Artificial Intelligence Lab

Project Report

Submitted To: M.s Anu Bajaj

Submitted by:

102103078

(Aradhak Kandhari)

102103077

(Saransh Mahajan)

102103081

(Gaurav Arya)

102103064

(Aditya Pratap)

102103094

(Shivang Agarrwal)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Problem Statement:

To make an AI which helps in image color recognition.

Link to the code:
https://github.com/saranshisgoat/Color_detection/upload

(We had added our code in this repository.)

Description:

Image color recognition using AI involves training computer algorithms to accurately identify and categorize colors in digital images. This involves analyzing the color values of each pixel in an image, and using machine learning techniques to identify patterns and features that correspond to different colors.
One common approach to image color recognition is to use deep learning models, such as convolutional neural networks (CNNs), which can learn to automatically extract and classify visual features from images. These models are typically trained on large datasets of labeled images, where each image is annotated with the correct color labels. During the training process, the CNNs learn to recognize the visual features that are associated with different colors, such as the hue, saturation, and brightness of each pixel in an image. Once the model is trained, it can be used to automatically identify the colors in new images.
Applications of image color recognition using AI include automatic image tagging, color-based image search, and color analysis in fields such as fashion, art, and design.

Requirements:

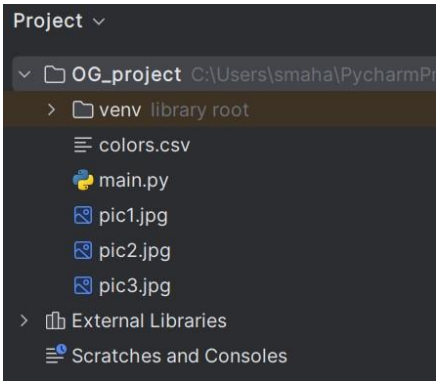
- cv2
- pandas

Code and Explanation:

Our Approach:

This code reads an image and a CSV file containing color names and RGB values, and displays the image in a window. When the user double-clicks on a pixel in the image, the program gets the RGB values of that pixel, finds the closest color match in the CSV file, and displays the color name and RGB values of the selected pixel in a rectangle and text box.

1. Importing necessary libraries - pandas and OpenCVImporting required modules.
2. Setting the path of the image and CSV file to be used.
3. Reading the CSV file into a Pandas Data Frame.
4. Reading the image and resizing it to 800x600.
5. Declaring global variables to be used later in the code.
6. Defining a function get_color_name that takes RGB values and finds the closest matching color name from the Data Frame The directory.
7. Creating a window named "image" and setting the mouse callback function to draw_function.Starting an infinite loop to continuously display the image until the user presses the "Esc" key.
8. Inside the loop, if the user has clicked on a pixel in the image, a rectangle is drawn on the image with the color of the selected pixel, and text is added to the rectangle displaying the color name and RGB values of the selected pixel.
9. Waiting for 20 milliseconds for a key event to occur, and breaking the loop if the "Esc" key is pressed
10. Destroying the window and closing the program. Structure:



The images folder consists of all the required images.

The .csv consists of the data set for colors.

The main.py consists of the main code to run the progRAM.

Installing important modules:

pip install pandas OpenCV-python

```
import cv2 import
pandas as pd
```

Then, the paths for the image and CSV file are set as strings.

```
img_path = 'pic2.jpg'
csv_path = 'colors.csv'
```

The CSV file is read into a Pandas DataFrame using the pd.read_csv() method. The names of the columns in the DataFrame are set using the names parameter, and the header row is ignored by setting header=None.

```
# reading csv file
index = ['color', 'color_name', 'hex', 'R', 'G',
'B']
df = pd.read_csv(csv_path, names=index,
header=None)
```

The image is read using the cv2.imread() method and then resized to 800x600 pixels using the cv2.resize() method.

```
# reading image img =
cv2.imread(img_path)
img = cv2.resize(img, (800,600))
```

Next, global variables are declared to store the values of the selected pixel, the mouse click position, and a flag to indicate if a pixel has been selected.

```
#declaring global variables clicked =
False
r = g = b = xpos = ypos = 0
```

The get_color_name() function takes three arguments representing the RGB values of a pixel, and returns the name of the color that is closest to the given RGB values. The function iterates over all rows in the DataFrame, calculates the Euclidean distance between the RGB values of each row and the given RGB values, and returns the name of the color with the smallest distance.

```
#function to calculate minimum distance from
all colors and get the most matching color def
get_color_name(R,G,B):
    minimum = 1000     for
i in range(len(df)):
        d = abs(R - int(df.loc[i,'R'])) + abs(G -
int(df.loc[i,'G'])) + abs(B -
int(df.loc[i,'B']))         if d <= minimum:
minimum = d
            cname = df.loc[i, 'color_name']

    return cname
```

The draw_function() function is called whenever a mouse event occurs. If the event is a left double-click (cv2.EVENT_LBUTTONDBLCLK), the function sets the global variables b, g, r, xpos, ypos, and clicked to the values of the selected pixel and the mouse click position.

```
#function to get x,y coordinates of mouse
double click def draw_function(event, x, y,
flags, params):     if event ==
cv2.EVENT_LBUTTONDBLCLK:         global b, g, r,
xpos, ypos, clicked         clicked = True
xpos = x         ypos = y         b,g,r =
img[y,x]         b = int(b)         g = int(g)
r = int(r)
```

The window is created using the cv2.namedWindow() method, with the window name set to "image". The cv2.setMouseCallback() method is used to attach the draw_function() function to the window so that it is called whenever a mouse event occurs.

```
# creating window
cv2.namedWindow('image')
cv2.setMouseCallback('image', draw_function)
```

If the user has clicked on the image, the code will then draw a filled rectangle on the image with the color selected by the user, and display the name of the color along with its RGB values using the 'cv2.putText()' function.

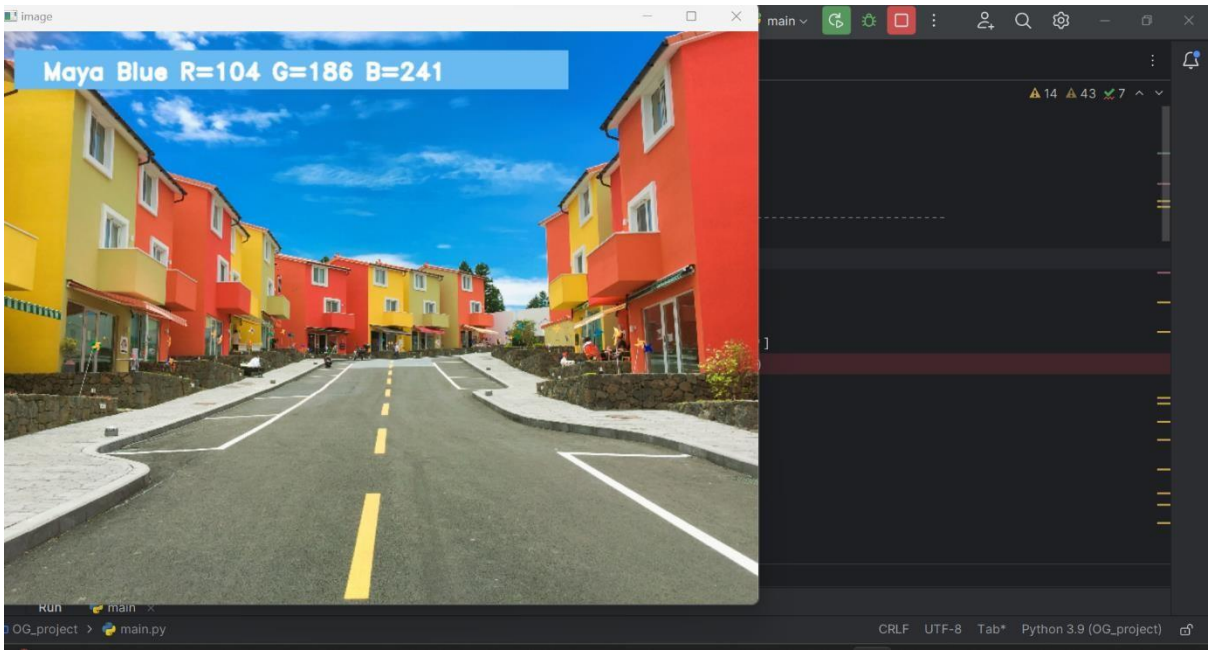
The function 'get_color_name()' is called to retrieve the name of the color based on the RGB values.

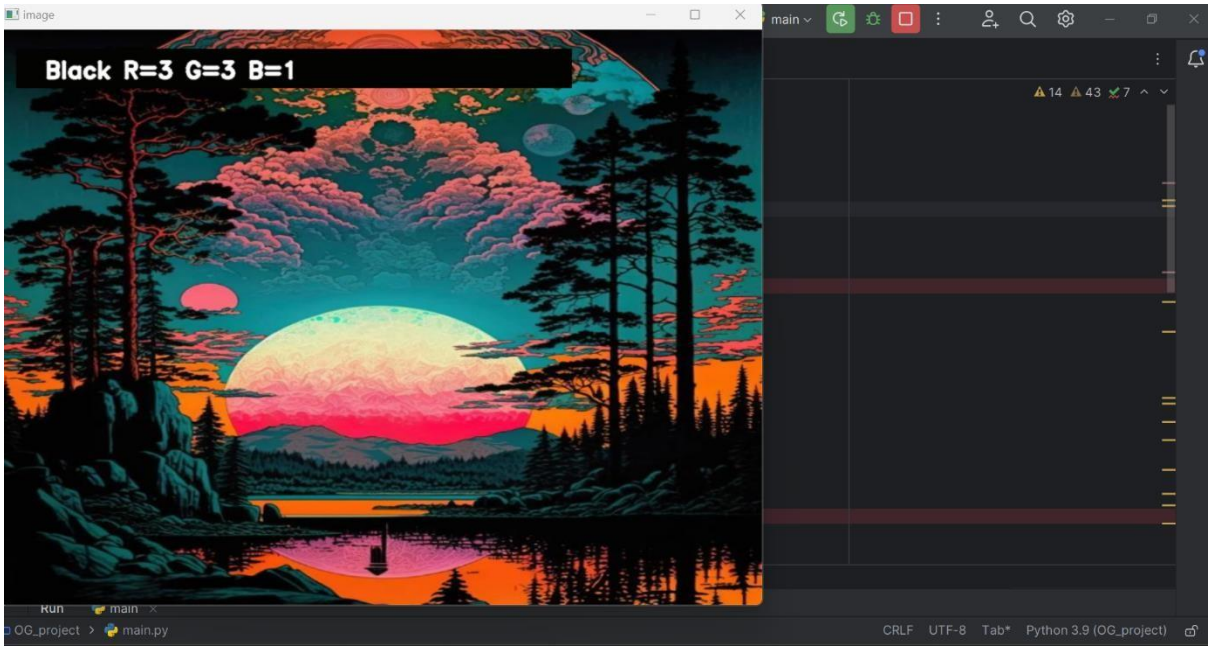
If the sum of the RGB values is greater than or equal to 600, which means the color is very light, the text is displayed in black color to ensure it is visible.

Finally, the 'cv2.waitKey()' function waits for a keyboard event and returns the key code of the pressed key. If the key pressed is the 'ESC' key (which has a key code of 27), the loop is exited, and the 'cv2.destroyAllWindows()' function is called to close the window.

```
while True:
    cv2.imshow('image', img)     if clicked:
       #cv2.rectangle(image, startpoint, endpoint, color,
fills entire rectangle         cv2.rectangle(img, (20,20),
(b,g,r), -1)

       #Creating text string to display( Color name and RG
       text = get_color_name(r,g,b) + ' R=' + str(r) + ' G
' B=' + str(b)
#cv2.putText(img,text,start,font(07),fontScale,color,thick
)         cv2.putText(img, text, (50,50), 2,0.8,
(255,255,255),2,cv2.LINE_AA)

       #For very light colours we will display text in bla
if r+g+b >=600:            cv2.putText(img, text, (50,50),
(0,0,0),2,cv2.LINE_AA)
     if cv2.waitKey(20) & 0xFF == 27:
       break

cv2.destroyAllWindows()
```

Running the Program





Future Scope

1.) Object Detection: We can enhance the project to identify and detect specific objects in an image based on their color. This can be useful in various fields like agriculture, healthcare, and security.

2.) Image Processing: We can explore and implement advanced image processing techniques to improve the accuracy and speed of color detection in images.

3.) Augmented Reality: We can use color detection to create augmented reality experiences, such as overlaying virtual objects on real-world objects of specific colors.

4.) Color-Based Sorting: Color detection can be used to sort different objects based on their color, such as sorting fruits based on their ripeness or sorting different types of waste for recycling.

5.) Color Correction: Color detection can be used to automatically correct the colors of images or videos, which can be useful in fields like photography and videography.

6.) Accessibility: Color detection can be used to create applications and tools that help people with color vision deficiencies to identify colors in their surroundings.