# DAYANANDA SAGAR UNIVERSITY

HAROHALLI, BANGALORE – 562112

**Bachelor of Technology**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**(Artificial Intelligence & Machine Learning)**

**AI Mini Project**

**(22AM2305)**

**Color Detection using OpenCV and Python**

By

**Harshith G R ENG22AM0021**

**Harshith P ENG22AM0022**

**Shivakumar P G ENG22AM0061**

**Sudarshan H Patil ENG22AM0063**

**Under the supervision of**

Prof. PRADEEP KUMAR K Dr. MARY JASMINE Prof. Mitha Guru

**Assistant Professor Assistant Professor Assistant Professor**

**CSE(AI &ML) CSE(AI&ML) CSE(AL&ML)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI&ML),**

#### SCHOOL OF ENGINEERING

#### DAYANANDA SAGAR UNIVERSITY, BANGALORE

**(2023-2024)**



**Day****ananda Sagar University**

**School of Engineering**

**Department of Computer Science & Engineering**

**(Artificial Intelligence & Machine Learning)**

**Harohalli, Bangalore – 562112**  **Karnataka, India**

# CERTIFICATE

This is to certify that the AI Mini Project (22AM2305) work titled **“Color Detection using OpenCV and Python”** is carried out by **Harshith G R** bearing **USN: ENG22AM0021** Bonafede student of Bachelor of Technology in Computer Science and Engineering (AI&ML) at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2023-2024**.

|  |  |  |
| --- | --- | --- |
| Prof. Pradeep Kumar K  Assistant Professor  Dept. of CS&E (AI&ML),  School of Engineering  Dayananda Sagar University  Date: | Dr. Mary Jasmine  Assistant Professor  Dept. of CSE (AI&ML)  School of Engineering  Dayananda Sagar University  Date: | Prof. Mitha Guru  Assistant Professor  Dept. of CSE(AI&ML)  School of Engineering Dayananda Sagar University  Date: |

**ACKNOWLEDGEMENT**

*I extend my heartfelt gratitude to everyone who contributed to the successful completion of the "Interactive Color Explorer" project. This endeavor would not have been possible without the collective support, guidance, and inspiration from various individuals and resources.*

*First, I take this opportunity to express my sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor’s degree in this institution.*

*I would like to extend my gratitude to* ***Dr. Jayvrinda Vrindavanam v, Department Chairman****,* ***Computer Science and Engineering (AI & ML)****,* ***Dayananda Sagar University*** *for providing us the opportunity to come forward with this project.*

*I would like to thank our guide* ***Prof. Pradeep Kumar K,******Dr. Mary Jasmine & Prof. Mitha Guru****,* ***Assistant Professors****,* ***Dept. of Computer Science and Engineering (AI & ML)****,* ***Dayananda Sagar University****, for sparing their valuable time to extend their support towards the project which paved the way for smooth progress and fruitful culmination of the project.*

*I am also grateful to my family and friends who provided me with every requirement throughout the course.*

# TABLE OF CONTENTS

[**ABSTRACT**........................................................................................................................ 5](#_TOC_250008)

[**1. INTRODUCTION**………........................................................................................... 6](#_TOC_250005)

* 1. Background

[1.2](#_TOC_250003) Problem Statement

1.3 Objectives

**2. PROJECT OVERVIEW**………................................................................................... 7

2.1 Task Definition

2.2 Methodology

2.3 Tools & Libraries

[**3. DATASET**..................................................................](#_TOC_250002)................................................... 8

3.1 Dataset Description

**4. IMPLEMENTATION**………………………………………………………………. 9

4.1. Code Overview

4.2. Colour Detection Algorithm

**5. IMAGE PROCESSING** …......................................................................................... 11

5.1. Reading The Image

5.2. Mouse Double-Click Event

**6. COLOUR RETRIEVAL**…………………………………………………………… 13

6.1 Colour Matching Function

**7. USER INTERACTION**……………………………………………………………… 14

7.1. Mouse Event Handling

7.2. Displaying Colour Information

**8. RESULTS AND VISUALIZATION**……………………………………………….. 16

8.1. Displaying Result

8.2. User Interface Enhancement

**9. DATASET INTEGRATION**……………………………………………………… 17

9.1. Utilizing External Dataset

**10. CHALLENGES AND SOLUTIONS**…………………………………………….. 18

10.1. Challenges Faced

10.2. Solutions Implemented

**11. CODE AND CONCLUSION**……………………………………………………… 20

11.1. Code

11.2. Output

11.3. Conclusion

**12. APPLICATIONS AND FUTURE SCOPE**……………………………………….. 23

12.1. Applications

12.2. Future Directives

**13. REFERENCES**……………………………………………………………………… 27

# ABSTRACT

The Color Detection Mini Project leverages the capabilities of OpenCV and Python to create a system for identifying and naming colors in images. The project utilizes a dataset (**'colors.csv'**) that maps RGB values to color names, providing a foundation for accurate color recognition. The report comprehensively outlines the project's objectives, dataset details, code implementation, and the color detection algorithm. Key components, such as image processing steps, user interaction, and results visualization, are explained in detail. Challenges encountered during the project are identified, and effective solutions are presented. The report concludes with a summary of findings, suggestions for future work, and reflections on the acquired skills and knowledge. The Color Detection Mini Project serves as a valuable exploration into computer vision applications and the practical implementation of color recognition algorithms.

# INTRODUCTION

# Background:

# In the realm of computer vision, colour detection plays a pivotal role in image processing, object recognition, and various applications that require an understanding of visual information. Accurate color recognition is essential in scenarios ranging from automated image analysis to robotics and beyond. This project is designed to delve into the significance of colour detection in computer vision applications, acknowledging its importance as a foundational step toward a deeper understanding of visual content.

# As technology advances, the ability to teach machines to perceive and interpret colours becomes increasingly relevant. This project addresses the need for a comprehensive and practical approach to color detection, utilizing widely-used tools such as OpenCV and Python. The integration of a structured dataset ('colors.csv') provides a basis for mapping RGB values to corresponding color names, facilitating a systematic and accurate recognition process. This background sets the stage for an in-depth exploration of the objectives, implementation details, challenges faced, and the broader implications of the Color Detection Mini Project.

# Problem Statement:

# Design and implement a Color Detection AI system for an image processing mini project. The goal is to develop a robust algorithm that can accurately identify and classify colors present in an image. The system should be capable of analyzing images of various formats and resolutions, providing a reliable output regarding the predominant colors and their respective proportions within the image.

# Objectives:

# Implement a colour detection algorithm using OpenCV and Python.

# Provide an overview of the dataset used for colour mapping.

# Explain the code structure and algorithm in detail.

# Showcase the results of colour detection through visualizations.

# Explore challenges faced during implementation and their solutions.

# PROJECT OVERVIEW

# Task Definition:

# The primary task of the provided code is color detection in an image using OpenCV and Python. The project aims to develop a system that, given an image, allows a user to double-click on a region, and the application will identify the color of that region. The identified color is displayed on the image in the form of a filled rectangle with the color's RGB values and name.

# Methodology:

# i. Installation: Python and OpenCV need to be installed. Use pip for Python packages

# ii. Image Loading: OpenCV is utilized to load and read the input image.

# iii. Color Space Conversion: Images are often represented in the RGB color space. Converting images to other color spaces like HSV (Hue, Saturation, Value) can simplify color detection.

# iv. Thresholding: Thresholding techniques are applied to segment the image, making it easier to identify specific colors.

# v. Contour Detection: Contours are utilized to identify and outline regions of interest within the image.

# vi. Color Detection Algorithm: A color detection algorithm is implemented to identify target colors based on user-defined ranges in the chosen color space.

# Tools and libraries:

# i. OpenCV (cv2):

# Purpose: OpenCV is a powerful computer vision library that provides tools and functions for image processing, computer vision, and machine learning.

# Application in the Code: OpenCV is used to read the image, handle mouse events (double-click), draw rectangles, and display images with color information.

# ii. Pandas:

# Purpose: Pandas is a data manipulation library for Python, often used for handling and analysing structured data.

# Application in the Code: Pandas is used to load and manage the color dataset from the 'colors.csv' file, providing an efficient way to work with tabular data.

# iii. Python:

# Purpose: Python is a versatile and widely-used programming language known for its simplicity and readability.

# Application in the Code: Python serves as the primary programming language for implementing the color detection algorithm, user interaction, and overall project development.

# DATA SET

# Dataset Description:

# The 'colors.csv' dataset is an essential component of the color detection project, serving as a structured reference for mapping RGB values to corresponding color names. This dataset, loaded into a Pandas DataFrame within the code, plays a crucial role in enhancing the accuracy and precision of color recognition.

# *Explanation of Columns:*

# color:

# Description: Descriptive names assigned to each color.

# Role in the Code: The 'color' column acts as a human-readable identifier for individual colors.

# color\_name:

# Description: Standardized names assigned to each color.

# Role in the Code: The 'color\_name' column provides a uniform reference for color identification within the color detection algorithm.

# hex:

# Description: Hexadecimal representation of the color.

# Role in the Code: The 'hex' column in the dataset represents colors in a widely used digital format. This representation is employed for various purposes, including web development and digital design.

# R (Red), G (Green), B (Blue):

# Description: Integer values representing the intensity of the respective color channel.

# Role in the Code: The 'R,' 'G,' and 'B' columns play a crucial role in the color detection algorithm. These columns store the RGB values for each color in the dataset. When a user double-clicks on an image, the RGB values are compared with those in the dataset to determine the closest matching color.

# *Significance in the Given Code:*

# Dataset Loading: The dataset is loaded into a Pandas DataFrame ('csv') at the beginning of the code, creating a structured lookup table for color information.

# Color Matching Algorithm: During the color detection process, the RGB values obtained from the image are compared with the RGB values in the 'colors.csv' dataset. The color with the minimum distance is identified and associated with its standardized name ('color\_name').

# User Interaction: The color information retrieved from the dataset is then displayed on the image, providing users with real-time feedback on the color they have selected through a double-click event.

# Consistency and Standardization: The inclusion of standardized color names ('color\_name') ensures consistency in color identification, promoting a uniform user experience.

# In essence, the 'colors.csv' dataset acts as the backbone of the color detection project, supplying a rich and structured collection of color information that facilitates accurate color recognition and user interaction within the implemented code.

# IMPLEMENTATION

# Code Overview:

# The code begins by importing necessary libraries, including OpenCV (cv2) and Pandas (pd). It then reads an image from a specified path using cv2.imread. A set of global variables is declared to store information related to mouse events and color values.

# The 'colors.csv' dataset is loaded into a Pandas DataFrame (csv) with named columns. This dataset serves as a lookup table for color information.

# The color detection process is embedded within a main loop (while True). Inside the loop, the image is continuously displayed using cv2.imshow. When a user double-clicks on the image, the draw\_function is triggered, capturing the RGB values of the clicked position. The color information is then displayed on the image, including a filled rectangle representing the detected color and associated RGB values.

# The loop continues until the user presses the 'esc' key, at which point the program exits.

# *How the Dataset is Loaded and Utilized in Color Detection:*

# The 'colors.csv' dataset is loaded using Pandas (pd.read\_csv), creating a DataFrame (csv) with named columns.

# The loaded dataset serves as a reference for color information during the color detection process.

# The get\_color\_name function takes RGB values as input, iterates through the dataset, and calculates the Manhattan distance to find the closest matching color. The name of the matching color is then returned.

# The color information retrieved from the dataset is displayed on the image in real-time during user interaction.

# Colour Detection Algorithm:

# Reading the Image:

# An image is read from the specified path using cv2.imread.

# Mouse Double-Click Event Handling (draw\_function):

# The cv2.setMouseCallback function is utilized to handle mouse events, particularly the double-click event (cv2.EVENT\_LBUTTONDBLCLK).

# When a double-click occurs, the draw\_function is triggered, capturing the RGB values of the clicked position and updating global variables.

# Color Detection Algorithm (get\_color\_name):

# The get\_color\_name function takes RGB values (from the clicked position) as input.

# It iterates through each row in the 'colors.csv' dataset, calculating the Manhattan distance between the target color and each color in the dataset.

# The color with the minimum distance is identified, and its standardized name (color\_name) is returned.

# Displaying Color Information:

# The detected color information is displayed on the image using OpenCV functions.

# A filled rectangle with the detected color is drawn on the image.

# Text containing the color name and RGB values is added to the image using cv2.putText.

# User Interaction:

# The code continuously displays the updated image in the main loop, providing real-time feedback to the user.

# The loop breaks when the user presses the 'esc' key.

# *Functionality of get\_color\_name and draw\_function:*

# get\_color\_name:

# Takes RGB values as input.

# Iterates through the dataset to find the color with the minimum distance.

# Returns the standardized name (color\_name) of the closest matching color.

# draw\_function:

# Handles the double-click event (cv2.EVENT\_LBUTTONDBLCLK).

# Captures the RGB values of the clicked position and updates global variables (b, g, r, x\_pos, y\_pos, clicked).

# The global variable clicked is used to signal that a double-click event has occurred.

# IMAGE PROCESSING

# Reading the image:

# Images are Read Using OpenCV:

# In the provided code, the cv2.imread function from the OpenCV library is used to read an image. The syntax is as follows:

# img\_path = r"C:\Users\Harshith GR\Downloads\1682127.jpg"

# img = cv2.imread(img\_path)

# img\_path is a string specifying the path to the image file.

# The r prefix before the path indicates a raw string, which is used to handle backslashes in Windows file paths.

# cv2.imread reads the image from the specified path and stores it in the variable img.

# *Path to the Image Used in the Project:*

# In this code snippet, the path to the image used in the project is:

# img\_path = r"C:\Users\Harshith GR\Downloads\1682127.jpg"

# This path points to an image file named "1682127.jpg" located in the specified directory. Users need to replace this path with the actual path to their image file.

# Mouse Double-click event:

# global variableThe double-click event is handled using the cv2.setMouseCallback function, which allows the program to respond to mouse events. Specifically, the code sets up a callback function called draw\_function to handle mouse events.

# cv2.setMouseCallback('image', draw\_function)

# 'image' is the window name where the mouse events are monitored, and draw\_function is the callback function.

# *Capture of Color Information from the Clicked Position:*

# The draw\_function is designed to capture color information when a double-click event occurs. The relevant part of the code is as follows:

# def draw\_function(event, x, y, flags, param):

# if event == cv2.EVENT\_LBUTTONDBLCLK:

# global b, g, r, x\_pos, y\_pos, clicked

# clicked = True

# x\_pos = x

# y\_pos = y

# b, g, r = img[y, x]

# b = int(b)

# g = int(g)

# r = int(r)

# The function is triggered when a double-click event (cv2.EVENT\_LBUTTONDBLCLK) occurs.

# It updates global variables (b, g, r, x\_pos, y\_pos, clicked) with information from the clicked position.

# The RGB values (b, g, r) are obtained from the image at the clicked position (img[y, x]).

# These RGB values are then converted to integers.

# This event handling mechanism captures the color information from the clicked position, allowing the subsequent color detection algorithm to determine the closest matching color based on these RGB values.

# Top of Form

# COLOUR NAME RETRIEVAL

# Colour Matching Function:

# The get\_color\_name function is a crucial component of the color detection algorithm in the provided code. It is responsible for identifying the color with the minimum distance from a given set of RGB values.

# *Below is an in-depth explanation of the function:*

# def get\_color\_name(R, G, B):

# minimum = 2073600

# for i in range(len(csv)):

# d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.loc[i, "B"]))

# if d <= minimum:

# minimum = d

# cname = csv.loc[i, "color\_name"]

# return cname

# Input Parameters (R, G, B): The function takes the RGB values of the color to be matched as input.

# Initialization of minimum: A large initial value (2073600) is set for minimum. This value acts as an upper limit for the minimum distance.

# Iteration through the Dataset:

# The function iterates through each row in the 'colors.csv' dataset using a for loop.

# For each color in the dataset, it calculates the Manhattan distance (d) from the target color represented by the input RGB values (R, G, B).

# Manhattan distance is calculated as the sum of absolute differences between corresponding RGB components.

# Updating Minimum Distance and Color Name:

# If the calculated distance (d) is less than or equal to the current minimum distance (minimum), the minimum distance is updated, and the corresponding color name (color\_name) is stored in the variable cname.

# Return Statement:

# The function returns the color name (cname) associated with the color in the dataset that has the minimum distance from the target color.

# *Calculation of the Minimum Distance from All Colors:*

# The minimum distance calculation involves iterating through each color in the dataset.

# The Manhattan distance is calculated for each color by comparing the target RGB values with the RGB values of the current color in the dataset.

# The color with the minimum distance is determined, and its name is returned.

# This color matching function ensures that the color closest to the target color is identified based on the RGB components. It is a key element in the color detection process, allowing the code to associate a standardized color name with the RGB values obtained from the user's interaction.

# USER INTERACTION

# Mouse Event Handling:

# mouse events are handled using the cv2.setMouseCallback function, which sets a callback function to be called for mouse events. The primary function for handling mouse events is draw\_function.

# cv2.setMouseCallback('image', draw\_function)

# The code specifies that the draw\_function should be called when a mouse event occurs on the window named 'image.'

# *Global Variables and Their Role in Interaction:*

# Several global variables are declared and utilized to facilitate communication between different parts of the code during user interaction. These variables include clicked, r, g, b, x\_pos, and y\_pos.

# clicked:

# Role: It is a Boolean variable that signals whether a mouse event, specifically a double-click, has occurred.

# Utilization: It is used to control the flow of the code, ensuring that color information is processed only when a double-click event occurs.

# r, g, b:

# Role: These variables store the RGB values of the color obtained from the user's double-click.

# Utilization: They are updated in the draw\_function and used for color identification and display.

# x\_pos, y\_pos:

# Role: These variables store the coordinates (position) of the mouse cursor when a double-click event occurs.

# Utilization: They are used to locate the pixel in the image where the double-click occurred, allowing extraction of the RGB values from that position.

# Displaying Colour Information:

# The color information is displayed on the image within the main loop of the code. The relevant portion of the code is as follows:

# if clicked:

# cv2.rectangle(img, (20, 20), (750, 60), (b, g, r), -1)

# text = get\_color\_name(r, g, b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' + str(b)

# cv2.putText(img, text, (50, 50), 2, 0.8, (255, 255, 255), 2, cv2.LINE\_AA)

# if r + g + b >= 600:

# cv2.putText(img, text, (50, 50), 2, 0.8, (0, 0, 0), 2, cv2.LINE\_AA)

# clicked = False

# Drawing a Rectangle:

# A rectangle is drawn on the image using cv2.rectangle. This rectangle serves as a background for displaying color information.

# The position, color, and thickness of the rectangle are determined by the RGB values (b, g, r) obtained from the user's double-click.

# Creating Text String:

# A text string is created, combining the color name, and RGB values.

# Placing Text on the Image:

# The cv2.putText function is used to add the text to the image. It specifies the text, position, font, font scale, color, thickness, and line type.

# The text is displayed in white ((255, 255, 255)) by default, and if the total intensity of the color is high (indicating a light color), the text color is changed to black ((0, 0, 0)) for better visibility.

# Resetting Clicked State:

# The clicked variable is reset to False to ensure that color information is processed only when a new double-click event occurs.

# This mechanism allows users to interactively explore the image, obtaining color information by double-clicking on different regions, with the color details displayed on the image in real-time.

# RESULTS AND VISUALIZATION

# Displaying Results:

# The color detection results are displayed in real-time as the user interacts with the image through double-click events. Here is the relevant code snippet:

# while True:

# cv2.imshow("image", img)

# if clicked:

# cv2.rectangle(img, (20, 20), (750, 60), (b, g, r), -1)

# text = get\_color\_name(r, g, b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' + str(b)

# cv2.putText(img, text, (50, 50), 2, 0.8, (255, 255, 255), 2, cv2.LINE\_AA)

# if r + g + b >= 600:

# cv2.putText(img, text, (50, 50), 2, 0.8, (0, 0, 0), 2, cv2.LINE\_AA)

# clicked = False

# if cv2.waitKey(20) & 0xFF == 27:

# break

# The code uses cv2.imshow to continuously display the image with color information in real-time.

# When a user double-clicks, a rectangle with the detected color is drawn, and text containing the color name and RGB values is displayed on the image.

# For better visibility, the text is displayed in white by default, and if the color is very light, it is displayed in black.

# User Interface Enhancement:

# Interactive Color Legend:

# Integrate an interactive color legend that dynamically updates as the user explores different regions of the image. This legend can provide a visual representation of detected colors along with their names.

# User Feedback Animation:

# Add animations or transitions for a smoother user experience when displaying color information. This can include fading in/out or highlighting the detected color region.

# Customizable Display Parameters:

# Allow users to customize the appearance of the displayed color information, such as the position, size, and color of the text and rectangle.

# Additional Color Information:

# Expand the displayed information to include more details about the detected color, such as the hexadecimal representation or additional color characteristics.

# Real-Time Color Sampling:

# Implement a feature that allows users to sample colors in real-time by hovering over different regions of the image without the need for a double-click.

# Error Handling:

# Implement error handling to gracefully handle cases where the user double-clicks outside the image boundaries or in areas where RGB values cannot be obtained.

# Keyboard Shortcuts:

# Provide keyboard shortcuts for common actions, such as clearing the color information or saving the current state of the image.

# By incorporating these enhancements, the user interface can become more intuitive, visually appealing, and user-friendly, providing a more enjoyable experience for users exploring color detection in images.

# DATASET INTEGRATION

# Utilizing External Dataset:

# the color dataset is loaded from an external CSV file named 'colors.csv.' Here is the relevant code snippet:

# index = ["color", "color\_name", "hex", "R", "G", "B"]

# csv = pd.read\_csv('colors.csv', names=index, header=None)

# The pd.read\_csv function from the Pandas library is used to read the external CSV file into a Pandas DataFrame (csv).

# The names parameter is used to provide column names, and header=None indicates that the CSV file does not have a header row.

# The external dataset is assumed to be structured with columns named "color," "color\_name," "hex," "R," "G," and "B," which aligns with the structure used in the code.

# *Possible Extensions for Using a More Extensive Dataset:*

# Dynamic Dataset Loading:

# Implement a more dynamic approach to dataset loading, allowing users to specify the file path or URL of the external dataset. This would enhance flexibility and allow the code to work with different datasets.

# Dataset Validation:

# Include robust validation mechanisms to ensure that the loaded dataset adheres to the expected structure (columns: "color," "color\_name," "hex," "R," "G," "B"). This can prevent errors due to incompatible datasets.

# Multiple Dataset Integration:

# Allow users to seamlessly integrate multiple external datasets, merging them into a comprehensive dataset. This could be useful for expanding the range of colors available for detection.

# Online Dataset Retrieval:

# Integrate functionality to fetch color information from online sources or APIs dynamically. This would enable the use of up-to-date and extensive color datasets without the need for local files.

# Color Space Conversion:

# Extend the dataset to include color information in different color spaces (e.g., HSL, HSV) to enhance the color detection algorithm's capabilities.

# By incorporating these extensions, the code can be adapted to work with diverse and extensive color datasets, offering users more flexibility and the ability to customize the color detection experience based on their specific needs.

# CHALLENGES AND SOLUTIONS

# Challenges Faced:

# i. Handling User Interactions:

# The code relies on mouse events, particularly double-clicks, to capture color information. Managing user interactions and ensuring a smooth and intuitive experience can be challenging.

# ii. Accurate Color Matching:

# Achieving accurate color matching based on RGB values from user interactions and comparing them with the dataset can be challenging, especially when dealing with variations in lighting conditions.

# Solutions Implemented:

# Handling User Interactions:

# Code Modularity: The use of modular functions, such as draw\_function and get\_color\_name, enhances code readability and maintainability. It also simplifies the handling of user interactions, making it easier to track and manage.

# Global Variables: The use of global variables (clicked, r, g, b, x\_pos, y\_pos) allows for effective communication between the mouse event handling function (draw\_function) and the main loop, streamlining the interaction process.

# Real-Time Feedback: Continuous real-time feedback to users through the display of color information ensures an interactive and responsive experience. The code captures and processes user interactions immediately, updating the image with relevant color details.

# Accurate Color Matching:

# Distance Metric Selection: The code employs a simple yet effective Manhattan distance metric for color matching. This metric is computationally efficient and suitable for the project's requirements. It minimizes the computational load while still providing reasonable accuracy in color matching.

# Dynamic Dataset Loading: The ability to load an external color dataset allows users to customize and extend the dataset as needed. This flexibility accommodates variations in color representation and enables the code to adapt to different color datasets, improving the accuracy of color matching.

# User Interface Enhancement: The suggestions for user interface improvements, such as an interactive color legend and real-time color sampling, contribute to addressing challenges related to user interactions. These enhancements can make the color detection process more intuitive and user-friendly.

# CODE AND CONCLUSION

# Code:

# 

# 

# Output:

# 

# 

# 

# 

* 1. **Conclusion:**

In conclusion, the provided code successfully demonstrates a simple yet effective color detection system using OpenCV and Python. The project achieves the following key outcomes:

1. **User Interaction:** The code allows users to interactively explore an image, capturing color information through double-click events. Real-time feedback is provided, with rectangles and text displaying the detected color and its RGB values.
2. **Color Matching Algorithm:** The color matching algorithm, based on the Manhattan distance metric, effectively identifies the closest matching color from an external dataset. This dataset, loaded from 'colors.csv,' serves as a reference for color information.
3. **Modularity and Readability:** The code is structured in a modular manner, enhancing readability and maintainability. Functions like draw\_function and get\_color\_name contribute to a clear and organized codebase.
4. **Flexibility with External Datasets:** The code accommodates the use of external color datasets, offering flexibility for users to customize and extend the dataset based on their specific needs.
5. **User Interface Enhancement:** Suggestions for user interface improvements, such as an interactive color legend and real-time color sampling, provide avenues for enhancing the overall user experience.

# APPLICATIONS AND FUTURE SCOPE

* 1. **Applications:**

**1.Image Editing Software:**

* **Usage:** The color detection system can be integrated into image editing software to assist users in identifying and selecting specific colors within an image. This can streamline the editing process, allowing users to make targeted color adjustments.

**2. Web Design and Branding:**

* **Usage:** Web designers and branding professionals can utilize the color detection system to identify and match colors for website elements or brand assets. This ensures consistency in color representation across digital platforms.

**3. Fashion and Apparel Industry:**

* **Usage:** In the fashion industry, the color detection system can be employed for analysing and categorizing colors in clothing and accessories. This can aid designers in creating cohesive color palettes for their collections.

**4. Digital Art and Graphics:**

* **Usage:** Digital artists and graphic designers can benefit from the color detection tool to analyse and replicate colors from source images. This can be particularly useful for maintaining color harmony in digital illustrations and designs.

**5. Quality Control in Manufacturing:**

* **Usage:** In manufacturing processes, especially those involving color-sensitive products (e.g., textiles, paints, cosmetics), the color detection system can assist in quality control by ensuring consistency in color output.

**6. Educational Tools for Color Theory:**

* **Usage:** The project can be extended into an educational tool for teaching color theory. It can help students understand concepts such as RGB color representation, color matching algorithms, and the relationship between color names and RGB values.

**7. Accessibility in User Interfaces:**

* **Usage:** The color detection system can contribute to enhancing accessibility in user interfaces by identifying and ensuring sufficient contrast between foreground and background colors. This is crucial for creating user-friendly interfaces for individuals with visual impairments.

**8. Product Packaging and Labelling:**

* **Usage:** In product packaging, the color detection tool can aid in ensuring consistency in color printing and packaging. It can be particularly important in industries where accurate color representation on labels is critical.

**9. Healthcare Imaging:**

* **Usage:** In medical imaging, the color detection system can assist in identifying specific regions or features in images, contributing to diagnostics and analysis. For example, it can be applied in pathology slides or medical imaging annotations.

**10. Artificial Intelligence and Robotics:**

* **Usage:** The color detection system can be integrated into robotic systems for color recognition tasks. This is applicable in scenarios where robots need to interact with or manipulate objects based on color cues.

**11. Digital Marketing and social media:**

* **Usage:** Digital marketers can leverage the color detection system for analysing and selecting colors for marketing materials and social media content. Consistent color representation enhances brand recognition.

**12. Environmental Monitoring:**

* **Usage:** In environmental monitoring, the color detection tool can be used to analyse images captured by sensors or cameras, assisting in the identification of specific environmental conditions or anomalies.

The versatility of the color detection system opens up numerous possibilities for its application across different industries and fields, providing valuable insights and tools for professionals and enthusiasts alike.

* 1. **Future Directives:**

***Project extensions:***

1. **Advanced Color Detection Algorithms:**

# Implement more advanced color detection algorithms, such as machine learning-based approaches or techniques involving color segmentation. This could enhance the accuracy and robustness of color detection, especially in complex images.

# Exploration of Additional Color Spaces:

# Extend the project to explore color detection in different color spaces (e.g., HSL, HSV). Understanding and utilizing various color spaces can provide more insights into color representation and improve the versatility of the color detection system.

# Real-Time Color Tracking:

# Integrate real-time color tracking capabilities, allowing users to track and analyse color changes in a live video stream. This extension would be valuable for applications involving dynamic and changing color scenarios.

# Object Recognition and Color Labelling:

# Combine color detection with object recognition to not only identify colors but also label objects based on their detected color. This expansion could be beneficial for applications involving color-based object categorization.

# *User Interface Enhancement:*

# Interactive Color Legend:

# Implement an interactive color legend that dynamically updates as users explore the image. The legend can provide a visual representation of detected colors, making it easier for users to interpret and understand the color information.

# Real-Time Color Sampling:

# Allow users to sample colors in real-time by hovering over different regions of the image without the need for a double-click. This enhancement would provide a more fluid and natural interaction experience.

# Customizable Display Parameters:

# Introduce user-configurable options for the display of color information. Users could customize the position, size, and appearance of the color information overlay to suit their preferences.

# *Integration of External Datasets:*

# i. Interactive Color Legend:

# Develop functionality to fetch color information from online sources or APIs dynamically. This would enable the use of up-to-date and extensive color datasets without the need for local files.

# ii. Multiple Dataset Integration:

# Extend the capability to seamlessly integrate multiple external color datasets. Users could merge datasets, providing a comprehensive reference for color detection across a wide range of scenarios.

# *Optimization and Performance:*

# i. Parallel Processing:

# Explore the possibility of optimizing the code for parallel processing, especially during color matching calculations. This optimization could lead to improved performance, making the color detection system more efficient.

# ii. GPU Acceleration:

# Investigate the feasibility of leveraging GPU acceleration for intensive image processing tasks. This could significantly speed up the color detection process, especially in scenarios involving high-resolution images.

# *Educational and Outreach Features:*

# i. Educational Mode:

# Develop an educational mode that provides explanations and insights into the color detection process. This mode could include tooltips, visual cues, and explanations to help users understand the underlying concepts.

# ii. Integration with Educational Platforms:

# Explore the integration of the color detection project with educational platforms, making it accessible to students and learners interested in computer vision and image processing.

Continued development and exploration in these areas could elevate the color detection project to new heights, offering a more comprehensive and feature-rich tool for users interested in understanding and utilizing color information in images.

# REFERENCES

* "Digital Image Processing" by Gonzalez and Woods.

<https://studylib.net/doc/25705174/digital-image-processing-4th-ed.---r.-gonzalez--r.-woods->...

* "Color Image Processing: Methods and Applications" by Rastislav Lukac

<https://www.taylorfrancis.com/books/edit/10.1201/9781315221526/color-image-processing-anastasios-venetsanopoulos-joost-van-de-weijer-yannis-avrithis-gerrit-polder-ioannis-kompatslaris-costas-cotsaces-nikos-nikolaidis-pamela-vercellone-smith-liron-yatziv-rastislav-lukac-konstantinos-plataniotis-philip-laplante-philip-laplante-ioannis-pitas-bogdan-smolka-alberto-del-bimbo-strintzis-matti-kalevi-pietikainen-stefano-berretti-hiroaki-kotera-ryoichi-saito-guillermo-sapiro-henryk-palus-alessia-de-rosa-lisimachos-paul-kondi-alireza-osareh-abdenour-hadid-yoshito-abe-matthias-carlsohn-theo-gevers-dan-witzner-hansen-andreas-kercek-choon-woo-kim-raimund-leitner-sanjit-mitra-vishal-monga-jayanta-mukherjee-abhay-sharma-yu-hoon-kim-stefano-piva-alessandro-piva-yiannis-kompatsiaris-savvas-argyropoulos-nikolaos-boulgouris-vito-cappellini-zuzana-cernekova-niranjan-damera-venkata-stamatia-dasiopoulou-brian-evans-graham-finlayson-fred-hamprecht-hu-michael-kelm-bjoern-menze-carlo-regazzoni-hwa-seok-seong-marcello-spirito-evaggelos-spyrou-harro-stokman-nikolaos-thomos-janna-birgitta-martinkauppi>

* "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein.

<https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf>

* Wikipedia

<https://en.wikipedia.org/wiki/Lists_of_colors>

<https://data-flair.training/blogs/project-in-python-colour-detection/>