VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JnanaSangama, Belagavi– 590018

Mini Project Report

On

"COLOR DETECTION SYSTEM"

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

By

DHANUSH A	**4MT21CS046**
DHANUSH S SHETTY	**4MT21CS047**
KARTHIK U SHETTIGAR	**4MT21CS064**

Under the Guidance of

Ms. Sunitha N V
Assistant Professor

Ms. Amrutha
Assistant Professor

MITE
Invent Solutions

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MANGALORE INSTITUTE OF TECHNOLOGY &ENGINEERING

Badaga Mijar, Moodabidri-574225, Karnataka

2023-2024

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(A Unit of Rajalaxmi Education Trust ®, Mangalore)

Autonomous Institute Affiliated to V.T.U, Belagavi , Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

MITE

Invent Solutions

DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

(Accredited by NBA)

CERTIFICATE

This is to certify that **Mr. Dhanush A(4MT21CS046), Mr. Dhanush S Shetty (4MT21CS047), Mr. Karthik U Shettigar(4MT21CS064)** has satisfactorily completed the mini project entitled **"Color Detection System"** for the **Computer Graphics and Image Processing Laboratory (21CSL66**) as prescribed by the VTU for 6th semester B.E. Computer Science & Engineering branch for the academic year 2023 – 2024. It is certified that all the corrections or suggestions indicated for internal assessment have been incorporated in the report and is been verified and validated.

.....	.....	.....
Ms. Sunitha N V	**Ms. Amrutha**	**Dr. Ravinarayana B**
Project Guide	**Project Guide**	**Head of the Department**

Name of Examiners	**Signature of the Examiners**
1.	**.....**
2.	**.....**

ABSTRACT

This project presents an efficient system for color detection and tracking using both RGB and HSV color spaces, implemented with OpenCV and Python. The system accurately identifies and labels colors in images and video streams, displaying this information with bounding boxes and text labels. By leveraging the strengths of both RGB and HSV models, the system achieves robust and versatile color detection, applicable in fields such as autonomous driving, industrial automation, and image editing. The project includes a user-interactive feature where double-clicking on any pixel provides the corresponding color name and RGB values. Additionally, the system tracks red, blue, and green objects in real-time, enhancing its utility in dynamic environments. The use of OpenCV ensures efficient processing, meeting the performance demands of modern applications. Future enhancements include expanding the color database, integrating machine learning for improved accuracy, and optimizing the system for deployment on various platforms.

ACKNOWLEDGMENT

The successful completion of any significant task is the outcome of invaluable aggregate combination of different people in radial direction explicitly and implicitly. We would therefore take opportunity to thank and express our gratitude to all those without whom the completion of project would not be possible.

We express our thanks to **Ms. Sunitha N V, Assistant Professor and Ms. Amrutha, Assistant Professor,** Department of Computer Science & Engineering for having provided all the facilities that helped us in timely completion of this mini project.

We express our sincere gratitude to **Dr. Ravinarayana B, Associate Professor and Head,** Department of Computer Science & Engineering for his support and guidance.

We would like to thank **Dr. Prashanth C M, Principal**, Mangalore Institute of Technology and Engineering, Moodabidri for his support and encouragement.

We express our sincere gratitude to our institution and management for providing us with good infrastructure, laboratory facilities, qualified and inspiring staffs, and whose guidance was of immense help in completion of this mini project successfully.

DHANUSH A **4MT21CS046**
DHANUSH S SHETTY **4MT21CS047**
KARTHIK U SHETTIGAR **4MT21CS064**

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The project focuses on the crucial role of color detection in various fields such as robotics, image editing, and automation. It aims to implement a system for identifying and tracking colors in images and videos, leveraging the RGB model to label colors and the CSV model for object detection and tracking. This approach is valuable in applications including self-driving cars for traffic signal detection, industrial robots for sorting objects, and even medical fields for skin tone and license plate recognition. By utilizing OpenCV and Python, the system addresses the need for robust and economical computer vision solutions in a variety of domains, including defense, security, and navigation.

1.1 Overview

This project centers on developing a color detection system to enhance visual recognition in various applications. The primary goal is to identify and track colors in images and videos using Python and OpenCV. The system employs the RGB color model to determine and display color names by analyzing pixel values, while the HSV color model is used for accurate object detection and tracking. Key applications include self-driving cars for traffic signal recognition, industrial robots for sorting-colored objects, and medical fields for skin tone and license plate detection. The project addresses the need for efficient and reliable color detection solutions in diverse fields such as automation, security, and navigation, providing a robust tool for both real-time and image-based analysis.

1.2 Problem Statement

The color detection to identify and track colors in images and videos. It uses the RGB model to name colors based on pixel values and the HSV model for precise object tracking. This system has diverse applications, including traffic signal detection in self-driving cars, sorting-colored objects in industrial robots, and detecting skin tones and license plates in medical and security contexts. By providing an efficient and reliable tool for color-based analysis, the project addresses needs across automation, security, and navigation.

1.3 Objectives

- Develop a color detection system using Python and OpenCV to identify and track colors in images and videos.
- Utilize RGB and HSV models for accurate color naming and object tracking based on pixel values and color properties.
- Apply the system to practical scenarios such as autonomous vehicle traffic signal detection, industrial object sorting, and medical and security color recognition.

CHAPTER 2

LITERATURE SERVEY

L. Feng, L. Xiaoyu and C. Yi,[1] presented "An efficient detection method for rare colored capsule based on RGB and HSV color space," that proposed an efficient detection method for rare-colored capsules using both RGB and HSV color spaces. Their approach is particularly focused on improving the detection of unusual colors which might be challenging to identify using a single color space. By combining RGB and HSV, the method enhances the ability to detect and differentiate rare colors in complex environments, such as medical imaging or quality control in manufacturing. This dual approach takes advantage of the strengths of each color space—RGB for its direct representation of colors in digital images and HSV for its ability to separate color information from intensity. The study contributes to the field by demonstrating how integrating multiple color spaces can improve detection accuracy and robustness in scenarios involving unusual or rare colors.

G. Pavithra, J. J. Jose and T. A. Chandrappa[2], addressed the challenge of real-time color classification in video streams. Their research focuses on the dynamic nature of video data, where objects and colors constantly change, requiring a system capable of rapid and accurate classification. The method they developed emphasizes real-time processing, which is crucial for applications such as surveillance, autonomous vehicles, and interactive systems where timely and accurate color information is essential. Their work is significant as it provides a solution for classifying colors on-the-fly, allowing systems to react immediately to changes in the visual environment, thereby enhancing the functionality of real-time systems in various practical contexts.

Vishesh Goel, Sahil Singhal, Silica Kole[3] conducted a comparative study on the CPU time consumption for image processing algorithms implemented in Matlab versus OpenCV. Their research highlights the performance differences between these two widely used platforms. The study found that OpenCV offers significant advantages in terms of processing speed and computational efficiency compared to Matlab. This comparison is crucial for applications that require real-time processing and high performance, as it demonstrates the practical benefits of using OpenCV for image processing tasks. The findings of this study underscore the importance of selecting the right tools for efficient image processing, which is essential for developing applications that need to handle large volumes of data or require rapid computation.

The project focuses on the crucial role of color detection in various fields such as robotics, image editing, and automation. It aims to implement a system for identifying and tracking colors in images and videos, leveraging the RGB model to label colors and the CSV model for object detection and tracking.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

For running the Image Filter Application smoothly, the following hardware specifications are recommended:

- Processor : Intel Core i5
- RAM : 8GB of RAM
- Storage : 500MB
- Graphics Card : AMD
- Hard Disk : 500 GB or above

3.2 Software Requirements

To set up and run the Image Filter Application, the following software components are needed:

- Operating System : Windows 11
- Language : Python
- IDE/Code Editor : Visual Studio Code
- Library : OpenCV,Pandas

CHAPTER 4

METHODOLOGY

The methodology for color detection involves several key steps and techniques:

1. **Image Loading and Display:**
    - Load an RGB image using OpenCV (cv2.imread). Display the image in a window (cv2.imshow) where users can interact with it.
2. **Color Data Management:**
    - Import a CSV file (colors.csv) containing color names and their corresponding RGB values using pandas (pd.read_csv). This dataset serves as a reference for color matching.
3. **User Interaction**:
    - Implement a function (draw_function) that captures mouse events, specifically double-clicks (cv2.EVENT_LBUTTONDBLCLK). Extract RGB values (img[y, x]) from the clicked pixel location.
4. **Color Matching Algorithm:**
    - Develop a function (getColorName) that iterates through the CSV dataset to compute the Euclidean distance between the clicked RGB values and each color's RGB values. Determine the color name with the closest match based on this distance metric.
5. **Visualization and Feedback:**
    - Upon double-clicking, update the image display to include a filled rectangle (cv2.rectangle) in the detected color. Overlay text (cv2.putText) showing the detected color name and its RGB values. Adjust text color based on background brightness for readability.
6. **User Interaction Loop:**
    - Continuously update the image display based on user interactions until terminated (cv2.waitKey with escape key check)

    .
7. **Integration with Visual Studio Code (VS Code):**

- Use VS Code as the integrated development environment (IDE) for writing, debugging, and running Python scripts. Leverage VS Code's built-in terminal for command execution and debugging tools for troubleshooting.

This methodology ensures robust color detection through interactive user input, leveraging computational techniques to match RGB values against a predefined dataset. Integration with VS Code enhances development efficiency and provides a seamless environment for coding and testing the color detection application.

CHAPTER 5

IMPLEMENTATION

The project begins with acquiring and preprocessing images or video streams to enhance their quality. It then converts these images into both RGB and HSV color spaces. The RGB model is used to identify and label colors based on pixel values, while the HSV model facilitates robust tracking of objects under diverse lighting conditions. Detected colors are highlighted with rectangular bounding boxes, and their names are displayed for easy identification. In the case of video streams, the system continuously processes each frame in real-time to track moving objects accurately. Performance is optimized through OpenCV, ensuring efficient handling of high-resolution images and videos with minimal processing delay.

```
ap = argparse.ArgumentParser()
ap.add_argument('--image', required=True,
help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']
```

Fig 5.1: Code Snippet for Argument Parsing

The above figure shows functions to parses the command line argument to get the image path.

```
ap = argparse.ArgumentParser()
ap.add_argument('--image', required=True,
help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']
```

Fig 5.2: Image and Color Data Loading

The figure 5.2 indicates Reading the image and CSV file containing color data.

```
def getColorName(R, G, B):
    minimum = float('inf')
    cname = ""
    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
```

Fig 5.3: Color Matching Function

The above code snippet finds the closest color name based on RGB values by calculating the distance from each color in the CSV.

```
def draw_function(event,x,y,flags, param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b, g, r, clicked
        clicked = True
        b, g, r = img[y, x]
        b, g, r = int(b), int(g), int(r)
```

Fig 5.4: Mouse Callback Function

The above figure shows functions to captures the RGB values of the pixel where the user double-clicks.

```
while True:
    cv2.imshow("image", img)
    if clicked:
        cv2.rectangle(img, (20, 20), (750,
60), (b, g, r), -1)
        text = f"{getColorName(r, g, b)}
R={r} G={g} B={b}"
        color = (0, 0, 0) if r + g + b >=
600 else (255, 255, 255)
        cv2.putText(img, text, (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2,
cv2.LINE_AA)
        clicked = False
    if cv2.waitKey(20) & 0xFF == 27:
        break
```

Fig 5.5: Main Loop

The above code snippet displays the image, shows color information and bounding box when a pixel is clicked, and exits the loop on 'Esc' key press.

CHAPTER 6

RESULTS AND SNAPSHOTS

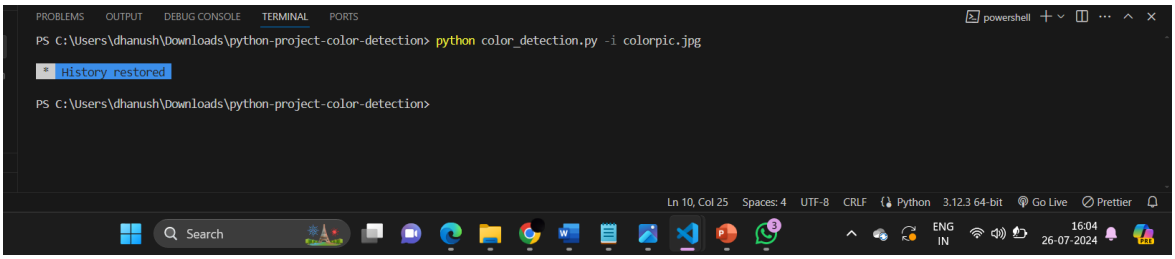

Fig 6.1 Compiling in terminal

The above figure ensure the colorpic.jpg file is in the same directory as the script or provide the correct path to the image file.

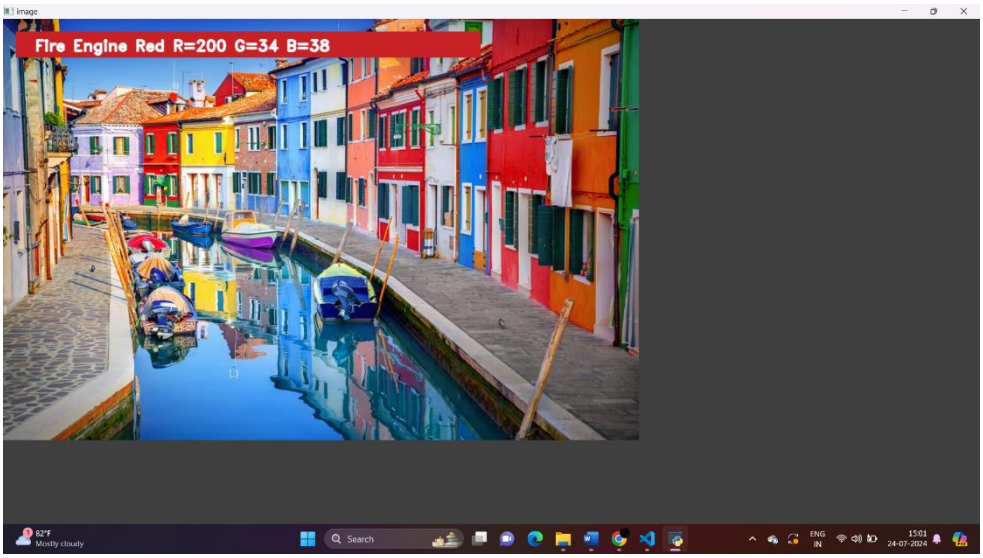

Fig 6.2 Detection of color from the image

The above figure 6.2 shows when you double-click on the image, it draws a rectangle and displays the name of the color and its RGB values.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this project, we developed an efficient system for color detection and tracking using both RGB and HSV color spaces. By leveraging OpenCV and Python, the application accurately identifies and labels colors in images and video streams, displaying this information with bounding boxes and text labels. The integration of RGB and HSV models enhances the system's robustness and versatility, making it applicable in various fields such as autonomous driving, industrial automation, and image editing. The use of OpenCV ensures efficient real-time processing, meeting the performance needs of modern applications.

7.2 Future Work

Future enhancements of this project could include expanding the color database to include more colors and shades for even more precise detection, and implementing machine learning algorithms to improve color detection accuracy and adaptability in different lighting conditions. Additionally, extending the system to track multiple colors simultaneously would provide more comprehensive object tracking in complex environments. Optimizing the system for deployment on various platforms, such as mobile devices and embedded systems, can increase its applicability in diverse scenarios. Finally, incorporating 3D object tracking capabilities would enhance the system's effectiveness in dynamic and three-dimensional spaces, such as robotics and augmented reality applications.

REFERENCES

[1] L. Feng, L. Xiaoyu and C. Yi, "An efficient detection method for rare colored capsule based on RGB and HSV color space," 2014 IEEE International Conference on Granular Computing (GrC), 2014, pp. 175-178, doi: 10.1109/GRC.2014.6982830.

[2] G. Pavithra, J. J. Jose and T. A. Chandrappa, "Real-time color classification of objects from video streams," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017, pp. 1683-1686, doi: 10.1109/RTEICT.2017.8256886.

[3] Vishesh Goel, Sahil Singhal, Silica Kole "Specific Color Detection in Images using RGB Modelling in MATLAB" International Journal of Computer Applications (0975 – 8887) Volume 161 – No 8, March 2017,