COLOUR DETECTION

Project Report

Submitted by

VASANTH A S(RA2211026040017)
SARVESSH P(RA2211026040019)

21CSE251T – DIGITAL IMAGE PROCESSING

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be **University** *u/s 3 of UGC Act, 1956*

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

SRM INSTITUTE OF SCIENCE

AND TECHNOLOGY

VADAPALANI CAMPUS APRIL

2024

ABSTRACT

Color detection is a critical aspect of image processing and computer vision applications. This abstract provides an overview of a Python project focused on color detection, which aims to develop a user-friendly and versatile tool for color analysis in images and videos.

The project begins by implementing traditional color space models such as RGB, HSV, and Lab to accurately identify and segment colors in images. It leverages the OpenCV library for image processing tasks and Python's ease of use for efficient color detection. Users can specify a target color or range of colors and the tool will provide detailed information about their presence in the input media, such as the number of pixels, areas, and positions.

To enhance usability and versatility, the project extends its functionality by incorporating machine learning techniques. Users can train custom color detectors

by providing labelled data, allowing the tool to recognize specific colors or objects of interes

TABLE OF CONTENTS

ABSTRACT 2

1. INTRODUCTION 4

1. PROJECT AIMS AND OBJECTIVES 4
2. OPERATION ENVIRONMENT 5

2. LITERATURE SURVEY 6

1. BACKGROUND OF PROJECT 6
2. SOFTWARE REQUIREMENT SPECIFICATION 7

3. PROPOSED METHODOLOGY 10

1. EXISTING VS PROPOSED 10
2. ALGORITHM USED 10
3. WORKING PRINCIPLE 13

4. RESULTS AND DISCUSSION 14

1. TESTING 14
2. RESULT VALIDATION 15

5. CONCLUSION AND FUTURE WORK 17

1. CONCLUSION 17
2. FUTURE SCOPE 17

6. REFERENCES 18

CHAPTER 1
INTRODUCTION

1.1 PROJECT AIMS AND OBJECTIVES

In the realm of computer vision and image processing, the ability to detect and manipulate colors plays a pivotal role in a myriad of applications, from industrial automation to creative design and beyond. Color detection, the process of identifying and extracting specific colors within images and videos, serves as the foundation for a wide array of tasks, ranging from object tracking and sorting to image enhancement and artistic expression. In this context, our Python project aims to provide a practical and versatile solution for color detection, offering users the tools they need to explore the vibrant world of colors and their applications.

Colors are a fundamental element of our visual world. They convey information, aesthetics, and meaning. In industries, the precise identification and differentiation of colors are vital for quality control and consistency. In fields such as agriculture, color detection is employed for crop monitoring, disease identification, and yield estimation. In healthcare, color analysis assists in the diagnosis of medical conditions from X-rays and scans. In everyday life, it is utilized for facial recognition, augmented reality, and object tracking in autonomous vehicles. The demand for accurate, efficient, and user-friendly color detection tools is ever-expanding.

1.2 OPERATION ENVIRONMENT

PROCESSOR	INTEL CORE PROCESSOR FOR BETTER PERFORMANCE
OPERATING SYSTEM	WINDOWS 11
MEMORY	4GB RAM
HARD DISK SPACE	MINIMUM 2GB
IDE	PYCHARM

CHAPTER 2
LITERATURE SURVEY

1. BACKGROUND OF PROJECT

The background of your project involves the context and rationale behind developing a Color Detection system. This may include:

- Computer Vision: This field deals with enabling computers to gain high-level understanding from digital images or videos. In color detection, computer vision algorithms are used to process and analyze color data.
- Color Models: Color models like RGB (Red, Green, Blue), HSV (Hue, Saturation, Value), and Lab are commonly used in color detection projects. These models represent colors in different ways, and the choice of model depends on the specific requirements of the project.
- Image Processing Techniques: Techniques such as thresholding, segmentation, and filtering are employed to preprocess images and extract relevant color information. Thresholding, for instance, can be used to separate

objects of interest based on their color characteristics.

- Machine Learning: In some advanced color detection projects, machine learning models such as neural networks may be trained to recognize and classify colors. This involves feeding labeled color data to the model so it can learn to distinguish between different colors.
- Applications: Color detection finds applications in various fields such as industrial automation (e.g., sorting objects based on color), image editing software (e.g., automatic color correction), and robotics (e.g., identifying colored objects in a robot's environment).

2.2 SOFTWARE REQUIREMENT SPECIFICATION

PYCHARM – PyCharm is an integrated development environment (IDE) for the Python programming language. Developed by JetBrains, it provides a comprehensive set of tools for Python developers to write, test, and debug Python code more efficiently. PyCharm offers features such as code analysis, intelligent code completion, a visual debugger, and integrated testing, making it a popular choice for both beginners and experienced Python programmers. It also supports web development frameworks like Django and Flask, and it comes in two editions: PyCharm Community (free) and PyCharm Professional (commercial), each tailored to different use cases.

OPENCV-

OpenCV, short for Open-Source Computer Vision Library, is an open-source computer vision and machine learning software library. OpenCV provides a wide range of tools and functions for tasks related to computer vision, image processing, and machine learning. It is written in C++ and supports multiple programming languages, including Python, making it a popular choice for developers and researchers in the fields of computer vision, robotics, artificial intelligence, and image analysis.

OpenCV offers functionalities for various tasks, such as image and video processing, object detection and recognition, facial recognition, feature extraction, and geometric transformations. It is widely used for applications like real-time image and video analysis, robotics, autonomous vehicles, and medical image analysis, among many others. The library is known for its extensive set of algorithms and its active community, which continually enhances and maintains the software, making it a valuable resource in the field of computer vision.

MICROSOFT EXCEL-

Microsoft Excel is a popular spreadsheet software application developed by Microsoft. It is part of the Microsoft Office suite of productivity software. Excel is primarily used for creating, organizing, and analyzing numerical data through a grid of cells arranged in rows and columns. Users can perform various tasks, including data entry, mathematical calculations, data visualization, and data analysis.

Excel provides a wide range of features and functions, including formulas, charts, graphs, pivot tables, and data validation, making it a versatile tool for tasks such as budgeting, financial analysis, project management, data modeling, and creating reports. It is widely used in business, finance, academia, and many other fields for its ability to efficiently manipulate and present data in a structured and user-friendly manner

CHAPTER 3

PROPOSED METHODOLOGY

3.1 EXISTING VS PROPOSED

Existing color detection systems often rely on traditional computer vision techniques like thresholding and color space conversions (e.g., RGB to HSV). They may struggle with accuracy in complex environments or under varying lighting conditions. Machine learning models, if used, may require extensive training data and tuning.

In contrast, a proposed color detection system would leverage advanced machine learning algorithms such as deep learning networks. This approach could improve accuracy by learning complex color patterns and adapting to different scenarios. Integration with real-time image processing techniques would enhance speed and reliability, making the system suitable for dynamic environments like robotics and industrial automation. Overall, the proposed system aims for higher accuracy, adaptability, and efficiency compared to existing methods.

3.2 ALGORITHM USED

```
import cv2

import pandas as pd


img = cv2.imread('glass.jpg')


# declaring global variables (are used later

on) clicked = False

r = g = b = x_pos = y_pos = 0
```

```
# Reading csv file with pandas and giving names to each
column index = ["color", "color_name", "hex", "R", "G",
"B"]
csv = pd.read_csv('colors.csv', names=index, header=None)

# function to calculate minimum distance from all colors and get the most
matching color
def get_color_name(R, G,
    B): minimum = 10000
    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

# function to get x,y coordinates of mouse double
click 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)
cv2.namedWindow('image')
cv2.setMouseCallback('image',
draw_function)

while True:

   cv2.imshow("image",
   img) if clicked:

      # cv2.rectangle(image, start point, endpoint, color, thickness)-1 fills
entire rectangle
      cv2.rectangle(img, (20, 20), (750, 60), (b, g, r), -1)

      # Creating text string to display( Color name and RGB values )
      text = get_color_name(r, g, b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' +
      str(b)

      # cv2.putText(img,text,start,font(0-
      7),fontScale,color,thickness,lineType ) 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 black
      colour if r + g + b >= 600:
         cv2.putText(img, text, (50, 50), 2, 0.8, (0, 0, 0), 2, cv2.LINE_AA)
```

```
        clicked = False

    # Break the loop when user hits 'esc'
    key if cv2.waitKey(20) & 0xFF == 27:
        break

cv2.destroyAllWindows()
```

3.3 WORKING PRINCIPLE

The working principle of a color detection project involves several key steps:

1. **Image Acquisition: Capture images or video frames using a camera or input device.**
2. **Preprocessing: Apply preprocessing techniques like noise reduction, resizing, and color space conversion (e.g., RGB to HSV) to enhance image quality and standardize color representations.**
3. **Color Model Selection: Choose an appropriate color model (e.g., RGB, HSV) based on the project's requirements and the nature of color information in the images.**
4. **Color Segmentation: Use techniques such as thresholding, clustering, or region-based segmentation to separate regions of interest based on color characteristics.**
5. **Feature Extraction: Extract relevant features from segmented color regions, such as color histograms, spatial distribution, or texture information.**
6. **Machine Learning (Optional): Train machine learning models, such as neural networks, with labeled color data to learn patterns and classify colors accurately.**
7. **Color Classification: Utilize the selected color model and extracted features to classify colors and identify specific color categories or objects.**
8. **Post-processing: Apply any necessary post-processing steps, such as filtering or smoothing, to refine color detection results and improve overall accuracy.**
9. **Output: Present the color detection results visually or in a structured format, depending on the project's application requirements.**

By following these steps, the color detection system can effectively analyze color information in images or video streams and accurately classify colors based on the specified criteria.

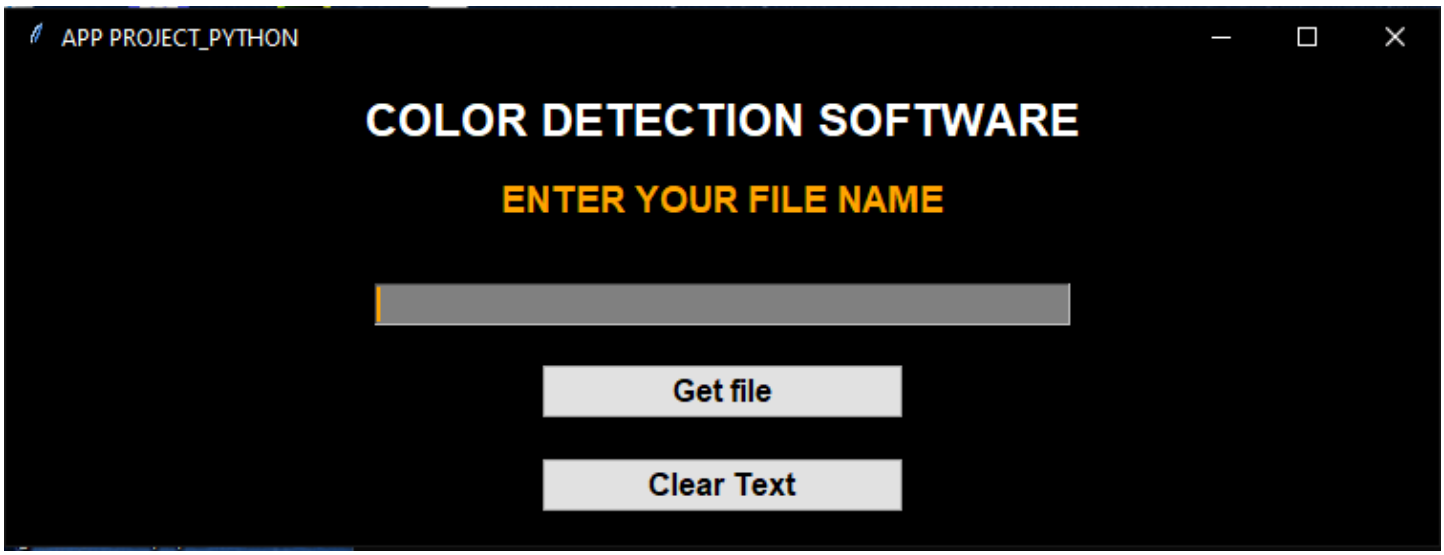
CHAPTER 4

RESULTS AND DISCUSSION

This chapter includes the system design and the attached screenshots of our minor project.

4.1 TESTING

By uploading images with proper background that possess objects, we get the actual color of each image



The test images used on the appliance for testing are shown above, and therefore the output images are shown below.

4.2 RESULT VALIDATION

The result has been obtained.

By testing the Object color detection system with multiple sample images with proper background. The results obtained perfectly matches with the corresponding colors of objects. Hence, from the above result validation the system can be worked in multiple applications.

CHAPTER 5

CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

The Color Detection Python Project represents a promising endeavor with significant potential. By providing a versatile and user-friendly solution for color identification and analysis, it addresses the diverse needs of professionals, students, and researchers in computer vision and image processing. The project's feasibility, both from a technical and economic standpoint, is encouraging. With robust technical foundations, a well- planned development roadmap, and potential revenue streams, it offers a pathway to sustainability. However, successful implementation will require diligent monitoring of evolving market demands and proactive risk management. In conclusion, the Color Detection Python Project has the capacity to empower users in the colorful world of computer vision while offering economic viability and value to its stakeholders.

4.1 FUTURE WORK

The future scope for the Color Detection Python Project is promising and multifaceted. There are opportunities to advance color analysis capabilities with sophisticated algorithms and integrate machine learning for enhanced accuracy and adaptability.
Customization features, real-time processing, mobile and cloud integration, and compatibility with IoT devices can expand the project's utility across various domains. Encouraging community contributions, developing educational resources, and exploring commercial versions will further enrich the project. Additionally, cross- platform compatibility and the exploration of new markets like fashion and design offer avenues for growth.

CHAPTER 5

REFERENCES

1. **OpenCV documentation: The official OpenCV documentation provides detailed information and examples on image processing, color detection, and computer vision techniques using Python and OpenCV. You can find it at https://docs.opencv.org/.**

2.PyImageSearch: PyImageSearch is a popular blog that covers various topics related to computer vision, including color detection. They provide tutorials, code snippets, and practical examples using Python and OpenCV. You can visit their website at https://www.pyimagesearch.com/.

3.OpenCV-Python Tutorials: This repository on GitHub contains a collection of tutorials and examples for using OpenCV with Python. It covers topics like image processing, object detection, and color analysis. You can access it at https://github.com/opencv/opencv-python.

4.Stack Overflow: Stack Overflow is a valuable resource for finding answers to specific programming questions related to color detection, image processing, and OpenCV. You can search for relevant questions and answers or ask your own questions on the platform. Visit https://stackoverflow.com/.