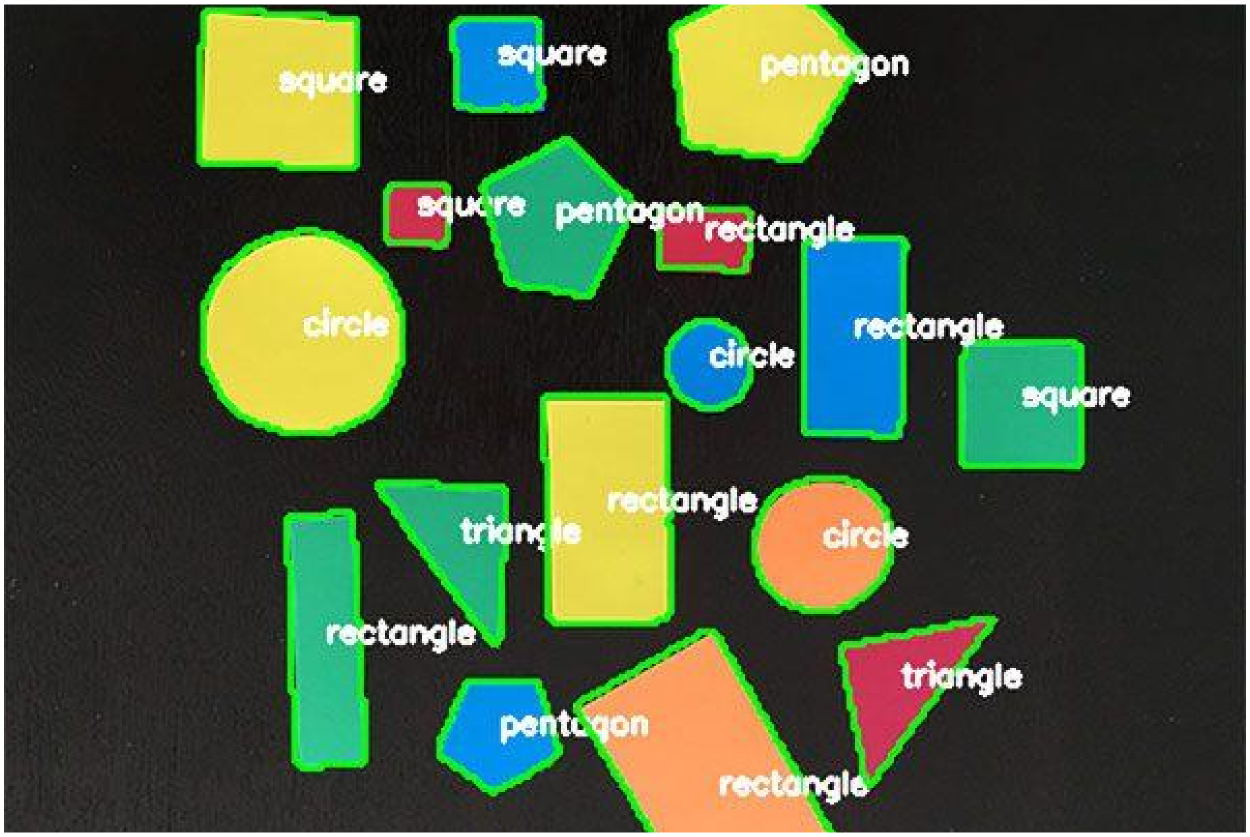

Shaped Contours Detection

Overview

This Python application captures live video from a camera, filters a specific color range, detects contours of specified shapes (rectangles, circles, and squares), and displays the original frame with drawn contours and shape labels.

Goals:

1. Color Filtering: Allows the user to specify a color range in HSV format for contour detection.

2. Contour Detection: Applies color filtering to the live video feed and detects contours using the Canny edge detection method.

3. Shape Classification: Classifies detected contours into shapes (Triangle, Square, Rectangle, Circle).

4. Efficiency Improvements: Optimizations are implemented to skip small contours for better performance.

Usage:

1. Run the Script:

   - Execute the script in a Python environment with OpenCV installed.

2. Color Configuration:

   - Replace the `lower_color` and `upper_color` values with the desired HSV range for the color you want to detect.

3. View Results:

   - The script will display the live video feed with detected contours and labeled shapes.

4. Exit:

   - Press the 'ESC' key to exit the script.

Dependencies:

OpenCV: Ensure that the OpenCV library is installed. You can install it using “pip install opencv-python”.

Notes:

1 - Fine-tune the color range (“lower_color” and “upper_color”) based on the specific color you want to detect, Here more details:

To change the color you want to detect , you need to specify the new color range in HSV (Hue, Saturation, Value) format. You can use an online HSV color picker tool to find the appropriate values for the color you're interested in.

Replace the lower_color and upper_color values with the HSV range of your desired color. The HSV values are usually represented in the range (0-179) for Hue, (0-255) for Saturation, and (0-255) for Value.

Example Color Configuration (for detecting red):

```
# Define the color range for detection (replace these values with the desired color)
lower_color = np.array([0, 100, 100])  # Lower HSV range for red
upper_color = np.array([10, 255, 255])  # Upper HSV range for red
```

2 - Adjust the minimum contour area threshold (“cv2.contourArea(contour) < 100”) based on the environment and lighting conditions, more details:

The minimum contour area threshold is a parameter that is used to filter out small contours during the shape detection process. Contours are regions with a continuous boundary, and their area is a measure of their size. By setting a minimum contour area threshold, you can exclude very small contours from being considered during the shape classification step.

For example : “if cv2.contourArea(contour) < 100:

continue”

This line ensures that contours with an area less than 100 pixels are skipped and not processed further. The specific value of 100 is a threshold that you can adjust based on your specific requirements and the characteristics of the input video feed.

Here's why you might adjust the minimum contour area threshold:

1- Filtering Noise: Small contours might represent noise or insignificant details in the video feed. By setting a minimum area threshold, you filter out these small contours, reducing the impact of noise on the shape detection process.

2- Improving Performance: Processing very small contours might be computationally expensive and unnecessary for shape classification. Skipping small contours can improve the efficiency of the shape detection algorithm.

3- Adapting to Environment: The appropriate threshold value depends on the characteristics of the input video feed, lighting conditions, and the size of the objects you want to detect. You may need to experiment and adjust this value based on your specific use case to find the best threshold value to find a balance between filtering small contours and retaining meaningful shape information based on your application's needs.