredible ability of our eyeballs and our ability to see. We can look at a scene and find the gorilla that's hidden over there behind the stack of boxes. Similarly, we can look at the street scene up ahead and "know" where the traffic signal is.
The sensors on the AI self-driving car have to put in a bit more work.
The cameras capture images of what's ahead of the self-driving car. The image needs to be analyzed by the system. The image might contain not only a traffic signal, but perhaps there's a plane flying through the sky that can be seen behind the traffic signal, maybe there's a few birds resting on the traffic signal, maybe there's rain coming down and the traffic signal is partially obscured by the heavy rain. And so on.

It's not such an easy thing to find the traffic signal in a picture. Yes, the traffic signal is likely the same kind of shape nearly all of the time. It's usually on a post of some kind. It's got the three lights. It's hanging over the intersection. These are all valuable clues. I'm not saying it is rocket science per se to find the traffic signal, but it is more work than we think. So, the first step involves capturing images via the sensors and analysing those images to find the traffic signal. We want to find the traffic signal and also not be fooled by something that might resemble a traffic signal. There could be other nearby lights such as for a lit-up billboard or maybe lights on the exterior of a building. Those might be red lights, yellow lights, green lights, and so you cannot just look for a particular colour of a light.

Therefore, from a camera mounted on the windshield inside the car will capture the images in real time and to avoid any other vehicles back lights or headlights, the camera will be placed in such a way that it will only capture the upper part of the view.

What Happens During Detection?

Everything has to be done in real time as the car is still moving and detection of traffic lights and recognizing which colour it is happening simultaneously, this process needs to take place very fast.
After capturing the images of the front view, the picture will be processed and traffic signal will be detected. For detecting traffic signal, we are using Haar Cascade algorithm. The camera will continuously capture the images and when traffic light signal will appear, using this algorithm that traffic signal will be detected.
After detecting traffic signal, it will be cropped and will be saved in a folder as jpeg file and from there, the cropped image will be taken to detect the traffic light colour.
For detecting traffic light color, we are using opencv module of machine learning. Using this, traffic light color will be detected and recognised and a text will appear accordingly if it is red- stop will appear in text, if it is amber- Get Ready will appear in text and if it is green - Go will appear in text.

Edge cases:

Suppose the traffic signal is there but it is not working? Maybe the power is out, maybe it has had a failure, etc. I know you might think that it low odds that a traffic signal exists but is not working – it is though a possibility. It is a very real possibility that you might not even be able to see a traffic signal such as suppose you are driving behind a big truck as you come up to an intersection. Your view is blocked by the truck. The AI of the self-driving car has to be advanced enough to deal with these various scenarios. Keep in mind that this is all life-or-death stuff. A wrong move by the AI self-driving car can harm the human occupants of the AI self-driving car, and harm other humans by hitting other cars or maybe hitting a pedestrian, etc.

Another variant to deal with involves the visibility of seeing the traffic signal. Suppose the camera has dirt on the lenses and only gets a partial image. The lighting near the traffic signal can also impact detecting the status of the traffic signal. Have you ever driven up to an intersection and the sun was directly in your eyes? You could barely see the traffic signal lights.You knew that there was a traffic signal there, but it was nearly impossible to see if the light was red, yellow or green.

Machine Learning: [8]

Machine learning (**ML**) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks.[1] It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.[2] Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

Approaches:

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Supervised learning:

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.[35] The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.[36] An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.[19]

Types of supervised-learning algorithms include active learning, classification and regression.[27] Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Unsupervised learning:

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function.[37] Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some *similarity metric* and evaluated, for example, by *internal compactness*, or the similarity between members of the same cluster, and *separation*, the difference between clusters. Other methods are based on *estimated density* and *graph connectivity*.

Reinforcement learning:

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP).

Many reinforcement learning algorithms use dynamic programming techniques.[39] Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

3. Design and Methodology:

3.1 Dataset Overview:

Initially we have made our dataset using a 3rd party software named "Cascade Classifier gui". In this software we have to provide only positive and negative images separately to make dataset in XML format.

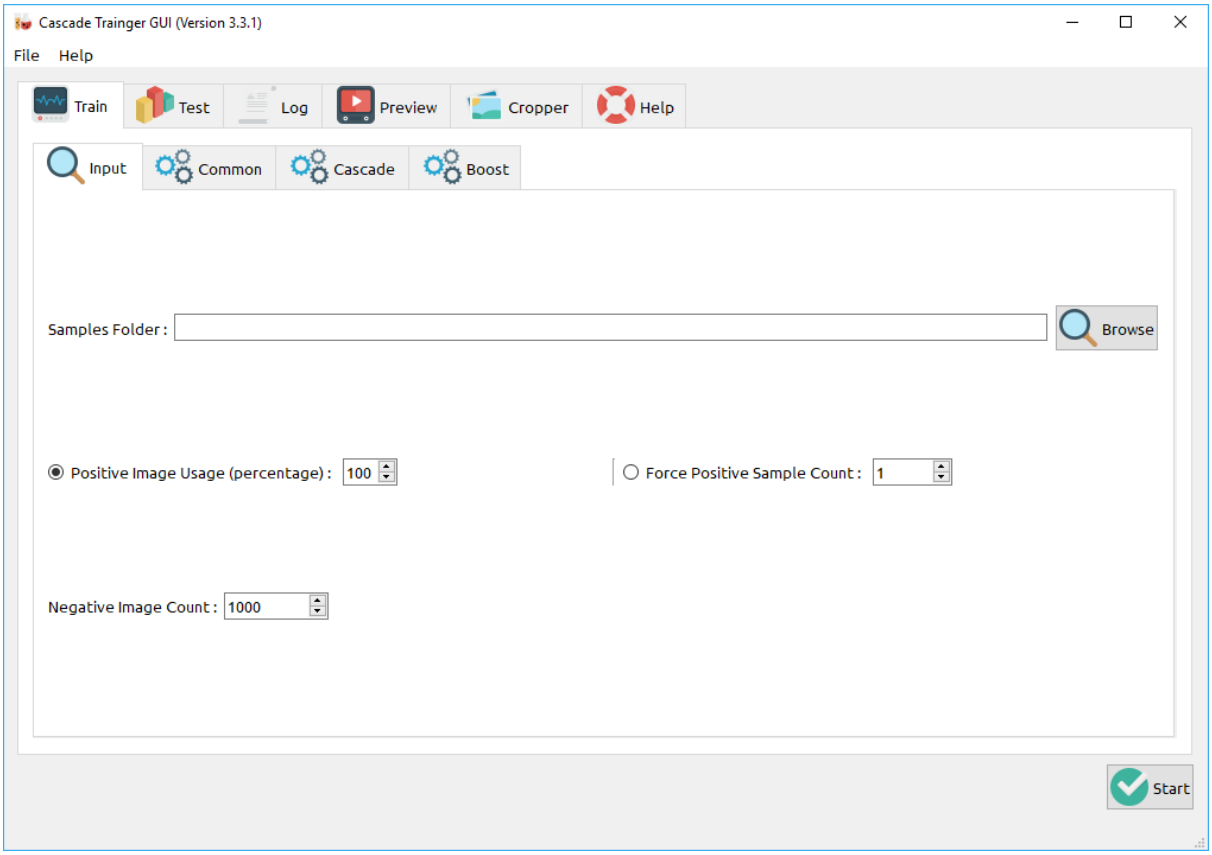
3.1.1 Data Pre-Processing:

We used Around 200+ images of "Traffic Light" which will be our target object to detect. These images will be used to train our model. We also used 100+ negative images to train our dataset. We downloaded all images from different website. Then we transformed all images into gray scale. We divided them into two folders, one is p (for positive images) and second one is n (for negative images).

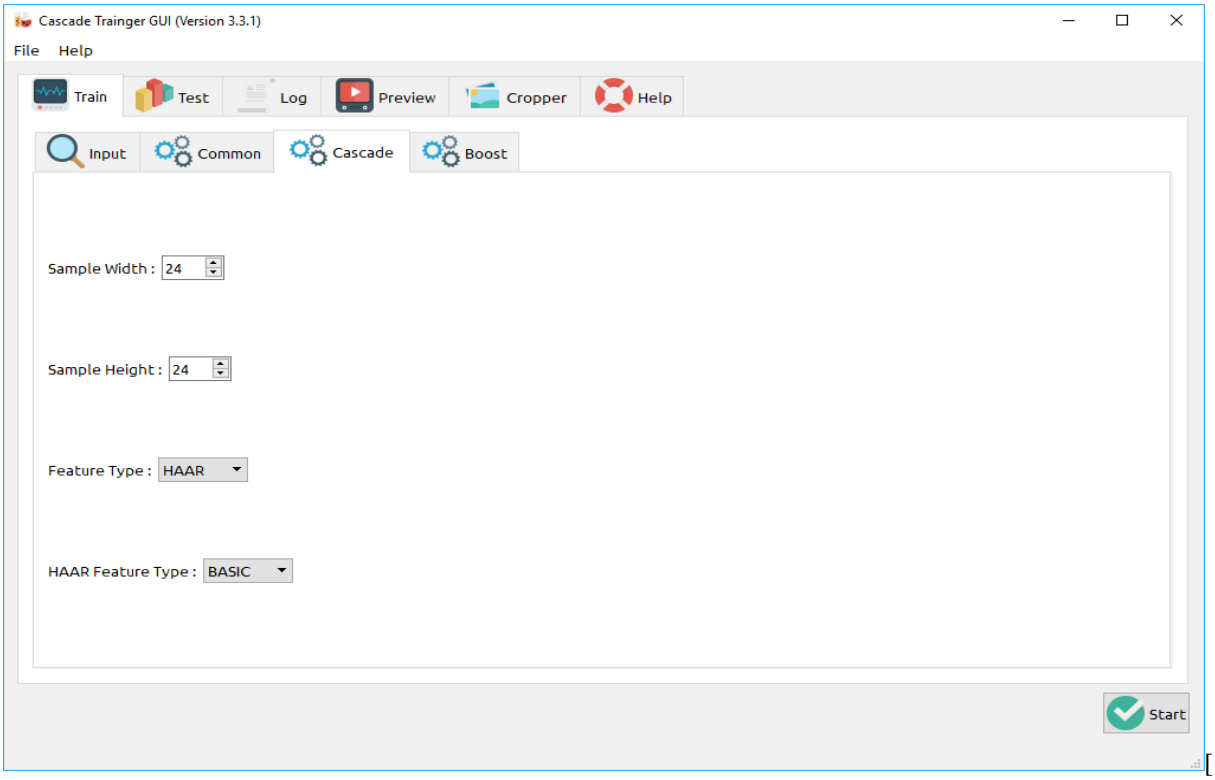
3.1.2 Final dataset:

We have make our dataset using cascade classifier gui. We provide p and n folder as input to the software. Then we select some options provided by the software. Like number of stages, pre calculated buffer size, pre calculated indices buffer size, number of threads and etc.

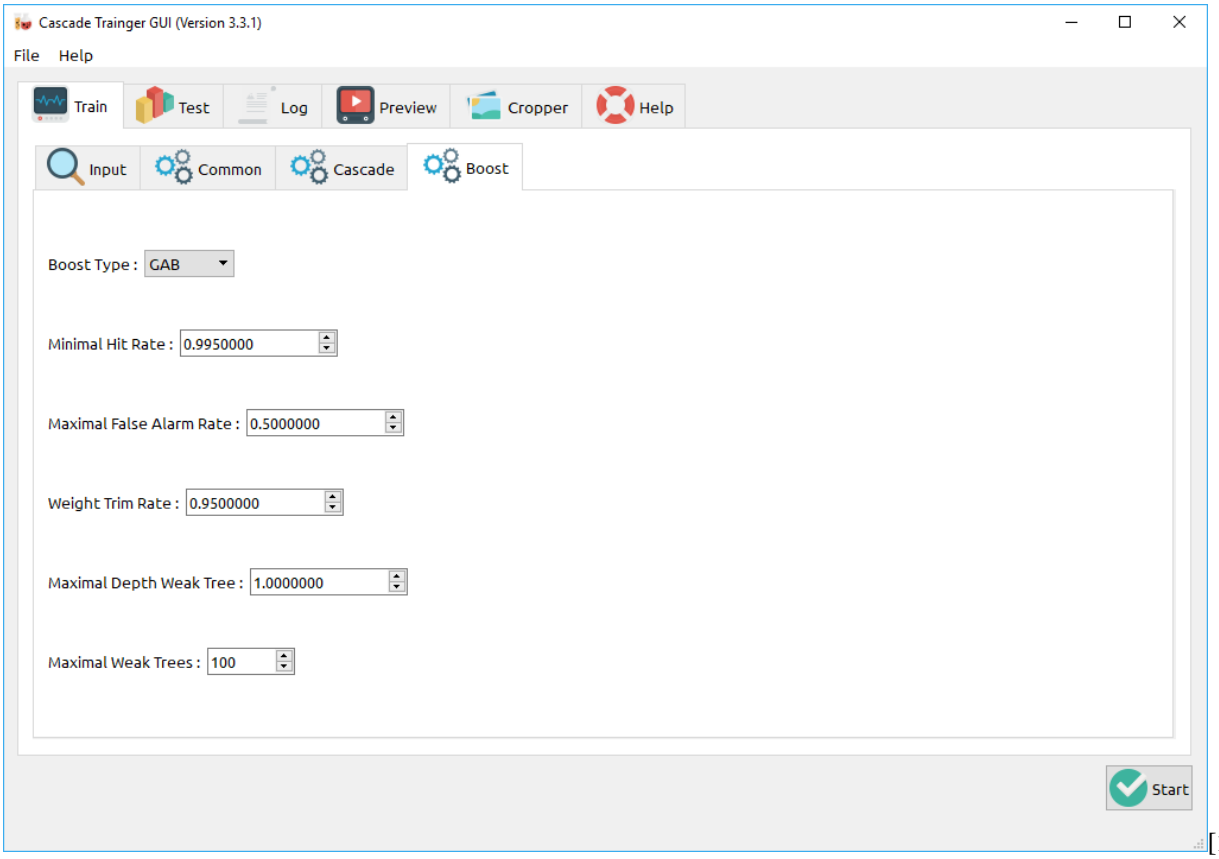
Some snipets of cascade classifier gui :



[9]



[10]



[11]

3.2 Training:

The next step is the actual training of the boosted cascade of weak classifiers, based on the positive and negative dataset that was prepared beforehand.

Command things of cascade classifier application grouped by purposes: [12]

- **Common options :**
  - -data <cascade_dir_name> : Where the trained classifier should be stored. This folder should be created manually beforehand.
  - -vec <vec_file_name> : vec-file with positive samples (created by opencv_createsamples utility).
  - -bg <background_file_name> : Background description file. This is the file containing the negative sample images.
  - -numPos < number_of_positive_samples> : Number of positive samples used in training for every classifier stage.
  - -numNeg <number_of_negative_samples> : Number of negative samples used in training for every classifier stage.
  - -numStages <number_of_stages> : Number of cascade stages to be trained.
  - -precalcValBufSize <precalculated_vals_buffer_size_in_Mb> : Size of buffer for precalculated feature values (in Mb). The more memory you assign the faster the training process, however keep in mind that -precalcValBufSize and -precalcIdxBufSize combined should not exceed you available system memory.
  - -precalcIdxBufSize <precalculated_idxs_buffer_size_in_Mb> : Size of buffer for precalculated feature indices (in Mb). The more memory you assign the faster the training process, however keep in mind that -precalcValBufSize and -precalcIdxBufSize combined should not exceed you available system memory.
  - -baseFormatSave : This argument is actual in case of Haar-like features. If it is specified, the cascade will be saved in the old format. This is only available for backwards compatibility reasons and to allow users stuck to the old deprecated interface, to at least train models using the newer interface.
  - -numThreads <max_number_of_threads> : Maximum number of threads to use during training. Notice that the actual number of used threads may be lower, depending on your machine and compilation options. By default, the maximum available threads are selected if you built OpenCV with TBB support, which is needed for this optimization.
  - -acceptanceRatioBreakValue <break_value> : This argument is used to determine how precise your model should keep learning and when to stop. A good guideline is to train not further than 10e-5, to ensure the model does not overtrain on your training data. By default this value is set to -1 to disable this feature.
- **Cascade parameters:**
  - -stageType <BOOST(default)> : Type of stages. Only boosted classifiers are supported as a stage type at the moment.
  - -featureType<{HAAR(default), LBP}> : Type of features: HAAR - Haar-like features, LBP - local binary patterns.
  - -w <sampleWidth> : Width of training samples (in pixels). Must have exactly the same value as used during training samples creation (opencv_createsamples utility).
  - -h <sampleHeight> : Height of training samples (in pixels). Must have exactly the same value as used during training samples creation (opencv_createsamples utility).
- **Boosted classifier parameters:**
  - -bt <{DAB, RAB, LB, GAB(default)}> : Type of boosted classifiers: DAB - Discrete AdaBoost, RAB - Real AdaBoost, LB - LogitBoost, GAB - Gentle AdaBoost.
  - -minHitRate <min_hit_rate> : Minimal desired hit rate for each stage of the classifier. Overall hit rate may be estimated as (min_hit_rate ^ number_of_stages),
  - -maxFalseAlarmRate <max_false_alarm_rate> : Maximal desired false alarm rate for each stage of the classifier. Overall false alarm rate may be estimated as (max_false_alarm_rate ^ number_of_stages)
  - -weightTrimRate <weight_trim_rate> : Specifies whether trimming should be used and its weight. A decent choice is 0.95.
  - -maxDepth <max_depth_of_weak_tree> : Maximal depth of a weak tree. A decent choice is 1, that is case of stumps.

- -maxWeakCount <max_weak_tree_count> : Maximal count of weak trees for every cascade stage. The boosted classifier (stage) will have so many weak trees (<=maxWeakCount), as needed to achieve the given -maxFalseAlarmRate.

- **Haar-like feature parameters:**
  - -mode <BASIC (default) | CORE | ALL> : Selects the type of Haar features set used in training. BASIC use only upright features, while ALL uses the full set of upright and 45 degree rotated feature set.

    After the opencv_traincascade application has finished its work, the trained cascade will be saved in cascade.xml file in the -data folder.

3.3 Object Detection Algorithm:

We have use **Viloa Jones Algorithm and Haar Cascade Classifier** for detecting traffic light object. And for color detection we use open-cv.

3.3.1 Viloa Jones Algorithm and Haar Cascade: [13]

Viola Jones is a novel approach to rapid detection of objects with running capabilities of 15 frames per second. It was the first to achieve real time object detection.

3.3.1.1 Viola Jones Detector

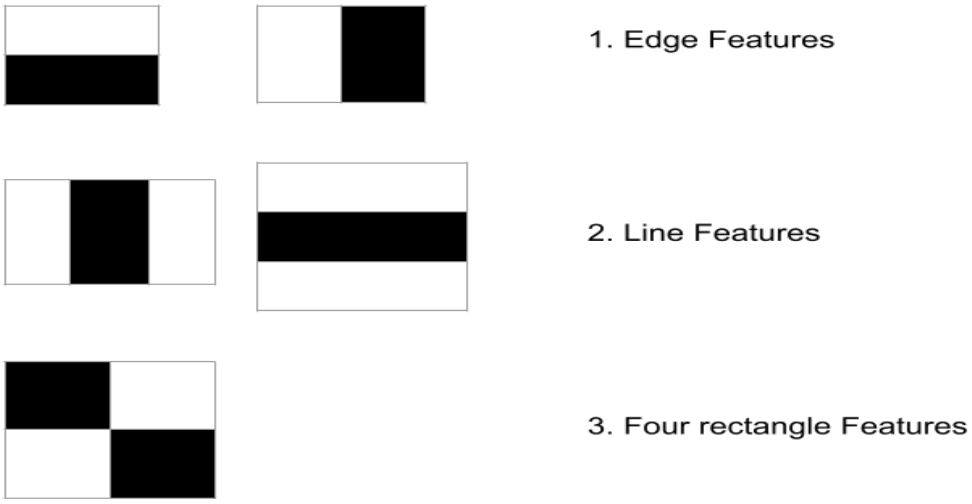
A Viola Jones detector consists of following steps :

1. Calculating Integral Image
2. Calculating Haar like features
3. AdaBoost Learning Algorithm
4. Cascade Filter

3.3.1.2 What are Haar like Features?

Haar features are the relevant features for face detection. It was proposed by Alfred Haar in 1909. They are like convolutional kernels. There are various types of haar like features but the most dominant features used are :

1. 2 Rectangular Haar features
2. 3 Rectangular Haar features
3. 4 Rectangular Haar features



The values of a 2 rectangular feature is the difference between the sum of the pixels within 2 rectangular regions. The regions have same shape and size and are horizontally and vertically adjacent. A three rectangular feature computes the sum in a centre rectangle. Finally a four rectangular feature computes the difference between diagonal pairs of rectangles. Various variations of these regions of different sizes are convolved through the image in order to get multiple filters that will be inputs to the AdaBoost training algorithm. Calculation of these features using the standard technique would require a high computation time. In order to reduce this time, a new approach called the integral image was suggested by the authors of the paper.

3.3.1.3 What is an integral Image?

Because we have to use haar-like features in all possible sizes and locations which eventually result in around 200k features to calculate which is a really big number. The problem with novel calculation of haar features is that we have to calculate the average of a given region multiple times and the time complexity of these operations are O(n*n). We can use an integral image approach to achieve O(1) running time. A given pixel in the integral image is the sum of all the pixels on the left and all the pixels above it.

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

Input Image

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

Integral Image

64	2	3	61	60	6	7	57
9	55	54	12	13	51	50	16
17	47	46	20	21	43	43	24
40	26	27	37	36	30	31	33
32	34	35	29	28	38	39	25
41	23	22	44	45	19	18	48
49	15	14	52	53	11	10	56
8	58	59	5	4	62	63	1

Original Image

64	66	69	130	190	196	203	260
73	130	187	260	333	390	446	520
90	194	297	390	484	584	683	780
130	260	390	520	650	780	910	1040
162	326	491	650	808	976	1145	1300
203	390	577	780	983	1170	1357	1560
252	454	655	910	1166	1364	1561	1820
260	520	780	1040	1300	1560	1820	2080

Integral Image

The sum of all purple boxes in the original image is equal to the sum of green boxes in the integral image subtracted by the purple boxes in the integral image.

Calculation of Haar like features with Integral Image

Using integral images we can achieve constant time evaluation of Haar features.

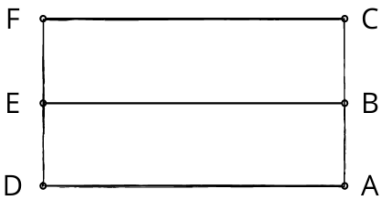
1. Edge Features or 2 Rectangular Features requires only 6 memory lookups.

2. Line Features or 3 Rectangular Features requires only 8 memory lookups.

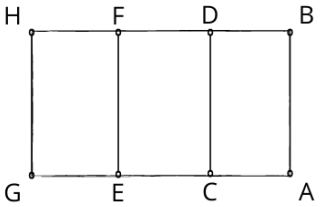
3. Diagonal Features or 4 Rectangular Features requires only 9 memory lookups.

2 Rectangle *= A-2B+C-D+2E-F* ***3 Rectangle*** *= A-B-2C+2D+2E-2F-G+H*

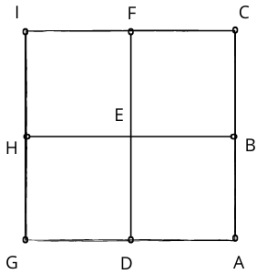
4 Rectangle *= A-2B+C-2D+4E-2F+H-2I+J*



2 Rectangle Feature



3 Rectangle Feature



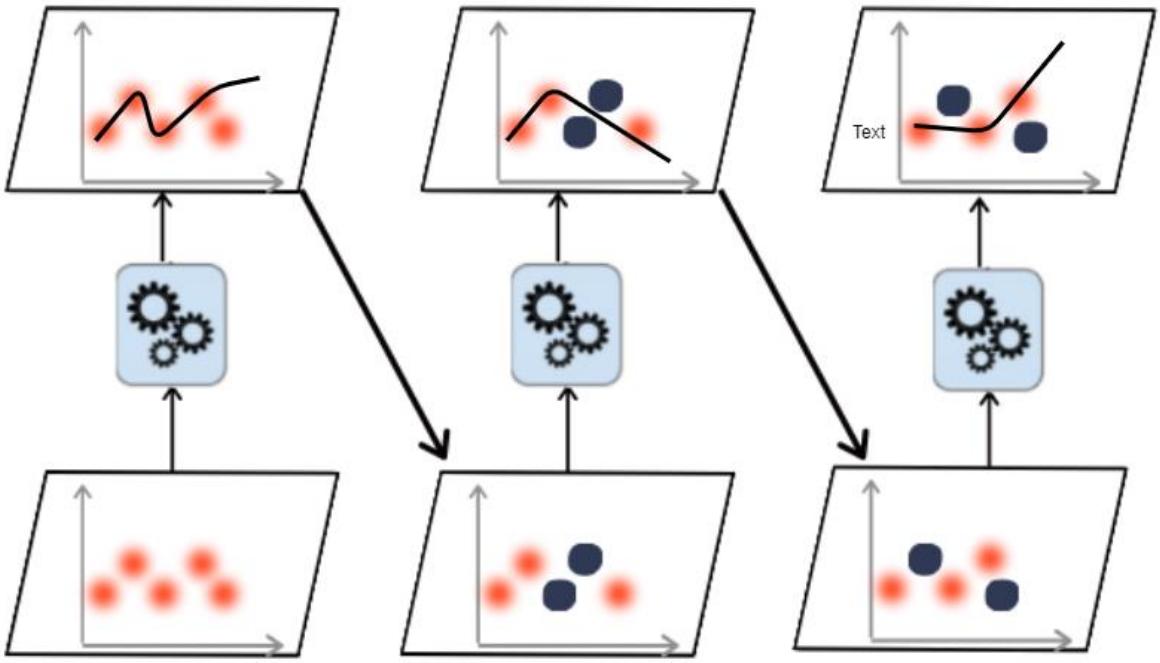
4 Rectangle Feature

Technique for calculation of sum of regions for calculation of haar like features in a constant amount of time. (Image by author)

3.3.1.4 Boosting and AdaBoost Algorithm

Boosting refers to any Ensemble method that can combine several weak learners into a strong learner. The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor. AdaBoost also known as Adaptive Boosting is one of the most popular Boosting techniques used.

AdaBoost : One way for a new predictor to correct its predecessor is to pay a bit more attention to the training instances that the predecessor under-fitted. This results in new predictors focusing more and more on the hard cases. This is known as Adaptive Boosting. For example, to build an Adaptive Boosting classifier, a first base classifier (such a Decision Tree or SVM classifier) is trained and used to make predictions on the training set. The relative weights of the misclassified predictions are altered and increased in order to lay more emphasis on these predictions while making the next predictor. A second classifier is trained using the updated weights and again it makes predictions on the training set, weights are updated and so on. Once all the predictions are trained, the ensemble method makes predictions very much like boosting except the predictors have different weights depending on their overall accuracy on the weighted training set. The drawback of this type of algorithm is that it cannot be parallelized thereby increasing time required. Thus after successfully running AdaBoost on all the features we are left with the most relevant features required for detection. Therefore, this reduces computational time as we don't have to go through all the features and is much more efficient.



Deep Dive into AdaBoost Algorithm

Let's take a closer look at the AdaBoost algorithm. Each instance weight w(i) is initially set to 1/m. A first predictor is trained and its weighted error rate r1 is computed on the training set

$$r_j = \sum_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)}$$

Here we take only the misclassified instances and sum up the weights of those instances to get the weighted error rate (Image by author)

The predictor's weight j is then computed using the formulae given below. The more accurate the predictor is, the higher its weight will be. If it is just guessing randomly, then its weight will be close to zero. However, most often, it is wrong and its weight will be negative.

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$

(Image by author)

Next the instance weights are updated using the formula provided below in order to boost the misclassified instances

$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & if\ \widehat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} \exp(\alpha_j) & if\ \widehat{y}_j^{(i)} \neq y^{(i)} \end{cases}$$

(Image by author)

Then all the predictors are normalized using the formulae provided below.

$$w^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$

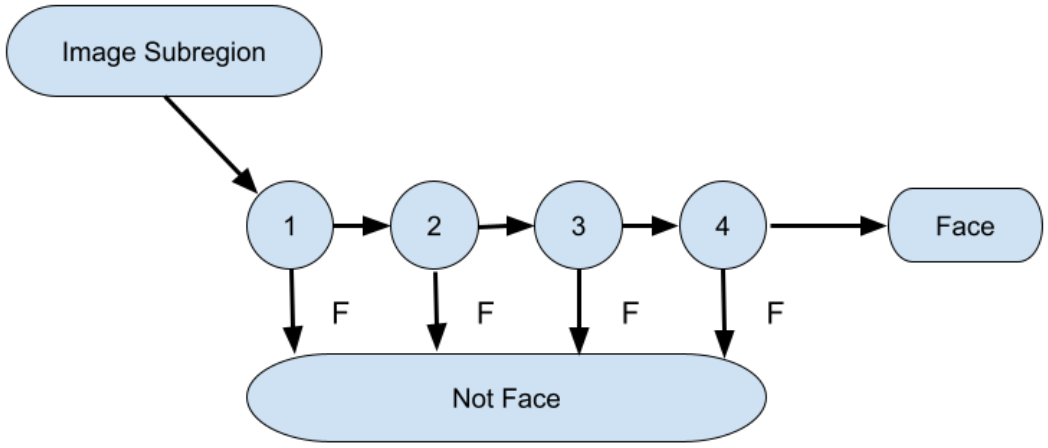
(Image by author)

Finally, a new predictor is trained using the updated weights, and the whole process is repeated until the desired number of predictors is reached which is specified by the user.

During inference, AdaBoost simply computed the predictions of all the predictors and weights then using the predictor weight αj. The predicted class is the one that receives the majority of weighted votes.

Cascade Filter

- Strong features are formed into a binary classifier. : Positive matches are sent along to the next feature. Negative matches are rejected and exit computation.
- Reduces the amount of computation time spent on false windows.
- Threshold values might be adjusted to tune accuracy. Lower threshold yield higher detection rated and more false positives.



In simple terms, each feature acts as a binary classifier in a cascade filter. If an extracted feature from the image is passed through the classifier and it predicts that the image consists of that feature then it is passed on to the next classifier for next feature existence check otherwise it is discarded and next image is checked. This thereby decreases computation time as we have to check only some features in windows where the object is not present rather than checking all features. This is the main part of the algorithm that allows it to process videos at a rate of approximately 15 frames per second and enables real time implementation.

4 Sample Code:

TrafficLightDetction.py

```
import cv2
from ColorDetection import ColorDetection
cd=ColorDetection()
faceCascade= cv2.CascadeClassifier("haarcascades/cascade7.xml")
vid=cv2.VideoCapture(0)
vid.set(3,640)
vid.set(4,480)
count=1
while True:
    res,img = vid.read()
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(imgGray,1.1,2) #imgname,scalefactor,nearest point
    # crop_img=img
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(1,0,0),2)
        crop_img=img[y:y+h,x:x+w]
        count+=1
        print("Traffic Light Found")
        cv2.imwrite("croped img/"+str(w) + str(h) + '_traffic'+str(count)+'.jpg', crop_img)
        path="croped img/"+str(w) + str(h) + '_traffic'+str(count)+'.jpg'
        print("output ",cd.detectColor(path))

    cv2.imshow("Result", img)
    if cv2.waitKey(1000) & 0xFF==ord('q'):
        break
```

ColorDetection.py:

```
import cv2
import numpy as np
class ColorDetection:
  def __init__(self):
    a=7
  def detectColor(self,path):
    img = cv2.imread(path)
    imgHSV=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
    kernal = np.ones((5,5), "uint8")

    # print("HSV image matrix:\n")
```

```
# print(imgHSV)
# red color
red_lower = np.array([0, 115, 14], np.uint8)
red_upper = np.array([2, 255, 255], np.uint8)
red_mask = cv2.inRange(imgHSV, red_lower, red_upper)
red_mask =cv2.dilate(red_mask,kernal)
red_result=cv2.bitwise_and(img,img,mask=red_mask)
# cv2.imshow("RED",red_result)

#amber color
amber_lower = np.array([21, 129, 43],np.uint8)
amber_upper = np.array([26, 255, 255],np.uint8)
amber_mask=cv2.inRange(imgHSV,amber_lower,amber_upper)
amber_mask =cv2.dilate(amber_mask,kernal)
amber_result=cv2.bitwise_and(img,img,mask=amber_mask)
# cv2.imshow("AMBER",amber_result)

# green color
green_lower = np.array([47,147, 10],np.uint8)
green_upper = np.array([102, 255, 255],np.uint8)
green_mask=cv2.inRange(imgHSV,green_lower,green_upper)
green_mask =cv2.dilate(green_mask,kernal)
green_result=cv2.bitwise_and(img,img,mask=green_mask)
# cv2.imshow("GREEN",green_result)
f=1
# Creating contour to track red color
contours, hierarchy = cv2.findContours(red_mask,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)
if f==1:
    for pic, contour in enumerate(contours):
        area = cv2.contourArea(contour)
        if(area > 1000):
            x, y, w, h = cv2.boundingRect(contour)
            img = cv2.rectangle(img, (x, y),
                                       (x + w, y + h),
                                       (0, 0, 255), 2)
            return ("STOP")
            f=0
            cv2.putText(img, "STOP", (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                        (0, 0, 255))
            break

contours, hierarchy = cv2.findContours(amber_mask,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)
if f==1:
    for pic, contour in enumerate(contours):
        area = cv2.contourArea(contour)
        if(area > 1000):
            x, y, w, h = cv2.boundingRect(contour)
            img = cv2.rectangle(img, (x, y),
                                       (x + w, y + h),
```

```
                                (0, 0, 255), 2)

            return("GET READY")
            f=0
            cv2.putText(img, "GET READY", (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                        (0, 0, 255))
    # Creating contour to track green color
    contours, hierarchy = cv2.findContours(green_mask,
                                cv2.RETR_TREE,
                                cv2.CHAIN_APPROX_SIMPLE)
    if f==1:
        for pic, contour in enumerate(contours):
            area = cv2.contourArea(contour)
            if(area > 10):
                x, y, w, h = cv2.boundingRect(contour)
                img = cv2.rectangle(img, (x, y),
                                (x + w, y + h),
                                (0, 0, 255), 2)
                return("GO")
                f=0
                cv2.putText(img, "Go", (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.5,
                        (0, 0, 255))
```

5. Testing, Results, Discussion on Results:

We tested our trained model on different images that we captured using our laptop camera and images were displayed from our mobile handset.

Camera was capturing real time images and from there traffic light image is taken and cropped and save in a separate folder and then using that folder path , the traffic light color is detected.

The accuracy of traffic light detection is approx 89% And color detection accuracy is approx 78%.

Some successful predictions by our model:

Input image:

Output image:

Input Image:

Output Image:

Console output:

1st input:

Traffic Light Found

output GO

2nd input:

Traffic Light Found

output None

explanation: In the 1st case our model detect the traffic light from the given image and as the traffic light is green its print go in console.

For 2nd case model detect the traffic light but it can't detect the color of traffic light as all are on. So the output in console is None.

Some successful predictions by our model:

Input:

output:

Can't detect the traffic light

6.

6.1 Scope for future environment:

We tried to implement a new accurate strategies to detect traffic light in real time using haar cascade and it gave us a good output as well as successfully detected most of the objects (traffic light). But, we are not fully aware how this will perform under low light or less visual features. We have to increase our accuracy for color classification. It can be made more robust by eliminating some of the limitations as listed below:

- In the Multiple Visual tracking the templates and their accuracy starts to deteriorate.
- Fully occluded traffic light cannot be tracked.
- To make the system fully automatic and also to overcome the above limitations, in future we have to work on color classification for better output.
- We would try to improve the accuracy with fine tuning the weights and hope to capture more details in a challenging or an obstructed environment.

6.2 Conclusion:

Traffic Light Detection is not a trivial task. In case of machine learning approaches, feature extraction is manual. The segmentation thus relies heavily on the methods followed for feature extraction and the quality and relevance of the extracted features. Connected components based feature extraction algorithms were found to be pretty effective in this context. But, the major drawback of this approach is that the algorithm in theory at least, is not generalizable.

References /Bibliography

1. https://www.python.org/
2. https://code.visualstudio.com/
3. https://docs.anaconda.com/anaconda/navigator/
4. https://amin-ahmadi.com/cascade-trainer-gui/
5. https://numpy.org/
6. https://pandas.pydata.org/
7. https://opencv.org/
8. https://en.wikipedia.org/wiki/Machine_learning
9. https://amin-ahmadi.com/wp-content/uploads/2016/10/cascade-trainer-gui-01-train-input.png
10. https://amin-ahmadi.com/wp-content/uploads/2016/10/cascade-trainer-gui-02-train-common.png
11. https://amin-ahmadi.com/wp-content/uploads/2016/10/cascade-trainer-gui-04-train-boost.png
12. https://www.aitrends.com/ai-insider/traffic-lights-and-ai-autonomous-cars/
13. https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8
14. https://www.researchgate.net/figure/Cascade-classifier-illustration-2_fig2_323057610
15. https://m.blog.naver.com/natalliea/222198638897
16. https://www.mathworks.com/help/images/integral-image.html
17. https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html
18. https://amin-ahmadi.com/cascade-trainer-gui/
19. https://medium.com/@vipulgote4/guide-to-make-custom-haar-cascade-xml-file-for-object-detection-with-opencv-6932e22c3f0e