Smart Agent Framework for Color Selection of Wall Paintings

Mallikarjuna Rao Gundavarapu, Abhinav Bachu, Sai Sashank Tadivaka, G. Saketh Koundinya, and Sneha Nimmala

Abstract Smart color selection agent for wall painting is extremely useful for people while selecting desired colors. In traditional approach, for painting the house/building, the physical agent on behalf of company visits customer site and provides bulky wall painting color catalog. However, the approach many times results customer unsatisfaction due to technical/manual mistakes. 'Smart Agent Framework' developed in this paper addresses this problem by providing all details of the selected color at customer site itself. This in turn reduces the time and human effort required for both customer and company. As our framework is built on Python platform, the agent is robust, scalable, and portable. For our experimentation, we have created a dataset containing 860 colors. Since the agent is embedded with Google Text-to-Speech (gTTS), customer will get auditory response for his/her color selection. The three best matches were provided to enhance the satisfaction as well as to avoid manual/technical errors.

Keywords Color detection · Text-to-Speech · RGB values

1 Introduction

The procedure of detecting the name of any color is known as color detection. The brain and eyes of humans work together to transform light into color stimulus. Firstly, the signal is sent to the brain from the light receptors which are present in eyes of a human being. Then, human brain recognizes the color that is seen. Humans have mapped certain lights with their color names. A similar strategy is used to detect color names. Any color is a mixture of primary colors (red, green and blue). A dataset which contains the color names and its values is used in this project.

M. R. Gundavarapu (✉) · A. Bachu · S. S. Tadivaka · S. Nimmala
Department of CSE, GRIET, Hyderbad, India
e-mail: gmallikarjuna628@grietcollege.com

G. S. Koundinya
SAP Consultant, Bangalore, India
e-mail: gsaketh@studentnitw.ac.in

V. Suma et al. (eds.), *Inventive Systems and Control*, Lecture Notes in Networks and Systems 436, https://doi.org/10.1007/978-981-19-1012-8_15

Firstly, the user is prompted to give the image path as input. This is facilitated by creating an argument parser. Additionally, the image is read using the Python OpenCV library. A window with the given input image opens. The RGB values of colors are recognized when the user clicks on any location on the image using the *XY* coordinates of the current location. The shortest distance is now determined, assisting in the retrieval of the top three colors and simultaneously a voice with the detected color also will play. This is attained with the help of Python gTTS and playsound modules. Color detection is helpful in creating an application which teaches colors to the kids, recognizing objects, and it is also used as a tool in different image editing and drawing apps. Various table text styles are provided. The formatter will need to create these components, incorporating the applicable criteria that follow.

2 Literature Survey

Color is defined as the appearance of objects as a result of the various characteristics of light that they reflect or emit. The method of recognizing the color name is known as color detection. For humans, light enters the eye and travels to the retina at the rear of the eye. The retina is covered with millions of light-sensitive cells called rods and cones. These cells help to identify colors. When these cells detect light, they send signals to the brain. Machines, on the other hand, do not work in the same way. They recognize colors based on the data they have. Normally, we define each color value within a range of 0 to 255. There are approximately 16.5 million different ways to represent a color. Humans can identify limited number of colors. Machines, on the other hand, can detect a wide range of colors and distinguish between different variants of the same color. Given how difficult it is for humans to identify color variants, the proposed method is particularly useful while choosing colors for wall paintings. It assists customers in selecting the exact color they require. This approach allows children to improve their color recognition skills by recognizing different hues of color. This application aids blind persons in identifying colors as it is enabled with a voice feature [1]. This application is used in image processing, digital signal processing, and object identification.

The project's main goal is to act as a guide for choosing colors for wall paintings. Color choices may not be correct if people are involved. For the same color, there are numerous variants or shades. There is a potential that the staff misunderstands the color scheme chosen by the consumer. As a result, this method removes such inaccuracies and delivers not only the best color, but also two more colors that are similar to the original color. This gives customers a range of options to choose from while still getting the intended result. The voice option will be an extra feature because there are some color names that are difficult to pronounce for many people. The speech option aids in accurate pronunciation and assists consumers in accurately conveying the color name.

The application also possesses few real-world applications. Along with object detection, color detection in real time is critical for a robot to view the environment.

Color detection helps to detect the traffic signals in self-driving cars like Tesla [2]. It reduces human effort and will be accurate. Many people now a days disregard traffic laws by jumping signals; nevertheless, this feature in self-driving cars requires people to obey traffic laws by requiring them to stop at red lights. It can also be utilized in sectors where machines are used to accomplish tasks such as picking and placing products in different colored objects. It saves time and improves work efficiency. It is also cost effective because it only requires a single payment. It saves money on labor expenditures, and the risks of error are negligible when compared to humans. People may make mistakes when allocating items to objects, but when a machine is involved, the possibilities of errors are minimal. It is usual to discover applications in image processing and computer vision, where it is important to recognize reference points with extreme color, such as a primary color RGB or complementary cyan, magenta, and yellow (CMY) with very high saturation. As a result, there are instances where a group of objects can be differentiated by their distinctive extreme color, which can be used as to identify objects. The significance of color abstraction has been dealt in previous research papers [3–5].

3 Proposed System Architecture

Architectural diagram is a representation of a system which helps in summarizing the entire outlines of programs, associations, conditions, and divisions between elements. It allocates an entire view of the concrete arrangement of the software system and its plan of advancement. The architecture depicted in Fig. 1 demonstrates the project's characteristics. It is portrayed as a well-defined model that hides the source code. It helps us to measure the performance of the application. The performance is determined by computing the time required to determine the possible colors. It also emphasizes the importance of extensively used modules such as OpenCV and Pandas.

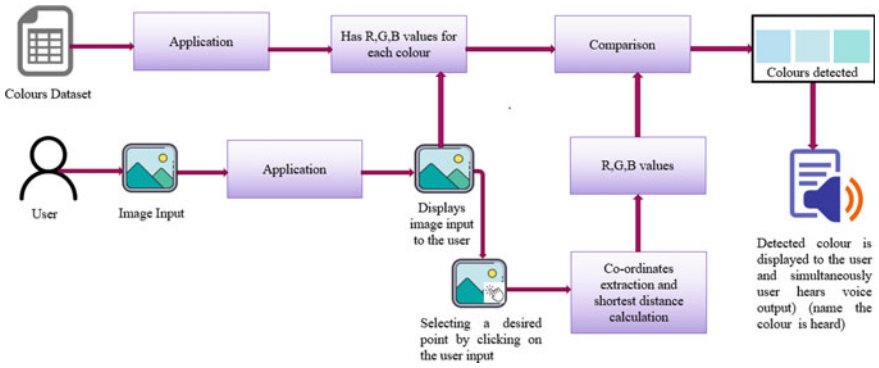

Fig. 1 System architecture

The identified dataset is read by the application using the Pandas library. The colors dataset which is used in our application has R, G, and B values and hexadecimal code for all the colors. These values are useful for comparison and help in detecting the user-selected color with ease. Initially, when the user provides image as input to the application, it displays the image to the user using draw function. After the user sees the given input image and double clicks on any desired point of the image, the application detects the coordinates and calculates the shortest distance. The coordinates extraction is done with the help of mouse click event. The calculated distance is useful to identify how close we are present to the color in the dataset and chooses the three colors having minimum distance. After the colors are detected, they are displayed on the screen along with RGB values, and the user will be able hear the text (name of the color along with RGB values) in the form of speech. This is implemented with the help of gTTs and playsound modules. The stored audio files satisfy the requirement of mp3 format.

4 Development Framework

4.1 Software Installation

Step-1: Visit the following website https://www.python.org/downloads/release/python-396/

Step-2: Download Python software according to your system requirements
Step-3: Now run the downloaded installer

a. Right click on the installer
b. Select Run as Administrator and click on Yes

Step-4: Now Python installer wizard appears

c. Check the option Add Python 3.9 to PATH
d. Click on Install Now

Step-5: After the setup was successful, click on close
Step-6: Open command prompt

e. On the taskbar, to the bottom left, there will be an option to search. Type cmd in search and open the command prompt.
f. Execute the following command to check if Python is properly installed.

- Python version

g. If the version is displayed, then the Python is properly installed

Step-7: Execute the following command to update pip

- pip install–upgrade pip

Step-8: Install Pandas by executing the following command.

- pip install pandas

Step-9: Install gTTS by executing the following command.

- pip install gtts

Step-10: Install playsound by executing the following command.

- pip install playsound

Step-11: Install OpenCV by executing the following command.

- pip install opencv-python

Step-12: Install argparse by executing the following command.

- pip install argparse

4.2 Execution

1. Image Input: The initial step is to take an image as input from the user. To get better results, a high-quality image with good resolution is preferred. Image input will be taken from the command prompt, and the user has to make sure that image should be present in the current working directory or the full path of the image must be specified. The aforementioned procedure is carried out with the assistance of the argparser module, which extracts the image from the command prompt.
2. Loading Image: Using Python's OpenCV module, the image from the previous step is now loaded into the application. Cv2.imread() function is used to load the image from the file. The application and image should be present in the same directory or the entire location of the image must be specified.
   image = Cv2.imread(image location).
3. Color Recognition: In this phase, we must teach the application to recognize colors. This can be accomplished by establishing a dataset including the names of the colors as well as their RGB values, which can then be mapped to the RGB values obtained. We used RGB format as our datapoints because many of the colors are specified by red, green, and blue. We have built a dataset which contains 865 different colors.
4. Loading Dataset: The Pandas module is used to load the external dataset into the program. The read_csv() method in Pandas assists in loading the dataset.
5. Extracting RGB Values: In this stage, when the user double clicks on any part of the image, the XY coordinates of the current location are extracted and are converted into RGB values using pixel() method or by arraying of XY coordinates. The obtained RGB values are then passed to another function to obtain the possible colors. This process gets executed when a mouse click event occurs.

b, g, r = image [X, Y].
(or).
RGB-value = Image.getpixel(X, Y).

6. Obtaining Best Possible Colors: The obtained RGB values are passed to another function which calculates the minimum distance. In this stage, the dataset is iterated in such a way that three possible RGB values which are closest to the obtained RGB values are extracted. The resulting RGB values are used to obtain their respective color names from the dataset. The color names are appended into a list in the decreasing order of their priority.
7. Displaying Color Names: In this phase, the color names for the list obtained in the above process are extracted and displayed on the screen in decreasing order of their priority separated by a new line. The color names are also accompanied by the RGB values of the current location. This process is executed with the help of OpenCV's PutText function.
8. Converting Text-to-Speech: Finally, the color names which present in the text format are converted into and audible object using gTTS module. Then it is played using playsound module. When the user double taps on any region of the image, three most probable colors and RGB values appear, which can also be heard as speech. And the time complexity of entire code was very speedy as compared to previous approaches which took 1–2 min.
9. Exiting the Application: Not only should an application be able to work well, but it should also be simple to exit. As a result, we designed a feature that closes the applications when the user presses the escape key.

5 Existing Approaches

There a quite a few approaches. A new scheme is introduced to improve the accuracy of face detection using RGB color space which results in improvement of equal error rate from 3.3% to 1.8% [6]. Another approach is by calculating hue-saturation-value. This technique converts the XY coordinates of the chosen place to RGB values, which are then converted to HSV (hue-saturation-value) color system [7] (Fig. 2).

Other approach is that an algorithm is used for edge detection of a color image using threshold technique [8]. A method is proposed for recursive detection of edges in color image based on green's function approach in color vector space field [9]. Another technique is color detection using hierarchical neural networks in RGB space [10]. A color recognition system is built using finger interaction method for object color detection [11]. An approach is proposed which detects the color in RGB-modeled images using MATLAB [12]. Finally, an approach uses computer vision for color detection [13]. Color components have been successfully explored even in vegetables classification. [14, 15]. The methodology used for fire detection as well as relationship between color and antioxidant capacity of fruits and vegetables by color pixel classification. The authors [16–18] color coding and image coding have been dealt in detail.

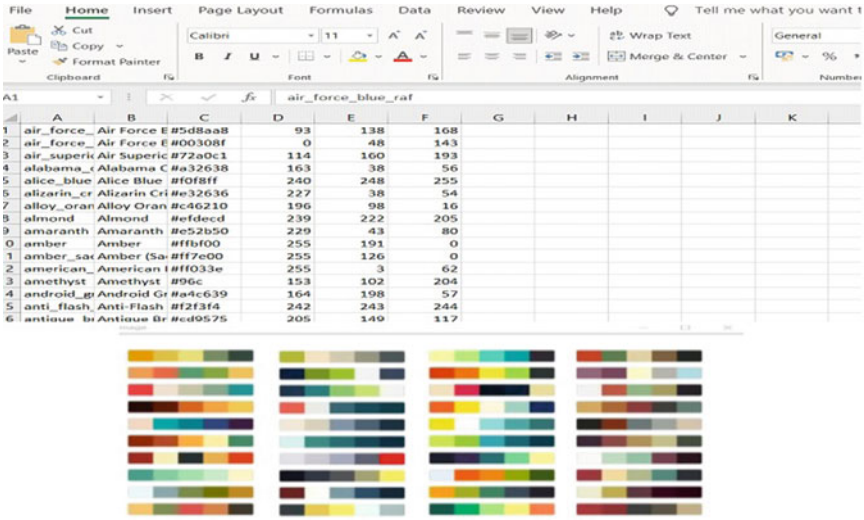

Fig. 2 Colour compositions

6 Experimental Results

Initially, we are using argparse module to create an argument parser which enables us to directly give the location of the image from the command prompt. Then we read the location of the image using OpenCV's 'imread' method and store it in a variable.

With the help of Pandas module, the command (pd.read_csv) reads the dataset which is in form of CSV file and then loads it into the Pandas DataFrame. We have assigned a name to each column for easy accessing.

Now we need to create a window which displays the input image. To develop this functionality, we used draw_function whenever a mouse event occurs. The 'draw_function' will also be useful to calculate the RGB values of the selected portion of the image.

The function parameters are the name of the event and coordinates of the pixel. When the event is double clicked, then we calculate the RGB values of the pixel using the coordinates. Now we define another function, which will return the color name form the CSV file by using the calculated RGB values.

To get the name of the color, we calculate a distance which tells us how close we are to the color and chose the one having least distance. Finally, when a double click event occurs, we draw a rectangle using 'cv2.rectange' method and get the name of the color to draw text on the rectangle using 'cv2.putText' method.

Fig. 3 Original image of wall painting

Finally, we convert the detected color into voice using gTTS module and play it using 'playsound()' method. Thus, when we double-click on any part of the image, we will be able to view the name of the color along with RGB values and also perceive it. When the user presses escape key, the application ends. We have to make sure that we give the image path using '-i' argument and also provide the exact location of the image.

Figure 3 depicts the image input given to the application. This is displayed to the user with the help of draw function. Figure 4 depicts the output image with three possible colors of the user-desired location. Along with the text displayed, the output will also come in form of voice. Here Pear, Amarnath, and Almond are the possible colors for the user-selected point.

Figure 5 depicts the image input given to the application. This is displayed to the user with the help of draw function.

Figure 6 depicts the output image with three possible colors of the user-desired location. Along with the text displayed, the output will also come in form of voice. Here Pine Green, Aliziran Crimson, and Alice Blue are the possible colors for the user-selected point.

Fig. 4 Output image with three possible colors of the specified location with speech

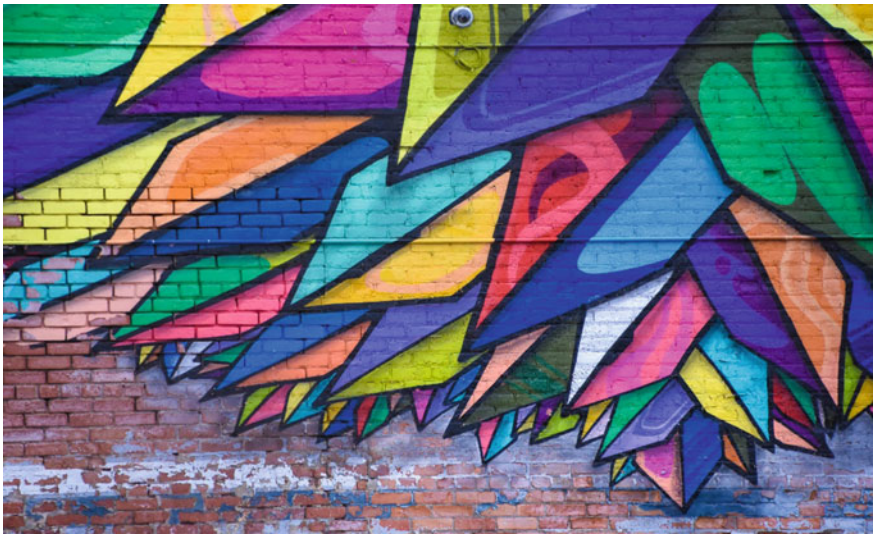

Fig. 5 Original image of wall painting

Fig. 6 Output image with three possible colors of the specified location with speech

7 Pseudocode

```
//Load the image to identify the color
Img=cv2.imread(image_path)
// load data set using pandas
cds = pd.read_csv('colors.csv')
// Extract R, G, B components of the selected image portion using in
components function
[R G B] = imcomponents(Image)
// Train the RGB dataset
// Obtain true color three best matches for testing place of image
using Euclidian distance
d=abs(R-int(csv.loc[i,"R"]))+
abs(G-int(csv.loc[i,"R"]))+abs(int(csv.loc[i,"R"]));
// generate auditory response using GTTS and play sound
myobj=gTTS(text=mytext,lang=language,slow=False)
playsound(name)
// Display original image with true color values
```

8 Conclusion and Future Scope

In this project, by using concepts of Python such as Pandas, gTTS, playsound, OpenCV, and ArgParse, we are able to successfully.

- Detect around 850–860 colors correctly.

- Detect the possible colors pointed in the image input given by the user.
- Display the color detected in form of text, and simultaneously play the generated voice output which reads out the name of the color.

This project can further be enhanced by converting this project into an app, and it would be more convenient for the user for using the developed features and by converting this project into a machine learning model, to make the color detection more efficient and also decrease the response time.

Data science field is mainly useful to imitate human tasks. Vision and speech are two essential components of human interaction, which data science has already begun to mimic. The main objective of this project is to detect a color and get to know about different shades of a color. There are 16.5 million ways to represent a color, though we did not use all the colors, but this project gives an idea about the basic colors and its different variations. Color detection is essential for recognizing objects, and it is also used as a tool in various drawing and image editing apps. There are different color names for which we do not know the correct pronunciation. The voice function makes it possible to know the exact pronunciation which is easily understandable. This project is helpful for any different applications like segmentation, image matching, recognition of objects, and visual tracking in the fields image processing and computer vision.

References

1. Navada BR, Santhosh KV, Prajwal S, Shetty HB (2014) An image processing technique for color detection and distinguish patterns with similar color: an aid for color blind people. International conference on circuits, communication, control and computing, pp 333–336. https://doi.org/10.1109/CIMCA.2014.7057818
2. Fleyeh H (2004) Color detection and segmentation for road and traffic signs. Cybernetics and intelligent systems 2004 IEEE conference on, vol 2, pp 809–814
3. Kumar ST (2021) Study of retail applications with virtual and augmented reality technologies. J Innovative Image Process (JIIP) 3(02):144–156
4. Kulshreshtha K, Niculescu AI, Wadhwa B (2017) On the design and evaluation of Nippon paint color visualizer application—a case study. IFIP conference on human-computer interaction, pp 372376
5. Podpora M, Korbas GP, Kawala-Janik A (2014) YUV vs RGB-choosing a color space for human-machine interaction. FedCSIS position papers, pp 29–34
6. Bours P, Helkala K (Aug 2008) Face recognition using separate layers of the RGB image. Information hiding and multimedia signal processing 2008. IIHMSP 08 international conference on, pp 1035–1042
7. Sebastian P, Voon YV, Comley R (June 2008) The effect of color space on tracking robustness. Industrial electronics and applications 2008. ICIEA 2008. 3rd IEEE conference on, pp 2512–2516
8. Dutta S, Chaudhuri BB (2009) A color edge detection algorithm in RGB color space. International conference on advances in recent technologies in communication and computing, pp 337–340
9. Zareizadeh Z, Hasanzadeh RPR, Baghersalimi G (2013) A recursive color image edge detection method using green's function approach. Optik—Int J Light Electron Optics 124(21):4847–4854

10. Altun H, Sinekli R, Tekbas U (2011) An efficient color detection in RGB space using hierarchical neural network structure. International symposium on innovations in intelligent systems and applications (INISTA), pp 154–158
11. Manaf AS, Sari RF (2011) Color recognition system with augmented concept and finger interaction. Ninth international conference on ICT and knowledge engineering, pp 118–123
12. Duth PS, Deepa MM (May 2018) Color detection in RGB-modeled images using MAT LAB. Int J Eng Technol [S.l.], 7(2.31):29–33. ISSN 2227-524X
13. Senthamaraikannan D, Shriram S, William J (2014) Real time color recognition. Int J Innovative Res Electr, Electron, Instrum Control Eng 2(3)
14. Manan A, Bakri NS, Adnan R, Samad, Ruslan FA. A methodology for fire detection using colour pixel classification. 2018 IEEE 14th international colloquium on signal processing and its applications (CSPA)
15. Comert ED, Mogol BA, Gokmen V (June 2020) Relationship between color and antioxidant capacity of fruits and vegetables. Curr Res Food Sci, Elsevier 2:1–10
16. Pasumpon (2021) Review on image recoloring methods for efficient naturalness by coloring data modeling methods for low visual deficiency. J Artif Intell 3(03):169–183
17. Bianco S, Gasparini F, Schettini R (2015) Color coding for data visualization. Encyclopedia of information science and technology, pp 1682–1691
18. Sathesh A (2020) Light field image coding with image prediction in redundancy. J Soft Comput Paradigm 2(3):160–167