Vehicle Detection

To build a vehicle detection program, we can use haar cascades. Using the coordinates of the vehicles detected, we can detect and display the colour of the cars using OpenCV.

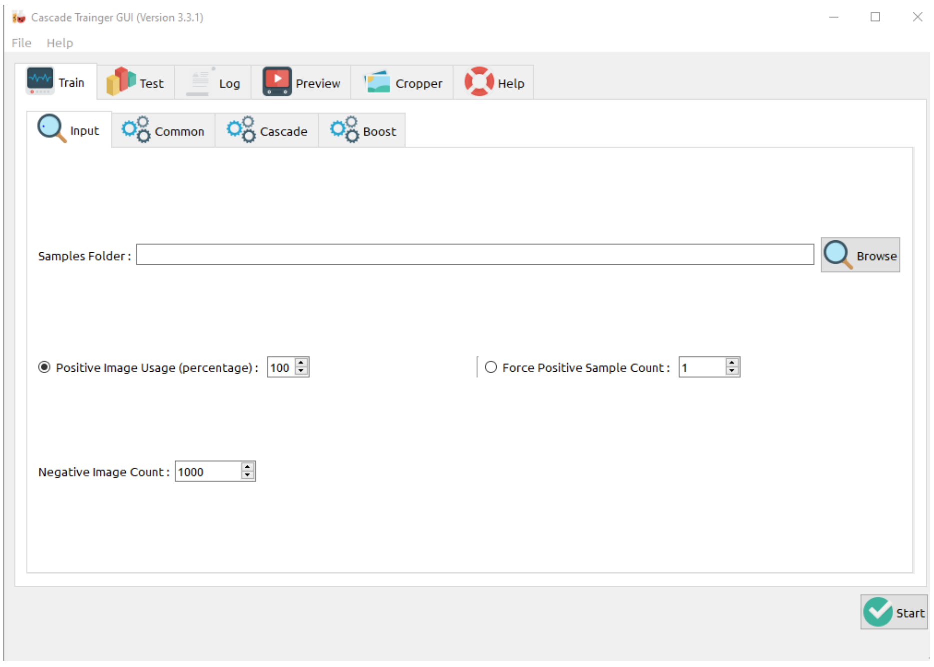
Haar Cascade

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features. The Classifier is trained using a lot of positive and negative images.

Positive images – These images contain the images which we want our classifier to identify.

Negative Images – Images of everything else, which do not contain the object we want to detect. These must never include any positive images, not even partial.

To create a cascade, we use a cascade trainer. 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.



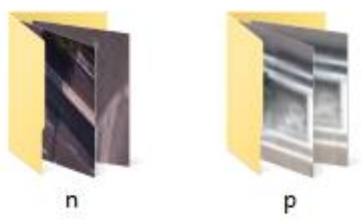
Samples Folder :

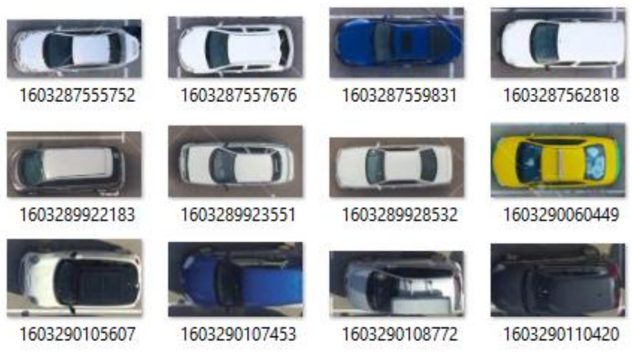
Positive Image Usage (percentage) : 100

Force Positive Sample Count : 1

Negative Image Count : 1000

To do this we create a folder named ‘p’ containing only positive images and a folder named ‘n’ containing only negative images. The number of negative images is approximately twice the number of positive images. Both folders are kept in the same location.

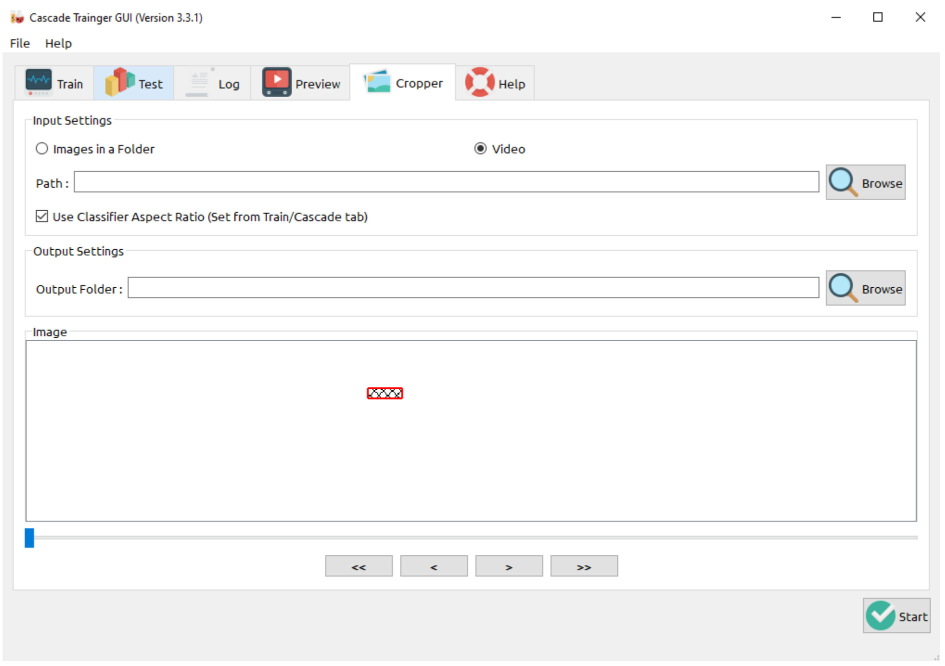




Positive images

Negative images

The images can be cropped into positive and negative images and stored in the specific folders using the cropper function available in the cascade trainer.



Input Settings

Images in a Folder [✓ Video]

Path :

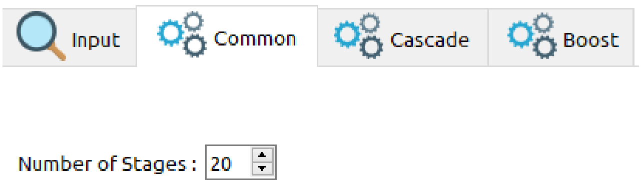
[✓] Use Classifier Aspect Ratio (Set from Train/Cascade tab)

Output Settings

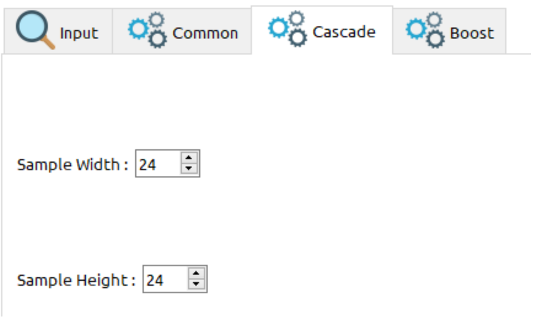
Output Folder :

Image

After creating and storing all images, we can train the classifier. This is done by giving the location where the positive and negative folders are contained. Then the number of stages and the dimensions of the object to be detected is entered. Higher number of stages decreases the percentage of false positives, but will take more time to complete training.



Number of Stages : 20



Sample Width : 24

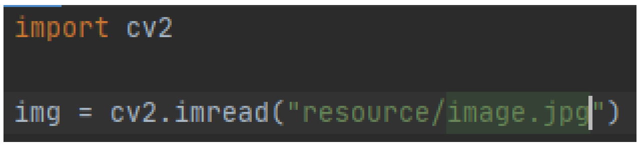
Sample Height : 24

After setting the conditions, the training is started. When is training is finished, a new folder is present at the same location as the positive and negative folders.

Inside the classifier folder, there are many files, but we only need the file named 'cascade.xml'. This file is copied to a resources folder.

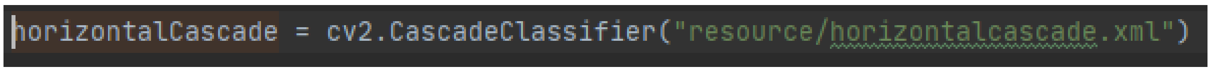
OpenCV – Boundary Boxes

To create the boundary boxes around the object detected, openCV is used. First, the test image is copied to the resources folder. Then it is imported using openCV.



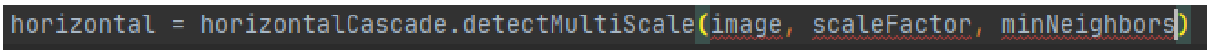
```
import cv2

img = cv2.imread("resource/image.jpg")
```

Then the cascade created is imported.



```
horizontalCascade = cv2.CascadeClassifier("resource/horizontalcascade.xml")
```

A function, 'detectMultiScale', is used.



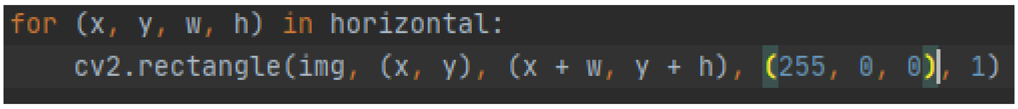
```
horizontal = horizontalCascade.detectMultiScale(image, scaleFactor, minNeighbors)
```

scaleFactor specifies how much the image size is reduced at each image scale.

minNeighbors specifies how many neighbors each candidate rectangle should have to retain it.

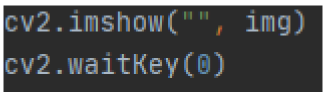
These values are set with trial and error. This function returns a list of the coordinated of the rectangles which enclose the detected objects.

Using a simple loop and rectangle function, all detected objects can be shown.

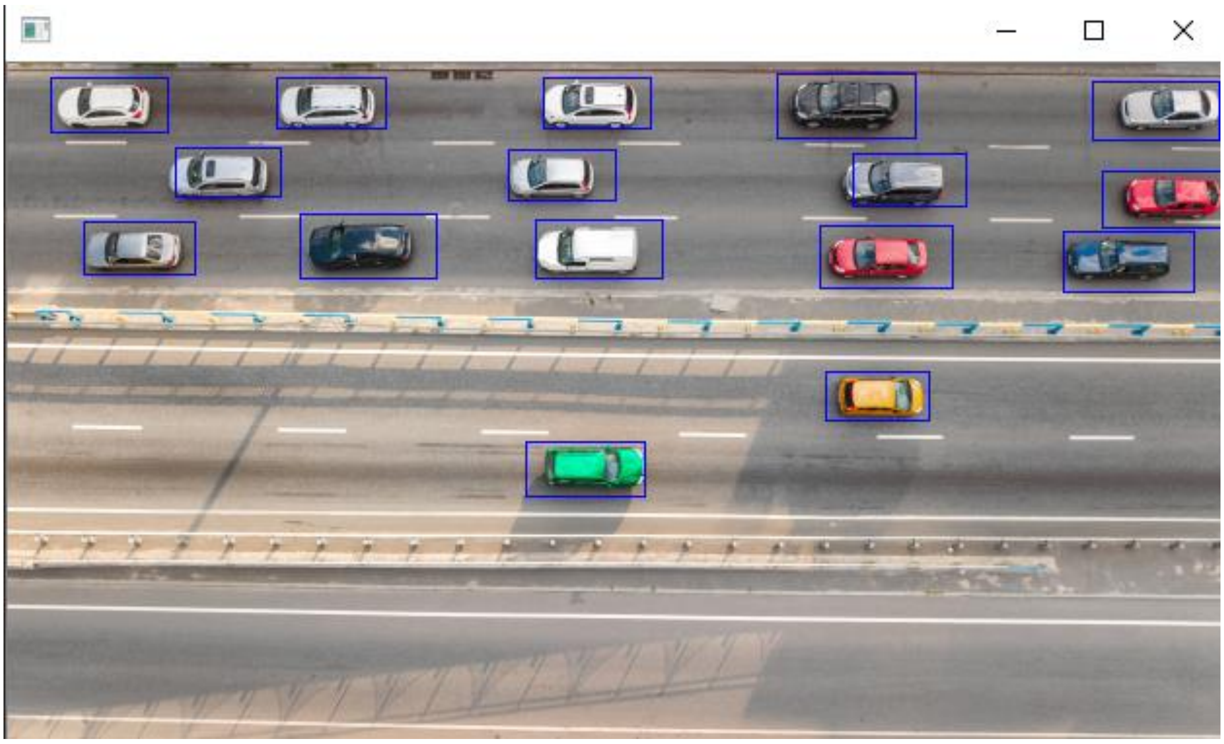


```
for (x, y, w, h) in horizontal:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 1)
```

The image is then displayed.

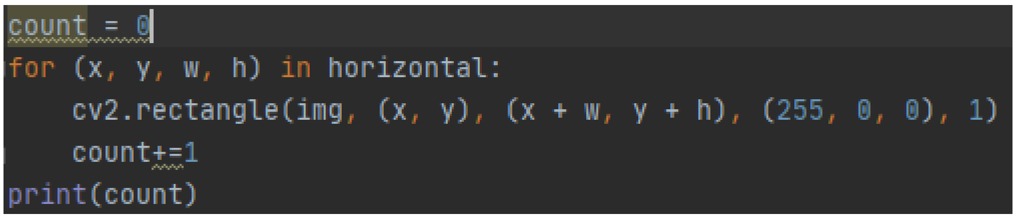


```
cv2.imshow("", img)
cv2.waitKey(0)
```



OpenCV - Count

To display the number of images, a variable can be declared outside the loop and iterated with every rectangle made.

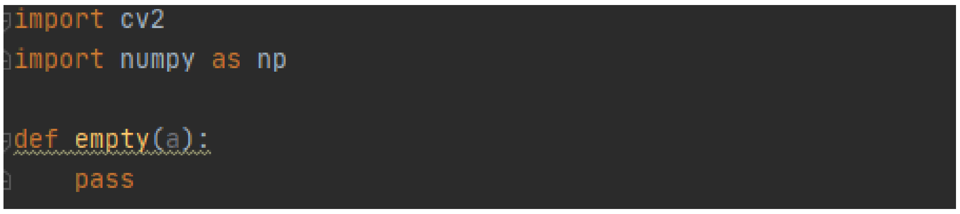


```
count = 0
for (x, y, w, h) in horizontal:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 1)
    count+=1
print(count)
```

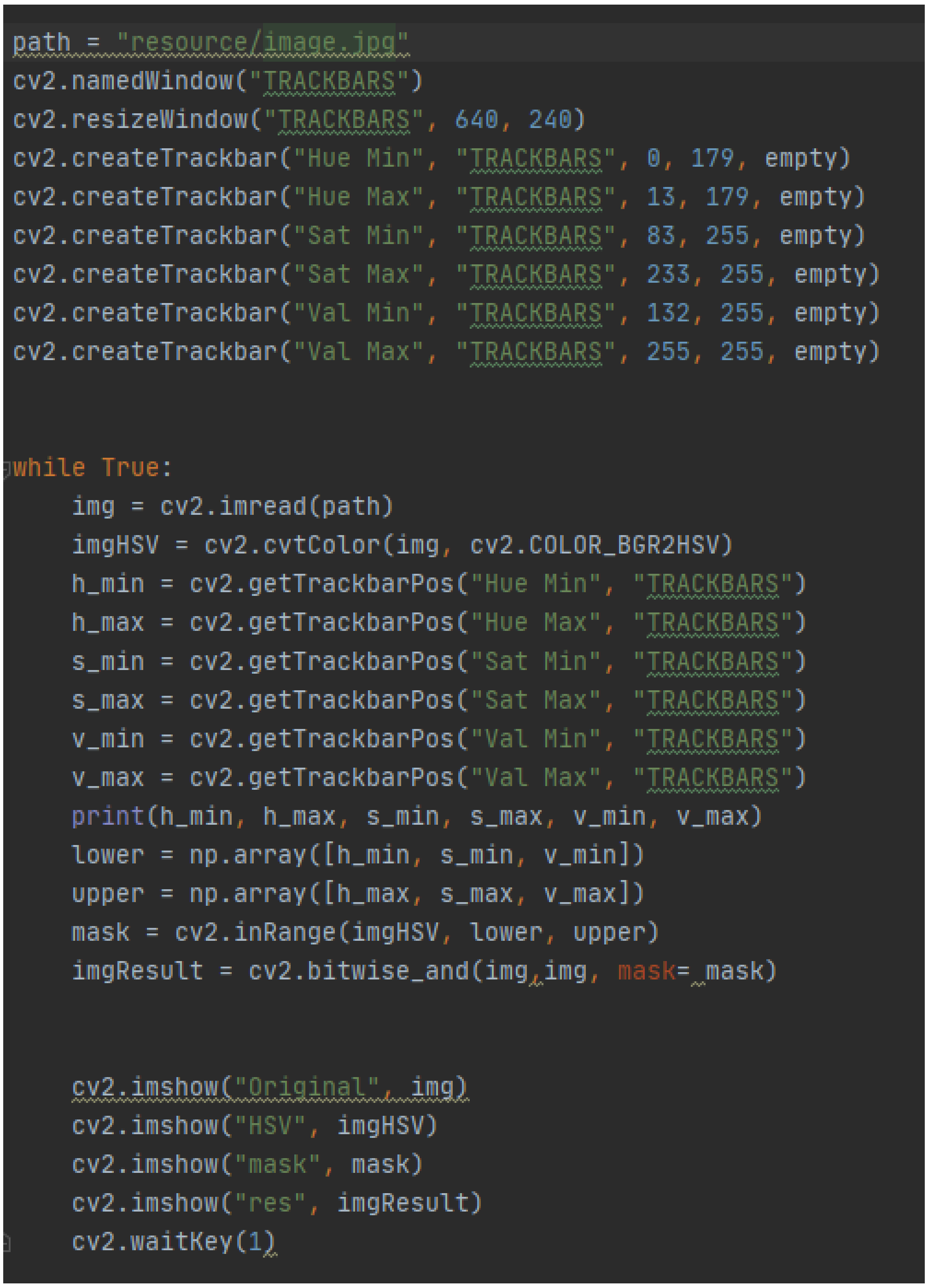
OpenCV – Colour

To detect colours we convert the image to hsv colour channel. This is done as it is easier to define colour thresholds in this channel.

Colour thresholds are defined by using an OpenCV trackbar program.

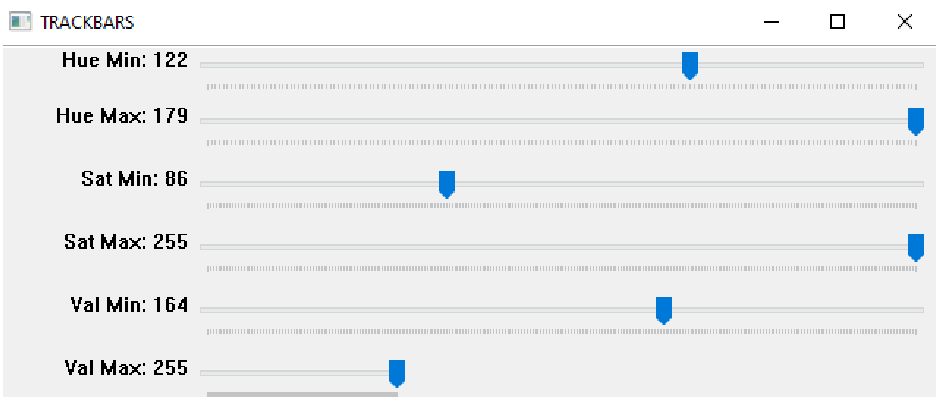


```
import cv2
import numpy as np

def empty(a):
    pass
```



```
path = "resource/image.jpg"
cv2.namedWindow("TRACKBARS")
cv2.resizeWindow("TRACKBARS", 640, 240)
cv2.createTrackbar("Hue Min", "TRACKBARS", 0, 179, empty)
cv2.createTrackbar("Hue Max", "TRACKBARS", 13, 179, empty)
cv2.createTrackbar("Sat Min", "TRACKBARS", 83, 255, empty)
cv2.createTrackbar("Sat Max", "TRACKBARS", 233, 255, empty)
cv2.createTrackbar("Val Min", "TRACKBARS", 132, 255, empty)
cv2.createTrackbar("Val Max", "TRACKBARS", 255, 255, empty)


while True:
    img = cv2.imread(path)
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h_min = cv2.getTrackbarPos("Hue Min", "TRACKBARS")
    h_max = cv2.getTrackbarPos("Hue Max", "TRACKBARS")
    s_min = cv2.getTrackbarPos("Sat Min", "TRACKBARS")
    s_max = cv2.getTrackbarPos("Sat Max", "TRACKBARS")
    v_min = cv2.getTrackbarPos("Val Min", "TRACKBARS")
    v_max = cv2.getTrackbarPos("Val Max", "TRACKBARS")
    print(h_min, h_max, s_min, s_max, v_min, v_max)
    lower = np.array([h_min, s_min, v_min])
    upper = np.array([h_max, s_max, v_max])
    mask = cv2.inRange(imgHSV, lower, upper)
    imgResult = cv2.bitwise_and(img,img, mask= mask)


    cv2.imshow("Original", img)
    cv2.imshow("HSV", imgHSV)
    cv2.imshow("mask", mask)
    cv2.imshow("res", imgResult)
    cv2.waitKey(1)
```

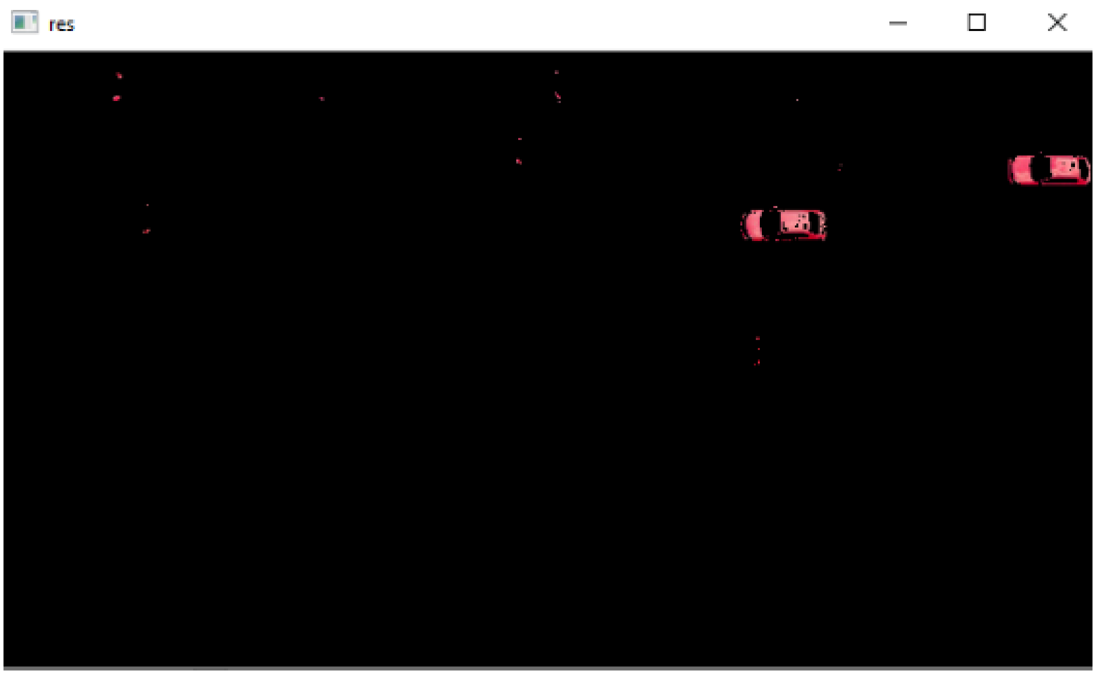
Output:



TRACKBARS

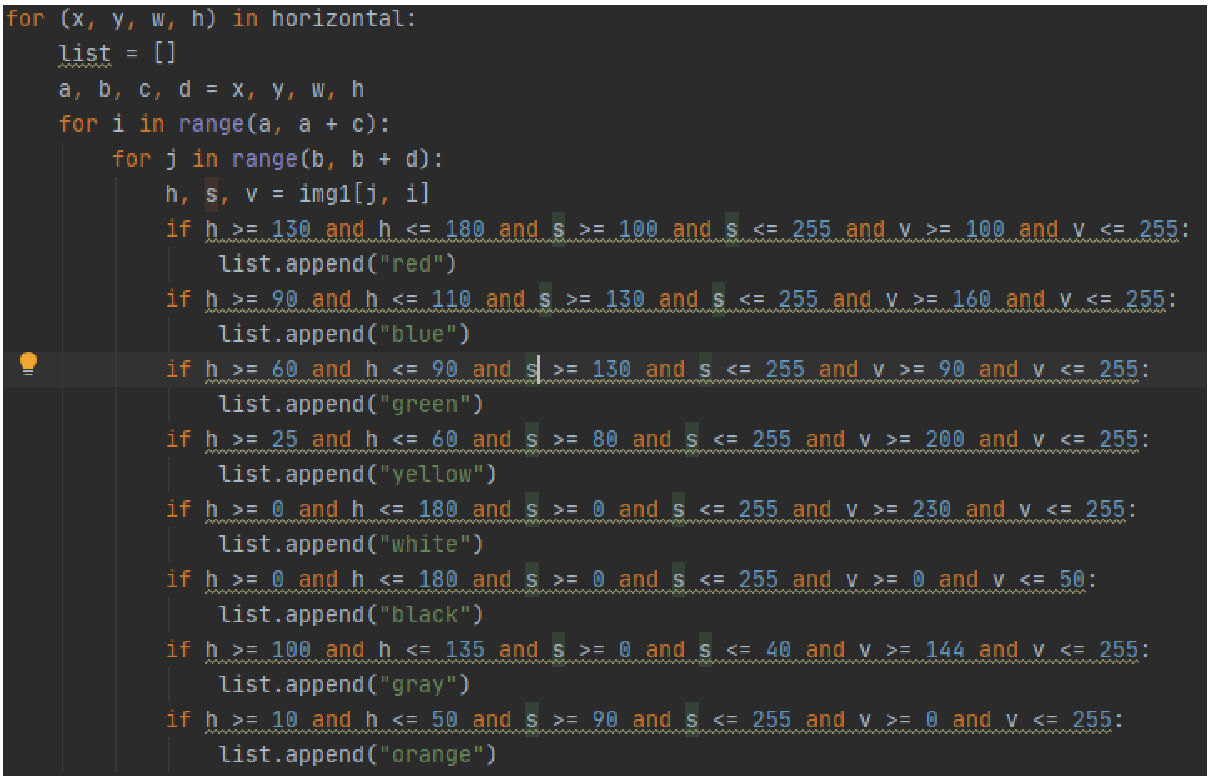
Hue Min: 122
Hue Max: 179
Sat Min: 86
Sat Max: 255
Val Min: 164
Val Max: 255

This is used to detect thresholds for certain colours. For example in the image, the thresholds are set for red.



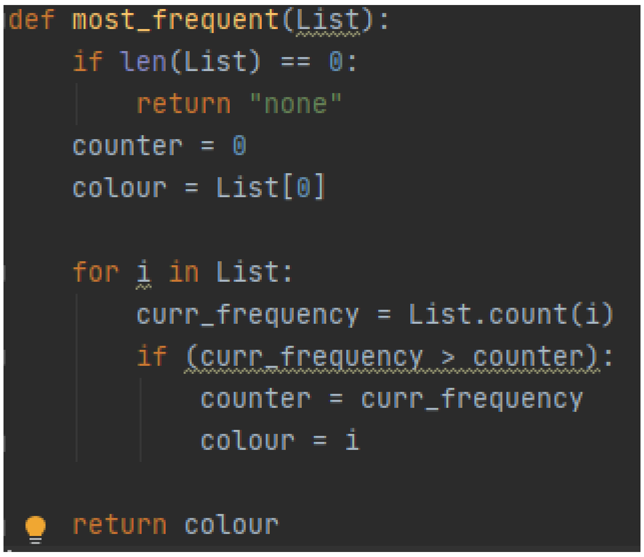
Using this method, thresholds for all colours can be created.

To detect the colours of the cars, a loop is used which iterates through each pixel enclosed by the boundary box. Using this, each pixel is clasiffied as a certain colour and added to a list.



```
for (x, y, w, h) in horizontal:
    list = []
    a, b, c, d = x, y, w, h
    for i in range(a, a + c):
        for j in range(b, b + d):
            h, s, v = img1[j, i]
            if h >= 130 and h <= 180 and s >= 100 and s <= 255 and v >= 100 and v <= 255:
                list.append("red")
            if h >= 90 and h <= 110 and s >= 130 and s <= 255 and v >= 160 and v <= 255:
                list.append("blue")
            if h >= 60 and h <= 90 and s >= 130 and s <= 255 and v >= 90 and v <= 255:
                list.append("green")
            if h >= 25 and h <= 60 and s >= 80 and s <= 255 and v >= 200 and v <= 255:
                list.append("yellow")
            if h >= 0 and h <= 180 and s >= 0 and s <= 255 and v >= 230 and v <= 255:
                list.append("white")
            if h >= 0 and h <= 180 and s >= 0 and s <= 255 and v >= 0 and v <= 50:
                list.append("black")
            if h >= 100 and h <= 135 and s >= 0 and s <= 40 and v >= 144 and v <= 255:
                list.append("gray")
            if h >= 10 and h <= 50 and s >= 90 and s <= 255 and v >= 0 and v <= 255:
                list.append("orange")
```

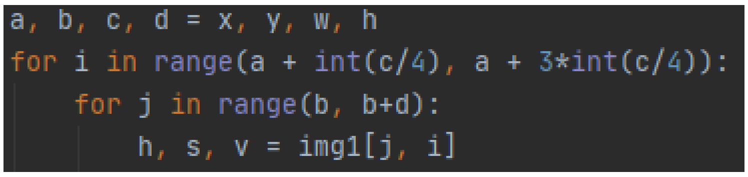
To detect the colour, a user defined most frequent function is used to find the most occuring colour.



```
def most_frequent(List):
    if len(List) == 0:
        return "none"
    counter = 0
    colour = List[0]

    for i in List:
        curr_frequency = List.count(i)
        if (curr_frequency > counter):
            counter = curr_frequency
            colour = i

    return colour
```

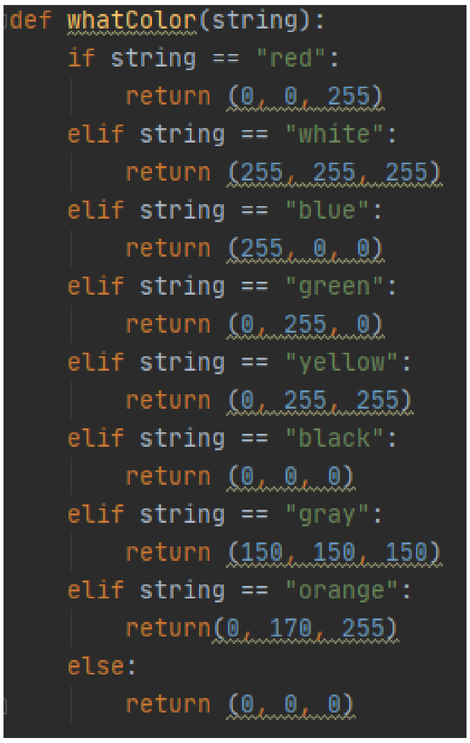
The most frequently occuring colour in the list is named the colour of the car.

To make this more accurate, the area of pixels is trimmed in the horizontal direction to decrease the amount of road pixels. This is done by editing the loop.



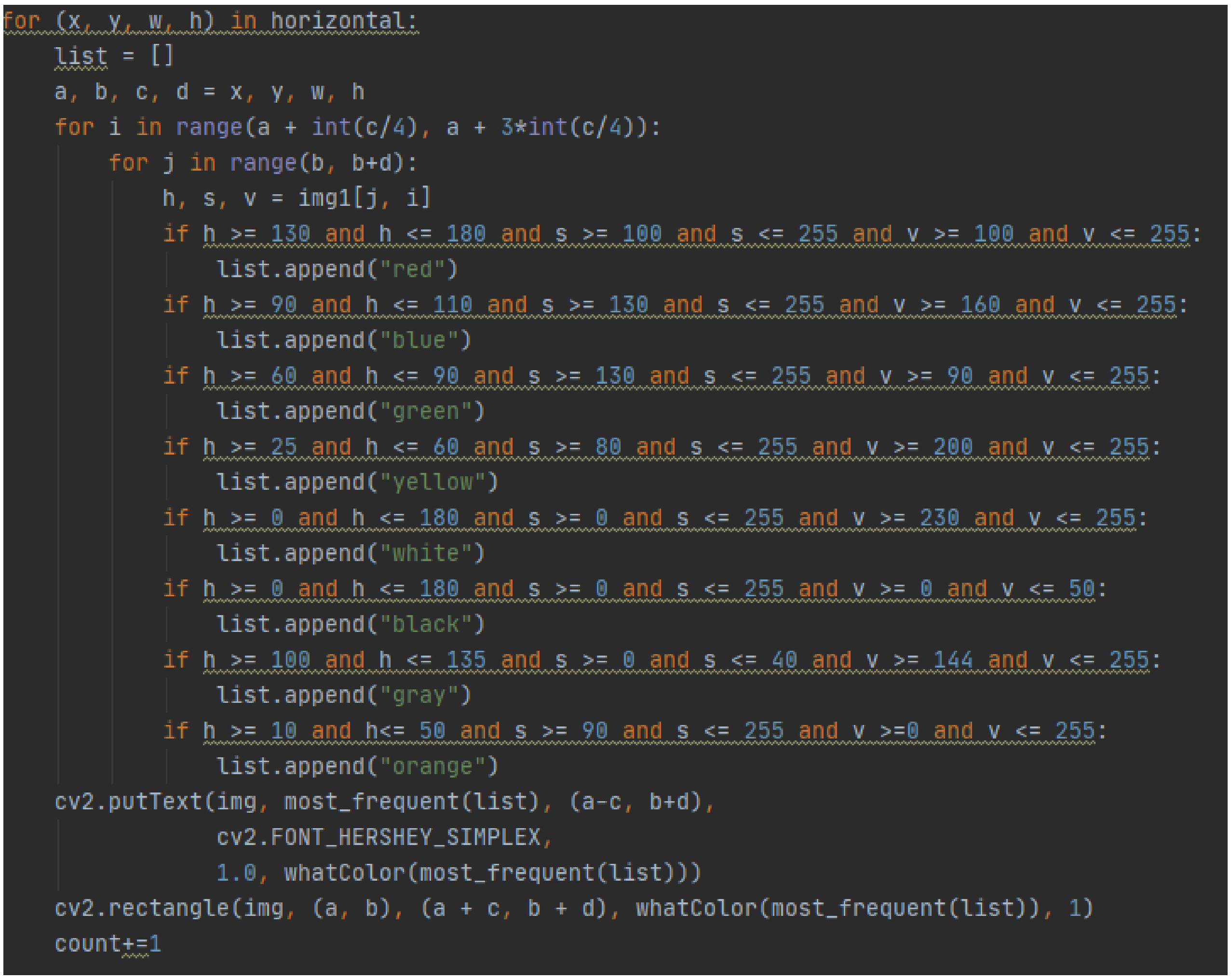
```
a, b, c, d = x, y, w, h
for i in range(a + int(c/4), a + 3*int(c/4)):
    for j in range(b, b+d):
        h, s, v = img1[j, i]
```

Another user defined function is used to convert the string colours to tuple format which can be read as BGR channel.

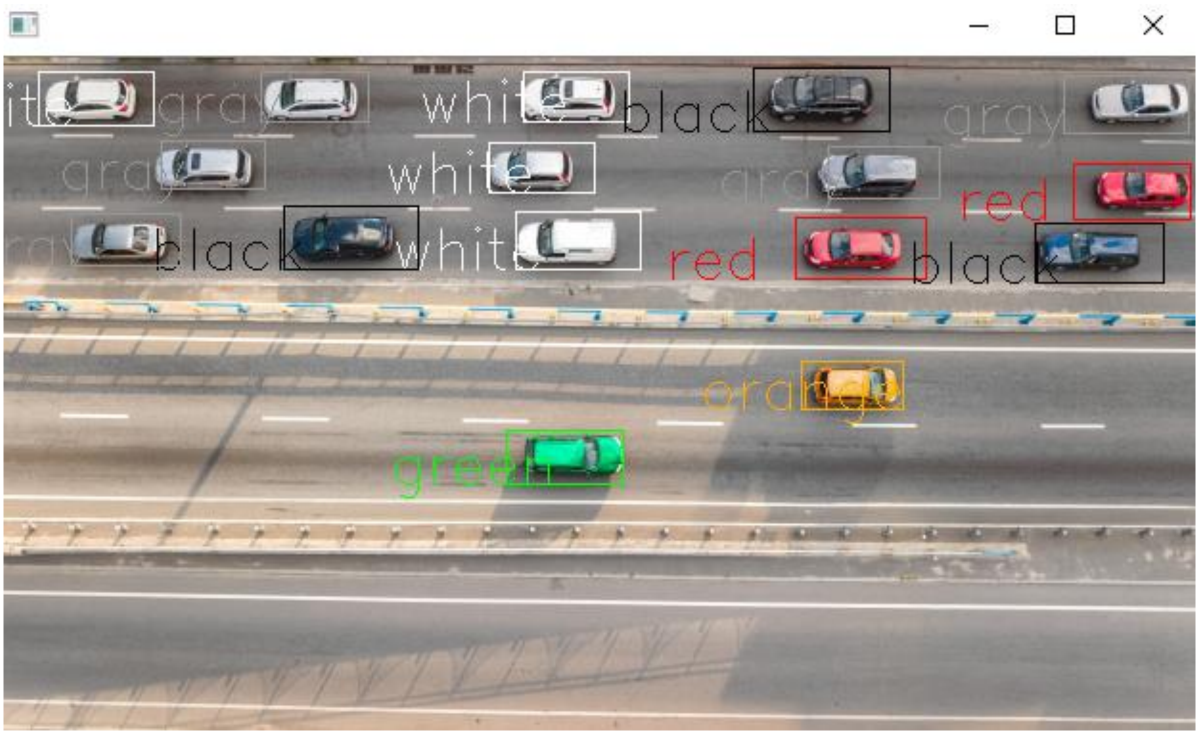


```
def whatColor(string):
    if string == "red":
        return (0, 0, 255)
    elif string == "white":
        return (255, 255, 255)
    elif string == "blue":
        return (255, 0, 0)
    elif string == "green":
        return (0, 255, 0)
    elif string == "yellow":
        return (0, 255, 255)
    elif string == "black":
        return (0, 0, 0)
    elif string == "gray":
        return (150, 150, 150)
    elif string == "orange":
        return(0, 170, 255)
    else:
        return (0, 0, 0)
```

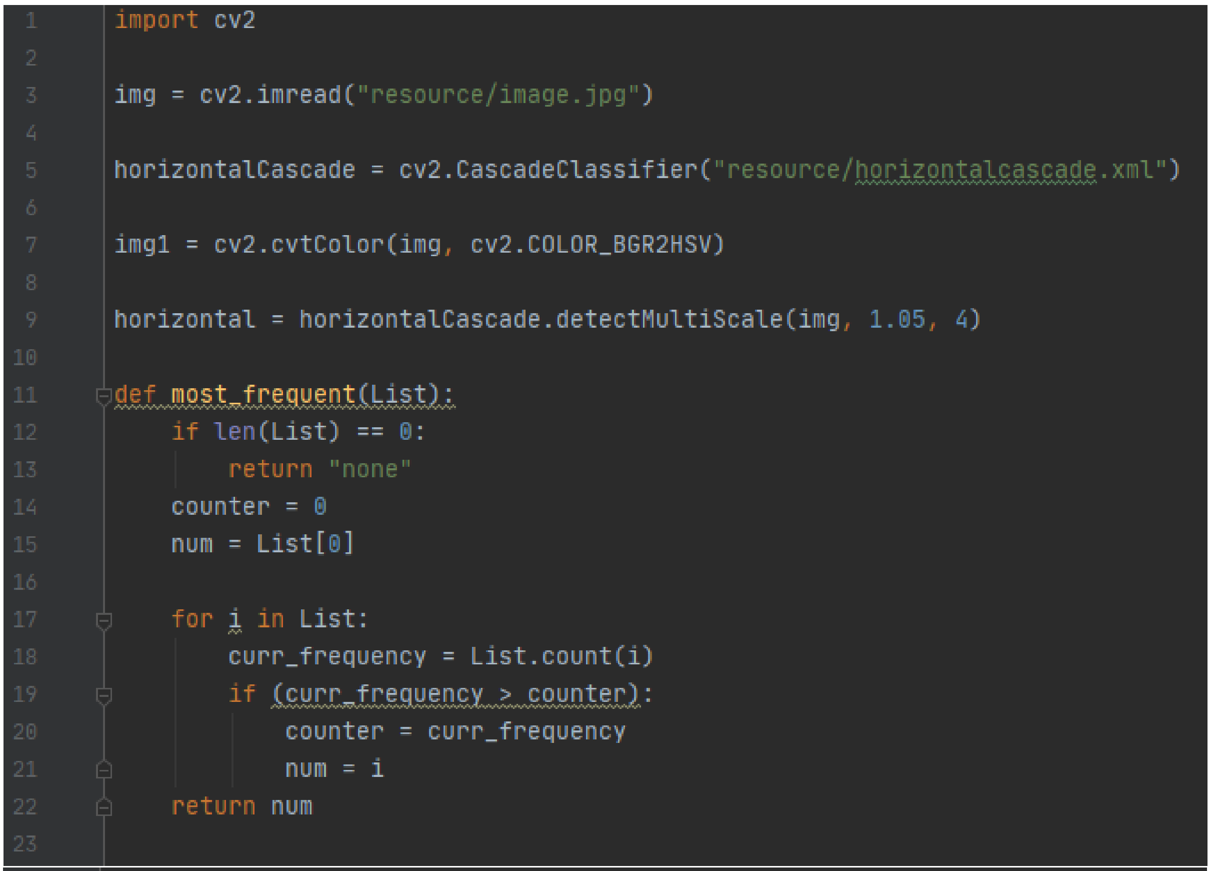
Now this is implemeted in the original loop to also display the colour of the cars.



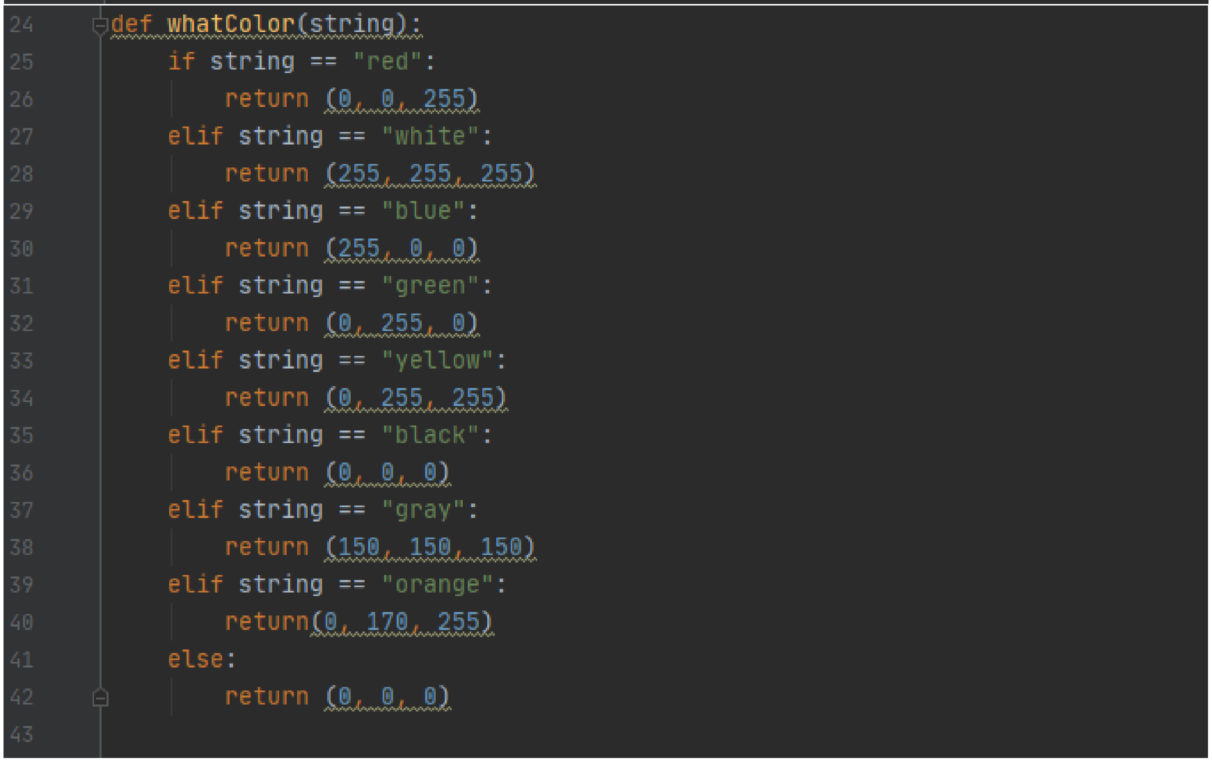
```python
for (x, y, w, h) in horizontal:
    list = []
    a, b, c, d = x, y, w, h
    for i in range(a + int(c/4), a + 3*int(c/4)):
        for j in range(b, b+d):
            h, s, v = img1[j, i]
            if h >= 130 and h <= 180 and s >= 100 and s <= 255 and v >= 100 and v <= 255:
                list.append("red")
            if h >= 90 and h <= 110 and s >= 130 and s <= 255 and v >= 160 and v <= 255:
                list.append("blue")
            if h >= 60 and h <= 90 and s >= 130 and s <= 255 and v >= 90 and v <= 255:
                list.append("green")
            if h >= 25 and h <= 60 and s >= 80 and s <= 255 and v >= 200 and v <= 255:
                list.append("yellow")
            if h >= 0 and h <= 180 and s >= 0 and s <= 255 and v >= 230 and v <= 255:
                list.append("white")
            if h >= 0 and h <= 180 and s >= 0 and s <= 255 and v >= 0 and v <= 50:
                list.append("black")
            if h >= 100 and h <= 135 and s >= 0 and s <= 40 and v >= 144 and v <= 255:
                list.append("gray")
            if h >= 10 and h<= 50 and s >= 90 and s <= 255 and v >=0 and v <= 255:
                list.append("orange")
    cv2.putText(img, most_frequent(list), (a-c, b+d),
                cv2.FONT_HERSHEY_SIMPLEX,
                1.0, whatColor(most_frequent(list)))
    cv2.rectangle(img, (a, b), (a + c, b + d), whatColor(most_frequent(list)), 1)
    count+=1
```



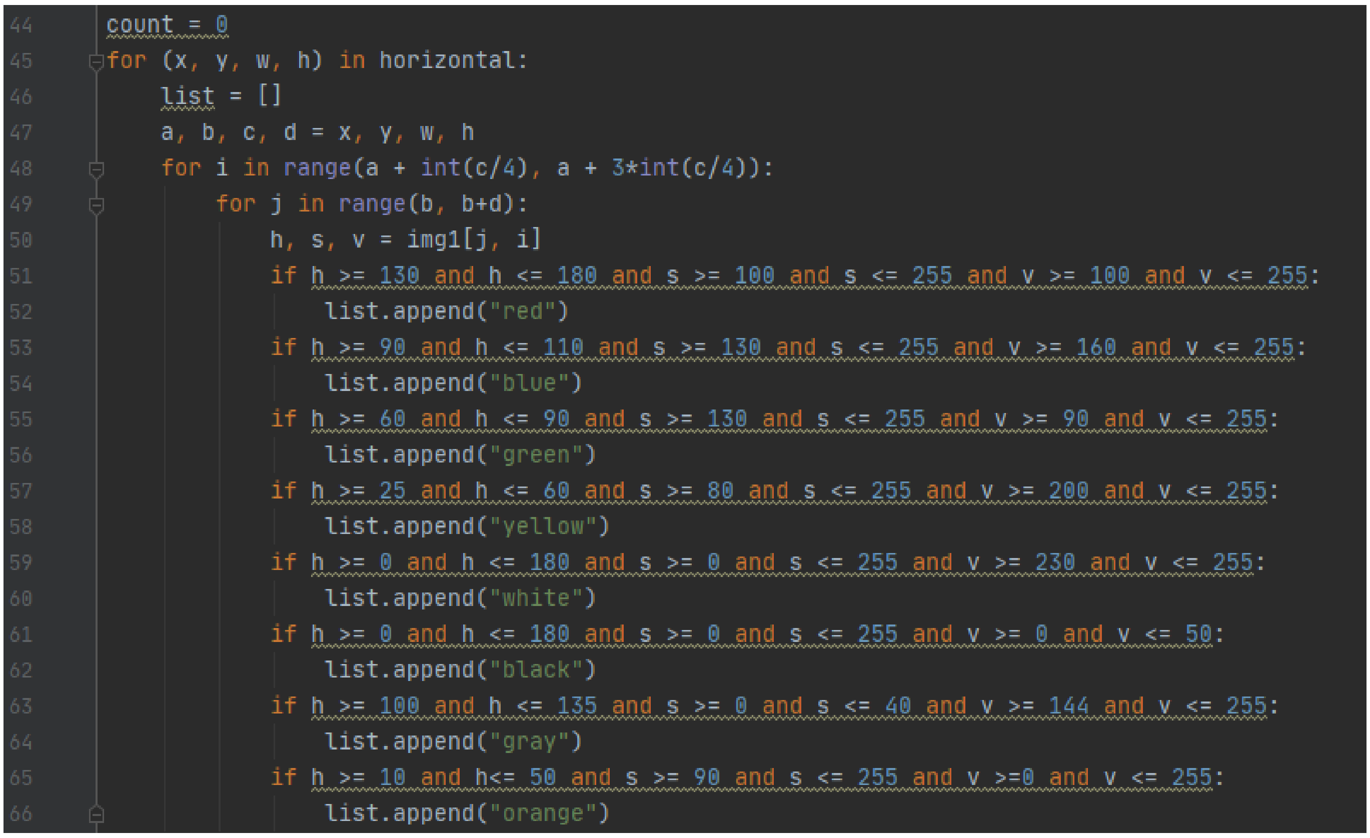
Full Code



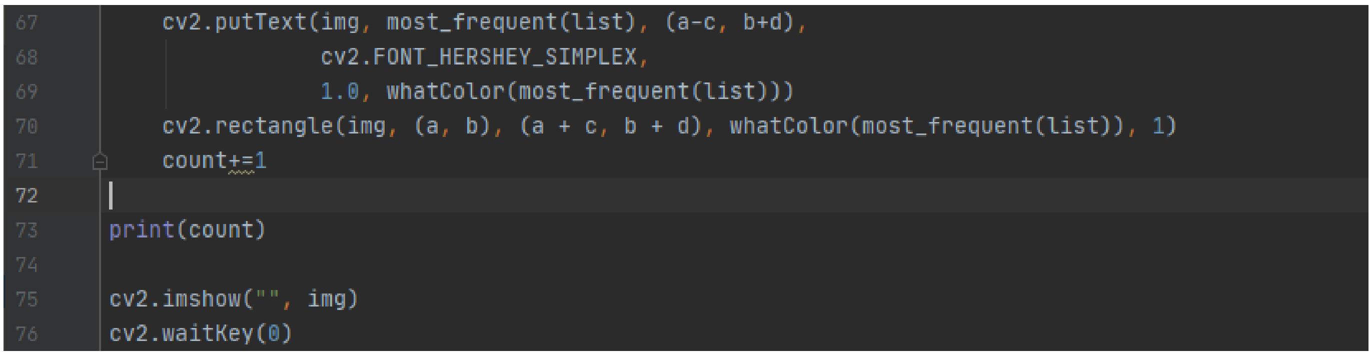
```python
import cv2

img = cv2.imread("resource/image.jpg")

horizontalCascade = cv2.CascadeClassifier("resource/horizontalcascade.xml")

img1 = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

horizontal = horizontalCascade.detectMultiScale(img, 1.05, 4)

def most_frequent(List):
    if len(List) == 0:
        return "none"
    counter = 0
    num = List[0]

    for i in List:
        curr_frequency = List.count(i)
        if (curr_frequency > counter):
            counter = curr_frequency
            num = i
    return num

```



```python
def whatColor(string):
    if string == "red":
        return (0, 0, 255)
    elif string == "white":
        return (255, 255, 255)
    elif string == "blue":
        return (255, 0, 0)
    elif string == "green":
        return (0, 255, 0)
    elif string == "yellow":
        return (0, 255, 255)
    elif string == "black":
        return (0, 0, 0)
    elif string == "gray":
        return (150, 150, 150)
    elif string == "orange":
        return(0, 170, 255)
    else:
        return (0, 0, 0)

```



```python
count = 0
for (x, y, w, h) in horizontal:
    list = []
    a, b, c, d = x, y, w, h
    for i in range(a + int(c/4), a + 3*int(c/4)):
        for j in range(b, b+d):
            h, s, v = img1[j, i]
            if h >= 130 and h <= 180 and s >= 100 and s <= 255 and v >= 100 and v <= 255:
                list.append("red")
            if h >= 90 and h <= 110 and s >= 130 and s <= 255 and v >= 160 and v <= 255:
                list.append("blue")
            if h >= 60 and h <= 90 and s >= 130 and s <= 255 and v >= 90 and v <= 255:
                list.append("green")
            if h >= 25 and h <= 60 and s >= 80 and s <= 255 and v >= 200 and v <= 255:
                list.append("yellow")
            if h >= 0 and h <= 180 and s >= 0 and s <= 255 and v >= 230 and v <= 255:
                list.append("white")
            if h >= 0 and h <= 180 and s >= 0 and s <= 255 and v >= 0 and v <= 50:
                list.append("black")
            if h >= 100 and h <= 135 and s >= 0 and s <= 40 and v >= 144 and v <= 255:
                list.append("gray")
            if h >= 10 and h<= 50 and s >= 90 and s <= 255 and v >=0 and v <= 255:
                list.append("orange")
```



```python
    cv2.putText(img, most_frequent(list), (a-c, b+d),
                cv2.FONT_HERSHEY_SIMPLEX,
                1.0, whatColor(most_frequent(list)))
    cv2.rectangle(img, (a, b), (a + c, b + d), whatColor(most_frequent(list)), 1)
    count+=1

print(count)

cv2.imshow("", img)
cv2.waitKey(0)
```