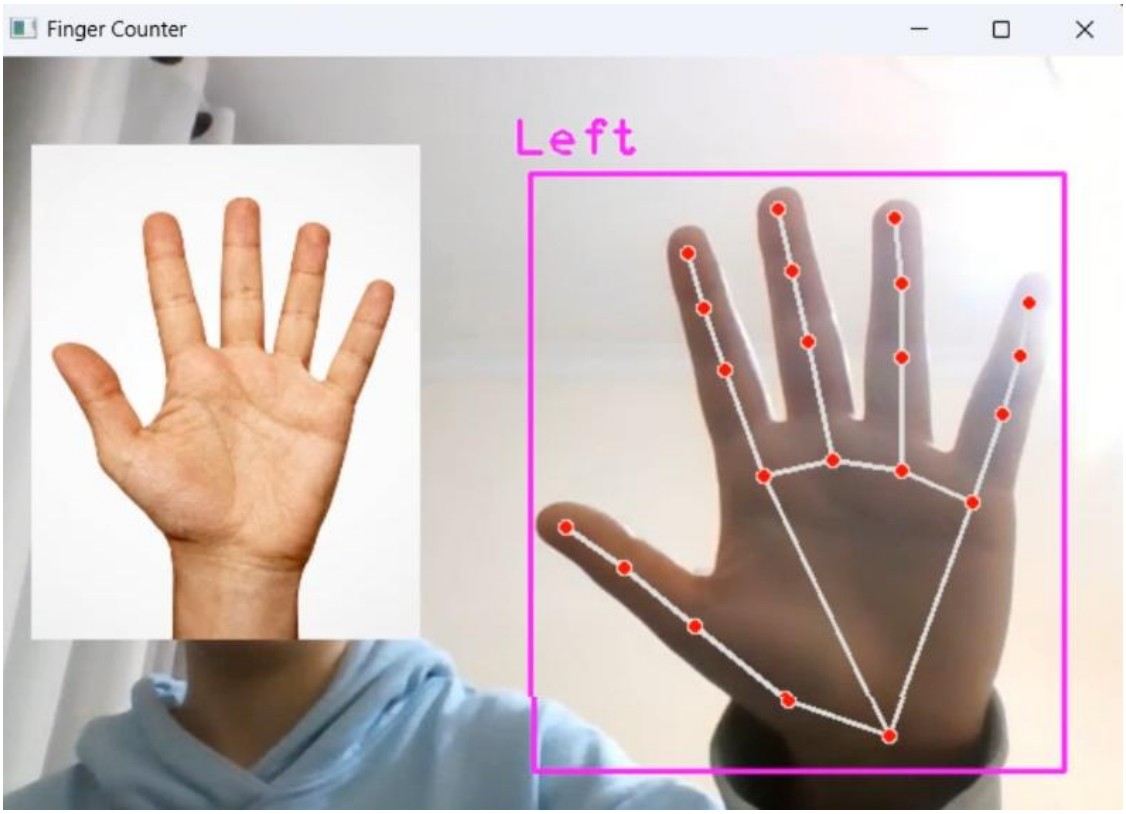
Finger Counter using Computer Vision and OpenCv in Python



Overview

This project is a Python-based application that uses OpenCV and the cvzone library to detect and count the number of fingers shown to the camera. The program overlays an image corresponding to the detected number of fingers on the video feed in real-time.

Required Libraries

To run this project, you need to install the following Python libraries:

```
pip install opencv-python cvzone
```

Explanation of Code

1. Importing Libraries

```
import cv2
import os
from cvzone.HandTrackingModule import HandDetector
```

cv2: OpenCV library for video processing.

os: Used for file path management and verifying the existence of files.

HandDetector: A class from cvzone for detecting hands and fingers in real-time.

2. Initializing Hand Detector

```
# Initialize HandDetector
detector = HandDetector(maxHands=1, detectionCon=0.8)
```

maxHands=1: Detects only one hand at a time.

detectionCon=0.8: Sets the confidence threshold for hand detection.

3. Setting Default Image Path

```
# Default image path
default_image_path = r"C:\Users\abedi\OneDrive\Desktop\OOP Prj\pic\zero.jpeg"
if not os.path.exists(default_image_path):
    print(f"Error: Default image not found at {default_image_path}. Exiting.")
    exit()
```

Checks if the default image for zero fingers exists. If not, the program exits.

4. Starting Video Capture

```
# Start video capture with the default camera (0)
video = cv2.VideoCapture(0)
```

Opens the default camera for video capture. The argument 0 refers to the primary camera.

5. Main Loop

```
while True:
    success, img = video.read()
    if not success:
        print("Error: Unable to access the camera.")
        break

    img = cv2.flip(img, 1)  # Mirror the camera feed for natural interaction
```

- Reads frames from the camera. If reading fails, the program exits the loop.

- Mirrors the video feed for natural interaction.

6. Detecting Hands

```
# Detect hands in the image
hands, img = detector.findHands(img, draw=True)
```

Detects hands in the video frame and draws hand landmarks on the image if detected.

7. Loading and Displaying Finger Count Image

```
# Load default image
fing = cv2.imread(default_image_path)

if hands:
    hand = hands[0]
    lmList = hand['lmList']  # List of landmarks
    label = "Right" if hand['type'] == "Left" else "Left"  # Fix mirrored logic

    # Detect the fingers that are up
    fingerup = detector.fingersUp(hand)
    print(f"Fingers detected: {fingerup}")
```

If a hand is detected:

- lmList: List of landmarks used for hand analysis.

- hand['type']: Identifies whether the hand is "Left" or "Right" (mirrored logic is applied).

- fingersUp: Determines which fingers are extended.

8. Mapping Finger Count to Images

```
# Image paths for finger count
image_paths = {
    "[0, 1, 0, 0, 0]": r"C:\Users\abedi\OneDrive\Desktop\OOP Prj\pic\one.jpg",
    "[0, 1, 1, 0, 0]": r"C:\Users\abedi\OneDrive\Desktop\OOP Prj\pic\two.jpg",
    "[0, 1, 1, 1, 0]": r"C:\Users\abedi\OneDrive\Desktop\OOP Prj\pic\three.jpg",
    "[0, 1, 1, 1, 1]": r"C:\Users\abedi\OneDrive\Desktop\OOP Prj\pic\four.jpg",
    "[1, 1, 1, 1, 1]": r"C:\Users\abedi\OneDrive\Desktop\OOP Prj\pic\five.jpg",
}

fing_path = image_paths.get(str(fingerup), default_image_path)
fing = cv2.imread(fing_path)
```

- Maps the detected finger configuration to an image path.

- Loads the corresponding image to be displayed.

9. Overlaying the Image

```
# Overlay the finger image on the video feed
if fing is not None and img.shape[0] > 330 and img.shape[1] > 240:
    img[50:330, 20:240] = fing
else:
    print("Warning: Image or video feed size mismatch.")
```

- Resizes the finger count image and overlays it on the video feed at a specific position.

10. Displaying the Video Feed

```
# Display the video feed with the overlay
cv2.imshow("Finger Counter", img)

# Exit on 'q' key press
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

- Shows the modified video feed in a window titled "Finger Counter."

- Exits the program when the 'q' key is pressed.

11. Releasing Resources

```
# Release resources
video.release()
cv2.destroyAllWindows()
```

- Frees the video capture and closes all OpenCV windows.

Algorithm for Finger Detection

1. **Hand Detection**:

   The HandDetector uses a pre-trained model to identify hand landmarks.

2. **Landmark Analysis**:

   Landmarks are analyzed to determine which fingers are extended using geometric relationships between joints.

3. **Finger Configuration**:

   fingersUp: Returns a list (e.g., [0, 1, 1, 0, 0]) where 1 means the finger is extended, and 0 means it is not.

4. **Mapping**:

   The configuration list is mapped to a corresponding image of the finger count.

5. **Overlay**:

   The selected image is resized and displayed on the video feed.