**Computer Vision Micro Project Report: Colour-Based Drawing Application**

By- Aarya Kulkarni – 2213066

Mrunali Dhoble – 2213229

Branch and Div- LY-4 AIA-1

**1. Introduction**

**1.1 Project Objective**

The objective of this project is to create a color-based drawing application using Computer Vision techniques. The system allows users to draw on a virtual canvas using a webcam feed that detects a specific colored object. This project employs the OpenCV library to detect a green-colored pointer in the webcam feed, enabling intuitive and real-time drawing on the screen.

**1.2 Motivation**

With the advancement of Computer Vision technology, gesture-based interactions have become an exciting domain to explore. This project aims to demonstrate the potential of real-time color detection to facilitate simple yet effective human-computer interaction without relying on physical touch input. It is particularly useful for virtual whiteboards and interactive applications.

**2. System Design**

**2.1 Tools and Libraries**

* **Programming Language**: Python
* **Library**: OpenCV
* **Hardware**: Standard webcam

**2.2 Concept**

The concept revolves around using a webcam to track a green-colored object. The tracked object acts as a pointer, allowing users to draw lines on a canvas based on its movements. A simple interface with a "Clear" button is provided to reset the canvas.

**2.3 Features**

1. **Real-time Color Tracking**: The system detects and tracks a green-colored object using HSV color space for color filtering.
2. **Drawing on Canvas**: The tracked object movements are visualized as lines on a virtual canvas.
3. **Dynamic Color Adjustments**: Adjustable trackbars enable dynamic setting changes for fine-tuning color detection.
4. **Clear Functionality**: A "Clear" button is provided to erase the entire drawing.

**3. Implementation**

**3.1 Color Detection Using HSV**

Color detection is achieved using the HSV (Hue, Saturation, Value) color space:

* **Hue** is responsible for distinguishing color.
* **Saturation** measures the intensity or purity of the color.
* **Value** measures the brightness.

The specific color values for detecting the green object are controlled via trackbars:

python

Copy code

cv2.createTrackbar("Upper Hue", "Color detectors", 80, 180, setValues)

cv2.createTrackbar("Upper Saturation", "Color detectors", 255, 255, setValues)

cv2.createTrackbar("Upper Value", "Color detectors", 255, 255, setValues)

cv2.createTrackbar("Lower Hue", "Color detectors", 40, 180, setValues)

cv2.createTrackbar("Lower Saturation", "Color detectors", 40, 255, setValues)

cv2.createTrackbar("Lower Value", "Color detectors", 40, 255, setValues)

**3.2 Object Detection and Tracking**

To detect and track the green pointer, the captured frame is first converted to the HSV color space. A binary mask is created by setting the specified color range:

python

Copy code

Mask = cv2.inRange(hsv, Lower\_hsv, Upper\_hsv)

Mask = cv2.erode(Mask, kernel, iterations=1)

Mask = cv2.morphologyEx(Mask, cv2.MORPH\_OPEN, kernel)

Mask = cv2.dilate(Mask, kernel, iterations=1)

Contours are extracted from the mask, and the largest contour is used to determine the pointer's location:

python

Copy code

cnts, \_ = cv2.findContours(Mask.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

**3.3 Drawing on Virtual Canvas**

The pointer's movement is tracked and mapped onto a virtual canvas using a series of deque data structures to store each point’s location. The cv2.line function is utilized to draw lines between consecutive points:

python

Copy code

for j in range(len(gpoints)):

for k in range(1, len(gpoints[j])):

if gpoints[j][k - 1] is None or gpoints[j][k] is None:

continue

cv2.line(frame, gpoints[j][k - 1], gpoints[j][k], color, 2)

cv2.line(paintWindow, gpoints[j][k - 1], gpoints[j][k], color, 2)

**3.4 Clear Canvas Functionality**

A "Clear" button is added at the top of the screen. If the detected pointer position intersects with this button’s area, all stored drawing points are reset:

python

Copy code

if center[1] <= 65:

if 40 <= center[0] <= 140: # Clear Button

gpoints = [deque(maxlen=512)]

green\_index = 0

paintWindow[67:, :, :] = 255

**4. Results**

**4.1 Performance**

The system successfully detects and tracks the green object, allowing smooth drawing on a virtual canvas. The interface for clearing the canvas functions effectively, enhancing user experience. The HSV trackbars provide flexibility for adjusting to different lighting conditions.

**4.2 Limitations**

* **Color Sensitivity**: The system can be affected by background colors similar to the detection range.
* **Lighting**: Proper lighting is crucial for accurate detection.
* **Object Size**: Very small or large objects can impact tracking accuracy.

**5. Applications**

* Virtual Whiteboards for online teaching and remote presentations.
* Interactive Art for engaging installations in galleries.
* Gesture-Based Controls for touch-free device interaction.
* Rehabilitation Therapy to assist motor skills recovery.
* Educational Games for kids to combine learning with fun.
* Collaborative Drawing for team projects and interactive classrooms.
* Touchless Presentations to highlight and navigate slides hands-free.
* Digital Signatures for secure, touchless authentication.
* AR Prototyping for augmented reality drawing and gaming.
* Creative Digital Art for abstract and interactive painting.

**6. Conclusion**

The project successfully demonstrates the capability of using Computer Vision for intuitive, real-time drawing applications. The color-based detection method provides a non-invasive and natural way to interact with the computer. Future improvements can involve multi-color detection, integration of gesture-based commands, and refining accuracy under different lighting conditions.

**7. Future Enhancements**

1. **Multi-Color Support**: Adding support for multiple colors.
2. **Gesture Commands**: Detect gestures like hand waves for more commands (e.g., changing colors).
3. **Improved Accuracy**: Use machine learning-based techniques for more robust object tracking.
4. **Save and Load Canvas**: Provide options to save and load drawings.

**8. References**

* OpenCV Documentation: https://docs.opencv.org
* HSV Color Space: Understanding its application for color detection.
* Python Official Documentation: <https://python.org>