|  |  |  |
| --- | --- | --- |
| **الجـامعــــــــــة الإســـــلاميــة بغــزة**  **كليـــة تكنــولــوجيـــا المعــلــومـــات**  **قـــــســــــم الــدراســـــــات العـلــيـــا**  **تخصـص تكنـولـوجيــــا المعلومــــات** | 2638f11f-5366-43fc-9b5a-c3323ac29be7 | **Islamic University of Gaza**  **Faculty of Information Technology**  **Department of Graduate Studies**  **Information Technology Specialization** |

**project:**

**Color Detection**

**Image Processing Course**

Prepared by:

|  |  |
| --- | --- |
| **120233046** | **Alaa Emad Al Hout** |

Submitted to:

**Dr. Ashraf Younis Al-Maghari**

May 14, 2025

Table of Contents

[Project Introduction 3](#_Toc198115278)

[Why is color detection important? 4](#_Toc198115279)

[Project Environment and Tools Used 4](#_Toc198115280)

[Code Workflow and Functionality 5](#_Toc198115281)

[Execution Results and Image Samples: 8](#_Toc198115282)

[Statistics and Frequency Analysis 11](#_Toc198115283)

[Lessons Learned 11](#_Toc198115284)

[Future Enhancements 12](#_Toc198115285)

[Conclusion and Recommendations 12](#_Toc198115286)

# Project Introduction

Color detection is one of the core tasks in the field of computer vision, serving as a critical foundation for higher-level applications such as object recognition, tracking, sorting, and even scene understanding. This project presents a practical implementation of a color detection system that leverages the OpenCV library to identify and process basic colors—specifically red, green, and blue—from both static images and real-time video feeds. What sets this project apart is its combination of real-time visual feedback and automated data recording.

Upon detecting a color in an image, the system not only annotates the result visually by drawing a colored rectangle and labeling the hex code but also stores the detection data—including the image name, color name, and its hex value—in a structured Excel file. To enhance readability and utility, the Excel file is dynamically formatted to fill cells with the detected color, offering a clear visual correspondence between the analysis and the dataset.

In addition to real-time detection and data logging, the system generates basic statistics on color frequency across the dataset, allowing users to understand patterns and distribution of colors. This makes the project valuable not only from a technical standpoint but also for educational purposes, demonstrations, and potential use in prototyping industrial inspection tools.

Designed with modularity and simplicity in mind, the project offers a clean codebase that can easily be expanded to support more colors, integrate a graphical interface, or incorporate machine learning models for enhanced accuracy. As such, it serves as both a learning platform and a functional tool—demonstrating how vision systems can bridge the gap between human perception and digital decision-making.

# Why is color detection important?

* Serves as a fundamental step in computer vision to interpret visual content.
* Helps in identifying and distinguishing objects based on their color.
* Used in industrial applications for product sorting and quality control.
* Enables intelligent systems to simulate human perception of color.
* Applied in medical imaging to detect conditions based on color patterns.
* Acts as an intuitive and educational entry point for beginners in image processing.
* Facilitates object tracking in videos and smart surveillance systems.
* Enhances user interfaces through gesture or color-based interactions.
* Integrates easily with data analytics to generate color-based reports.
* Works seamlessly with other technologies like machine learning and 3D vision.

# Project Environment and Tools Used

This project was developed using the following tools:

* Python 3.11
* OpenCV for image processing (reading, color conversion, and masking)
* NumPy for array computations and logical operations
* Pandas for handling structured data and exporting to Excel
* openpyxl for formatting Excel sheets and coloring cells using hex values
* datetime and os libraries for file management and timestamping
* Microsoft Word for documentation

# Code Workflow and Functionality

1. **Mode Selection**:  
   When the program starts, it prompts the user to select between two modes:

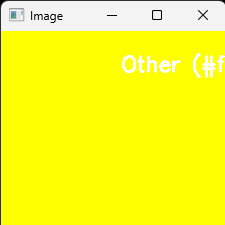
* **Camera Mode (1)**: Processes real-time frames from the webcam.
* **Folder Mode (2)**: Processes all images inside the predefined img\_test folder.

1. **Image Acquisition**:
   * In camera mode, cv2.VideoCapture(0) is used to continuously capture frames from the default camera.
   * In folder mode, images with .png, .jpg, or .jpeg extensions are loaded from the folder using os.listdir() and cv2.imread().
2. **Color Space Conversion:**  
   Each image or frame is converted from BGR to HSV color space using cv2.cvtColor(image, cv2.COLOR\_BGR2HSV), which allows for more effective color filtering under variable lighting conditions.
3. **Color Detection via Masking**:
   * For each predefined color (Red, Green, Blue), a mask is created using cv2.inRange() with specific HSV lower and upper bounds.
   * The mask is refined using:
     + **Gaussian Blur**: cv2.GaussianBlur() to reduce noise.
     + **Morphological Opening**: cv2.morphologyEx() with a kernel to clean the mask.
4. **Fallback for Undefined Colors:**
   * If none of the defined masks detect a color, the program converts the image to grayscale and creates a thresholded mask to label the result as **"Other"**.
5. **Hex Color Calculation and Annotation:**
   * The average color of the masked region is computed using cv2.mean().
   * This BGR value is converted to a hex color string.
   * A rectangle is drawn on the image using cv2.rectangle() filled with the detected color.
   * The color name and hex code are displayed with cv2.putText().
6. **Data Storage for Logging:**
   * Each result is appended to a list as a row: [image name / timestamp, color name, hex code, ''].
   * At the end of processing, the data is exported to folder\_results.xlsx using pandas.DataFrame.to\_excel().
7. **Excel Formatting for Color Visualization:**
   * The saved Excel file is reopened with openpyxl.
   * Each cell in the “Color” column is filled with the actual detected color using PatternFill().
8. **Color Occurrence Statistics:**
   * A global color\_counts dictionary tracks the number of detections per color throughout the session.
   * After completion, statistics are printed to summarize how many times each color was found.
9. **Modular and Extendable Design:**
   * The code is divided into clear functions like detect\_color(), save\_to\_excel(), camera\_mode(), and folder\_mode().
   * This makes it easy to modify, scale, or upgrade in future versions (e.g., adding a GUI, more colors, or ML-based detection).

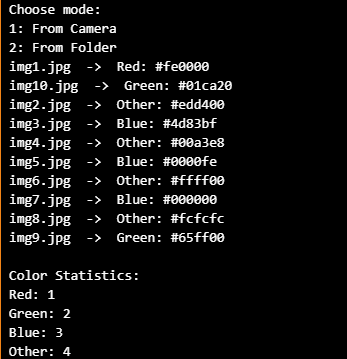
# Execution Results and Image Samples:

**- While running the code:**

During direct execution of the code, the detected color appears marked on the image, along with the color value and name, and changes depending on the image it loads.



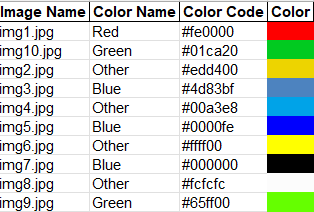
**-When the code stops**

****

The image above shows the final output of the system log after processing a batch of images. Each image is listed with its detected color and hexadecimal code, and the final color statistics are displayed below the list. This data is saved to an Excel file after printing it on the screen.

**-Results inside the Excel file**

The Excel file displays a row for each image containing the image name, color name, hexadecimal code, and a cell colored according to the image color the color was applied to for visual confirmation.



# Statistics and Frequency Analysis

The system was tested on a small set of 10 images. The results showed that blue was the most frequently detected color (3 times), followed by green (2 times), and red (1 time). The remaining images were labeled as 'Other' since they did not match any of the predefined HSV ranges. These numbers reflect the diversity of the sample images and demonstrate the system's ability to isolate primary colors using HSV masking techniques.

# Lessons Learned

* HSV color space is far more suitable for color detection than RGB because it separates hue from light intensity.
* Masking using well-defined color ranges yields reliable results when coupled with Gaussian blur and morphological operations.
* Organizing results using Pandas and exporting them to Excel makes data handling more efficient and professional.
* Excel formatting using openpyxl allows the system to create visually meaningful and readable logs.
* Modular code design makes it easier to maintain, expand, and test specific parts independently.
* Real-time image annotation improves the user experience and simplifies debugging during development.

# Future Enhancements

There are several ways this project can be extended. A graphical user interface can be created using Tkinter or PyQt to allow for user-friendly interaction, making it easy for non-programmers to utilize the tool. The system can also be extended to include more color categories such as yellow, orange, or purple by increasing the HSV ranges. Another potential upgrade would be to use machine learning models that can identify subtle or mixed colors more precisely. In addition, providing video file input support and constructing a dashboard for real-time visual tracking and record keeping would bring the system to a semi-professional level of usage.

# Conclusion and Recommendations

The present project skillfully integrates basic computer vision techniques with structured data management to create an extensive and informative system for detecting colors. By providing annotated feedback on images and presenting results in a color-coded Excel spreadsheet, it demonstrates the potential of code to bridge visual understanding and data interpretation. The simplicity of implementation, coupled with the possibility of future improvements, makes this project a solid foundation for more advanced real-time computer vision systems.