International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Color Detection Using Python

Mr.A.Krishna Kumar[1], Y.Sri Devi[2], Shaik Baba Bhasheer Lal[3], Bh.Sri Sai Rakesh Varma[4], N. Sai Santhosh Sarma[5], V. Madhupavani[6]

[1]Associate Professor, CSE Department, Vsm College Of Engineering, Ramachandrapuram, Dr.B.R.Ambhedkar Konaseema Dt, Andhra Pradesh - 533255
[2,3,4,5,6]Student, CSE Department, Vsm College Of Engineering, Ramachandrapuram, Dr.B.R.Ambhedkar Konaseema Dt, Andhra Pradesh - 533255

ABSTRACT
In the ever-evolving digital landscape, the significance of color detection is paramount across a variety of fields such as computer vision, design, medical imaging, and artificial intelligence. The ability to accurately detect and identify colors plays a crucial role in numerous real-time applications. This project introduces a user-centric, efficient, and reliable Python-based system for color detection using the OpenCV library. The proposed solution enables users to upload any image, interact with it through mouse clicks, and instantly retrieve the color name and corresponding RGB values of the selected pixel. The system matches the clicked pixel's RGB values with a database of predefined colors to determine the closest match. It also visually displays the detected color on the screen using a color-filled rectangle and textual information, making the tool intuitive and informative.
Color detection can be applied to various applications. This color information can also be applied in license plate detection which aids in tracking the vehicle. Here, the name of the color is detected using RGB color model and the corresponding Red, Green and Blue values are displayed, the object detection and tracking is performed using HSV color model in OPENCV, python.

Keywords: Color Detection, Python, OpenCV, RGB,Image Processing, Real-Time Applications, Computer Vision

INTRODUCTION

Colors form a fundamental component of human perception and digital communication. Whether it is in branding, user interface design, robotics, or digital photography, color information conveys meaning, enhances aesthetics, and provides critical cues. The ability to programmatically detect and interpret color data from images opens up avenues for automation, intelligent systems, and innovative user experiences. Color detection refers to identifying and retrieving the specific color present at a particular pixel location in a digital image. Manual methods of color identification are prone to human error and lack precision, especially when dealing with subtle hues and gradients. Moreover, tools like Adobe Photoshop, though powerful, are not suitable for quick or automated color identification tasks.
The goal of this project is to simplify the color detection process by developing a Python application using OpenCV. The system provides users with the ability to click on any point in an image and instantly obtain the color name and RGB values. Additionally, the application visually reinforces the detection result by displaying a color-filled rectangle and the detected color name. This interactive approach makes it an

excellent tool for education, experimentation, and professional use.

EXISTING SYSTEM

In the current digital environment, color detection and selection are typically performed using specialized software or web-based utilities. Some of the commonly used tools include:

Adobe Photoshop or Illustrator: These are powerful image editing tools that allow users to select a pixel using the eyedropper tool and retrieve its color values in various formats such as RGB, HEX, CMYK, etc.

Online Web-Based Color Pickers: Numerous websites offer simple tools where users can upload an image, click on it, and get the color details. Examples include imagecolorpicker.com or htmlcsscolor.com.

Built-In Developer Tools in Browsers: Browsers like Chrome or Firefox provide tools for front-end developers to inspect page elements and retrieve color codes used in CSS and design layouts.While these systems serve the purpose of color identification, they come with limitations when applied in learning environments or when integration with custom applications is needed. For instance, tools like Photoshop are paid and complex, making them less ideal for beginners or academic use. Online pickers lack flexibility, cannot be integrated into other Python-based workflows, and often depend on a stable internet connection.Additionally, these tools generally provide limited information about color beyond RGB or HEX values. For projects that require deeper analysis, such as converting to HSV or matching with a custom dataset of color names, these tools fall short. They also do not support real-time interaction through code, making them unsuitable for automated or smart systems development.In contrast, the proposed system in this project offers a programmatic, flexible, and free solution for identifying colors through direct interaction with images using Python.

PROBLEM STATEMENT

With the rising importance of automation and visual data interpretation, a reliable tool for detecting and identifying colors from images is necessary. Traditional color detection methods either require high-end software or offer limited precision. Moreover, there is a lack of tools that are both educational and functional for real-time color identification.

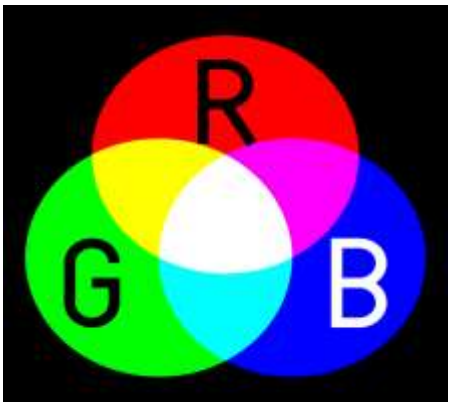


Fig:1

PROPOSED SYSTEM

The proposed solution is a Python-based application that uses the OpenCV library for image handling and processing. The system reads an image from the user, displays it in a window, and tracks the user's mouse interactions. When the user double-clicks on a particular pixel in the image, the RGB values of that pixel

are extracted. These values are compared with a predefined dataset of color names and RGB values to find the closest match based on a distance metric.

Functional Features:

Easy to install and run using Python
Loads images of various formats including JPG, PNG, BMP
Detects and displays color name and RGB value on mouse click
Uses Euclidean distance for closest color matching
Displays detected color using a graphical rectangle
Provides real-time, dynamic interaction with the image

Non-Functional Features:

Fast response time (~50ms per click)
Low memory usage (under 50MB)
Platform-independent(works on Windows, Linux, macOS)

LIBRARIES USED

OpenCV: Python is a library of Python bindings designed to solve computer vision problems. cv2.imread () method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Syntax: cv2.imread (path, flag)

Using OpenCV library, you can Read and write images ,Capture and save videos ,Process images (filter, transform),Perform feature detection, Detect specific objects such as faces, eyes, cars, in the videos or images. Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it. OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux,FreeBSD, Net BSD, Open BSD, etc.

NumPy: Numpy stands for Numerical Python. It is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. It is an open source project and you can use it freely. In Python we have lists that serve the purpose of arrays, but they are slow to process. It aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

Pandas : pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

ALGORITHMS

Color Space Conversion and Thresholding

One of the most common and fundamental approaches in color detection is based on color space conversion and thresholding using the OpenCV library. In this method, the image is first converted from the BGR color space (used by OpenCV) to the HSV (Hue, Saturation, Value) color space. HSV is preferred over RGB for color detection because it separates the color information (hue) from the intensity (value),

making detection more accurate under varying lighting conditions. After conversion, specific color ranges are defined using lower and upper HSV boundaries. A mask is created using the cv2.inRange() function, which highlights the pixels falling within the specified range. This mask is then applied to the original image to isolate the desired color.

K-Means Clustering

For identifying dominant colors in an image, the K-Means clustering algorithm is widely used. K-Means is an unsupervised machine learning technique that groups similar pixel values into clusters. By reshaping the image into a two-dimensional array of pixels, the algorithm can be applied to find cluster centers representing dominant colors. This method is particularly useful in applications like color palette generation, image segmentation, or visual content analysis. It efficiently highlights the most frequent colors by analyzing pixel similarity and clustering them based on their RGB values.

Contour Detection and Masking

Another important method in color detection is contour detection and masking, often used in real-time object tracking based on color. Once a binary mask is created for a specific color using cv2.inRange(), contours essentially the boundaries of colored regions are detected using OpenCV's find Contours() function. These contours can then be drawn onto the image to visually represent the location and shape of the color region. This technique is commonly used in applications such as traffic sign detection, object counting, or gesture recognition, where identifying and tracking colored shapes is essential.

SYSTEM ARCHITECTURE

The system architecture of the **Color Detection Using Python** project represents the interaction between the user and various components of the application to achieve accurate color identification. The architecture is structured in a **modular and sequential flow**, ensuring smooth operation from image input to color detection and result display.

Key components of the architecture Architecture:

User Input Module:

The user initiates the process by loading an image through the Python program.

The user interacts with the image via **mouse double-click** to select a pixel.

ImageProcessingModule(OpenCV): Displays the image in a GUI window.

Captures mouse events and passes pixel coordinates to the processing layer.

Pixel Analysis Module (NumPy):

Receives the (x, y) coordinates from the image display module.

Extracts the **RGB values** of the selected pixel.

Converts the RGB values to **HSV format** for enhanced perceptual accuracy.

Color Matching Module (Pandas + CSV):

Loads a **CSV dataset** containing known color names and RGB values.

Compares the extracted RGB values with the dataset using a distance metric.

Identifies the closest matching color name.

Result Display Module:

Overlays the color name and its RGB/HSV values directly on the image window using OpenCV.

Draws a rectangle filled with the detected color near the selected pixel for visual feedback.

Future Enhancements in Architecture:

Integration of **text-to-speech module** for audio output.

Addition of **web interface layer** using Flask/Django for online access.
Upgrade to **real-time camera feed support** for video-based color detection.

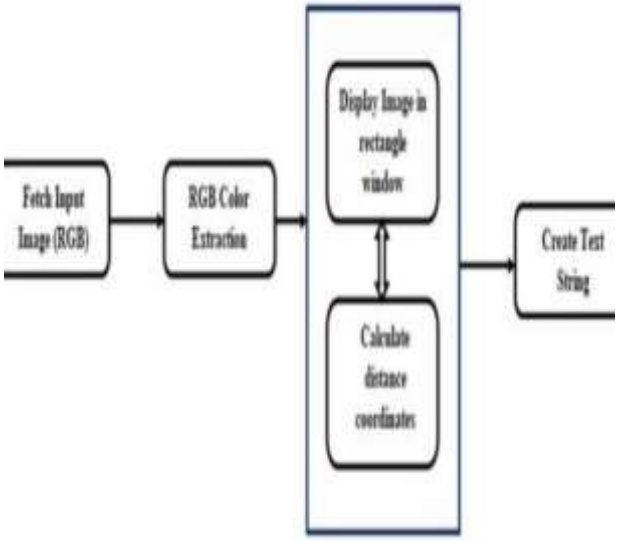
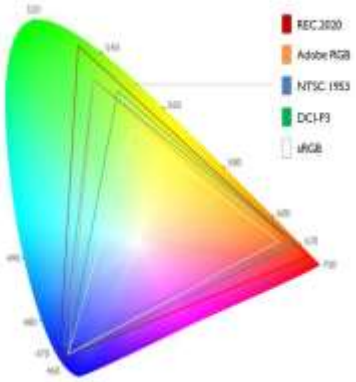


Fig:2 System Architecture Diagram

RGB AND HSV COLOR MODELS

PHYSICAL PRINCIPLES FOR THE CHOICE OF RED, GREEN AND BLUE:

A set of primary colors, such as the RGB primaries, define a color triangle; only colors within this triangle can be reproduced by mixing the primary colors. Colors outside the color triangle are therefore shown here as gray. The primaries and the D65 white point of sRGB are shown. The background figure is the CIE xy chromaticity diagram. The choice of primary colors is related to the physiology of the human eye; good primaries are stimuli that maximize the difference between the responses of the cone cells of the human retina to light of different wavelengths, and that thereby make a large color triangle.



9.2 HSV COLOR MODEL

Hue specifies the angle of the color on the RGB color circle. A 0° hue results in red, 120° results in green, and 240° results in blue.Hue is the color portion of the model, expressed as a number from 0 to 360 degrees:
Red falls between 0 and 60 degrees.
Yellow falls between 61 and 120 degrees.

Green falls between 121 and 180 degrees.
Cyan falls between 181 and 240 degrees.
Blue falls between 241 and 300 degrees.
Magenta falls between 301 and 360 degrees.

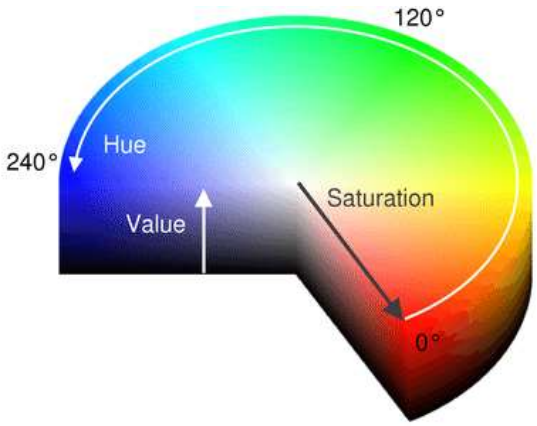


Fig:4

RESULT ANALYSIS
COLOR DETECTION USING PYTHON

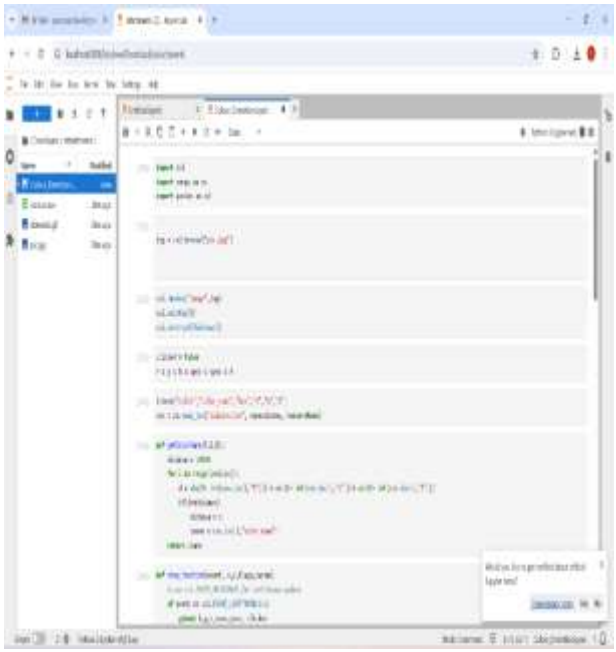


Fig:5

COLORED DATA SET

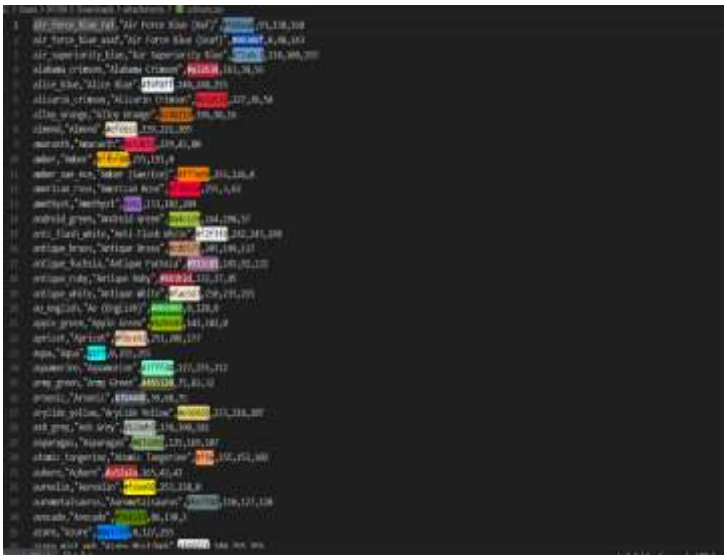


Fig:6

Output:

Fig:7

Fig:8

CONCLUSION

To achive the goal the first step is to capture the video through webcam.This is taken as input .The image frames are accessed from the video stream. The HSV color model can handle changes caused by lighting. Hence the image stored in RGB format has to be transformed into HSV .This color model describes color in terms of the amount of gray and their brightness value. Hue value is extended from 0-179,Saturation value is extended from 0-255 and Value is extended from 0-255 respectively. The corresponding is made by defining the range of each color (red, green and blue).Noise causes internal imperfections,hence it has to be removed. Morpholoagical technique called diation is used for this purpose. To specifically detect red,green and blue and discard others ,bitwise and is performed between the image frame and mask. All the points which are having same color or intensity have to be bounded by a rectangular box .As the object with red, green or blue color advances in the live recording, the bounding box alse advances with it .Here the goal for detecting and tracing the red ,green and blue colored object is achieved.

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.175178,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. S. Matuska, R. Hudec and M. Benco, "The comparison of CPU time consumption for image processing algorithm in Matlab and OpenCV," 2012 ELEKTRO, 2012, pp. 75-78, doi: 10.1109/ELEKTRO.2012.6225575.
4. 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.
5. K. Cameron and M. S. Islam, "Multiple Objects Detection using HSV," NAECON 2018 - IEEE National Aerospace and Electronics Conference,2018,pp.270-273, doi:10.1109/NAECON.2018.8556