Day80_Multi-Color_Detection_using_OpenCV

September 4, 2025

1 Multi-Color Detection using OpenCV (with Colored Bounding Boxes)

1.1 Introduction

Color detection is one of the most fundamental tasks in **computer vision**.
It allows us to identify and isolate specific colors in real-time video streams.

1.1.1 Why is this unique?

- Instead of detecting just one color, we will detect **multiple colors**.
- We'll use **bounding boxes** (like face detection) to highlight where those colors appear.
- Bounding boxes will be drawn in the **same color as the detected object** (unique visual effect).
- Works with **live webcam feed**.

1.1.2 Applications:

- Object sorting (by color).
- Traffic light detection.
- Industrial inspection.
- Educational/learning projects.

1.2 Theory

1.2.1 Colors in Computer Vision

- OpenCV reads images in **BGR format**.
- Detecting colors directly in BGR is hard → **we convert to HSV**.

Why HSV?

- **Hue** = color type (Red, Blue, Green, etc.).

- **Saturation** = intensity of the color.
- **Value** = brightness.

HSV makes it easier to **define color ranges** (for example, red 0° in Hue).

1.2.2 Steps for Color Detection

1. Capture video from webcam.
2. Convert frame **BGR → HSV**.
3. Define **lower and upper HSV ranges** for each color.
4. Create a **mask** using `cv2.inRange()` → highlights only that color.
5. Find **contours** of detected areas.
6. Draw **bounding boxes** around them.
7. Use **the same color for bounding box + label**.

2 Implementation

2.1 Import Libraries

- cv2 → OpenCV library for computer vision.
- numpy → handle arrays, ranges, masks.

[ ]:
```
import cv2
import numpy as np
```

2.2 Open Webcam

cv2.VideoCapture(0) opens your **default camera**.

- 0 = built-in camera.
- 1 or 2 = external cameras.

[ ]:
```
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: Could not access the webcam")
```

2.3 Define Color Ranges

We define HSV ranges for multiple colors & shades.

- lower HSV range (minimum color values).
- upper HSV range (maximum color values).

[ ]:
```
color_ranges = {
    "Red": ([0, 120, 70], [10, 255, 255]),
    "Dark Red": ([170, 120, 70], [180, 255, 255]),
    "Green": ([36, 25, 25], [86, 255, 255]),
    "Light Blue": ([90, 50, 50], [110, 255, 255]),
    "Dark Blue": ([111, 84, 46], [131, 255, 255]),
    "Yellow": ([20, 100, 100], [30, 255, 255]),
    "Orange": ([10, 100, 20], [25, 255, 255]),
    "Purple": ([129, 50, 70], [158, 255, 255]),
    "Pink": ([160, 50, 70], [170, 255, 255])
}
```

2.4 Assign BGR Colors for Bounding Boxes

Example: (0,0,255) = Red, (0,255,0) = Green.

[ ]:
```
color_bgr = {
    "Red": (0, 0, 255),
    "Dark Red": (0, 0, 200),
    "Green": (0, 255, 0),
    "Light Blue": (255, 200, 100),
    "Dark Blue": (200, 100, 50),
    "Yellow": (0, 255, 255),
    "Orange": (0, 165, 255),
    "Purple": (128, 0, 128),
    "Pink": (180, 105, 255)
}
```

3 Live Loop for Detection

- Reads one frame at a time.
- Converts it to HSV.

[ ]:
```
while True:
    ret, frame = cap.read()
    if not ret:
        break

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

3.1 Detect Each Color

Steps inside loop:

- cv2.inRange() → binary mask of detected color.
- cv2.findContours() → finds boundaries of detected regions.
- cv2.contourArea() → filters out small noisy detections.
- cv2.boundingRect() → draws a rectangle around detected area.

- cv2.rectangle() → draws bounding box in the same color.
- cv2.putText() → writes detected color name above the box.

[ ]:
```
    for color_name, (lower, upper) in color_ranges.items():
        lower_np = np.array(lower)
        upper_np = np.array(upper)

        # Create mask
        mask = cv2.inRange(hsv, lower_np, upper_np)

        # Find contours
        contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.
↪CHAIN_APPROX_SIMPLE)

        for cnt in contours:
            area = cv2.contourArea(cnt)
            if area > 1000:  # ignore small detections
                x, y, w, h = cv2.boundingRect(cnt)
                cv2.rectangle(frame, (x, y), (x + w, y + h),␣
↪color_bgr[color_name], 3)
                cv2.putText(frame, color_name, (x, y - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7,␣
↪color_bgr[color_name], 2, cv2.LINE_AA)
```

3.2 Show Frame & Exit Key

- Shows only one window (Color Detection).
- Press Esc to close.

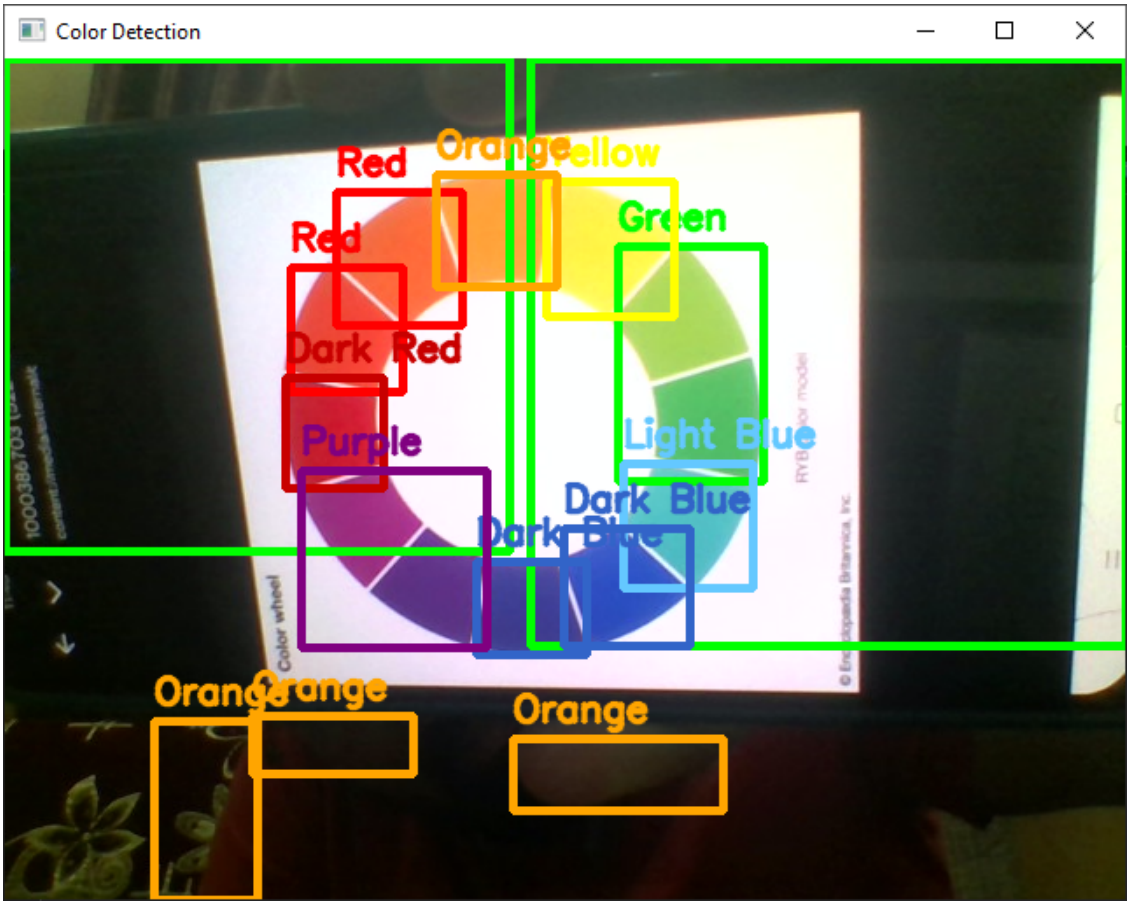
[ ]:
```
    cv2.imshow("Color Detection", frame)

    if cv2.waitKey(1) & 0xFF == 27:  # Esc key
        break

cap.release()
cv2.destroyAllWindows()
```

3.3 Final Output

- Original webcam feed is shown.
- When a color is detected → bounding box in that exact color is drawn.
- Color name appears above the box.
- Works for multiple colors at once.

3.4 Example Output

Below is a sample output screenshot from my project:



This shows how the program detects multiple colors in real-time, draws bounding boxes, and labels them with the correct color names.

4 Key Learnings

- HSV color space is better for detection than BGR.
- Using masks + contours helps locate colored regions.
- Bounding boxes + labels make detection intuitive (like face detection).
- Mapping HSV to BGR makes boxes look natural.
- You can extend this to detect any number of colors by just adding ranges.