Hand Gesture Recognition

Real-time Open/Closed Hand Detection using OpenCV + HSV + Convexity Defects + Tkinter GUI

📌 Overview

This project performs **real-time hand gesture recognition** (Open / Closed hand) using:

- HSV Skin Detection
- Morphological Filtering
- Contours + Convex Hull
- Convexity Defects
- Tkinter GUI for live video display

All processing is done inside a defined ROI to reduce noise and increase stability.

🧩 Features

- Real-time gesture detection
- No AI or machine learning required
- Lightweight and fast
- GUI built using Tkinter
- Accurate detection using convexity defects
- Easy to add your own actions (`action_on` / `action_off`)

---

1. Camera Setup & Frame Capture

The camera is opened using:

```
cap = cv2.VideoCapture(CAM_INDEX)
```

Frames are processed every ~30ms through:

```
update_frame()
```

with a safety check:

```
if not ret or frame is None:
    return
```

---

2. ROI (Region of Interest)

A fixed ROI is used to stabilize detection and reduce background noise.

- **Width**: half of the frame
- **Height**: 4/5 of the frame

```
work_frame = frame[y1:y2, x1:x2]
```

---

3. Convert Frame to HSV

HSV is used because it separates lightness from color, making skin detection stable.

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

---

4. Skin Detection via HSV Threshold

A predefined skin color range is used:

```
mask = cv2.inRange(hsv, (0, 30, 60), (20, 150, 255))
```

- White = skin
- Black = background

---

5. Mask Cleaning (Noise Removal)

Multiple filters are applied for better contour extraction:

```
mask = cv2.GaussianBlur(mask, (7, 7), 0)
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
mask = cv2.dilate(mask, kernel)
```

---

6. Contour Extraction

Skin areas are detected using:

```
contours, _ = cv2.findContours(mask, ...)
```

The largest valid contour is considered the hand:

```
if area >= MIN_CONTOUR_AREA:
    max_contour = cnt
```

---

7. Convex Hull

A convex hull is drawn around the hand contour:

```
hull_points = cv2.convexHull(max_contour)
```

This creates a tight outer boundary around the hand.

---

8. Convexity Defects

Convexity defects represent the gaps between fingers.

```
hull_idx = cv2.convexHull(max_contour, returnPoints=False)
defects = cv2.convexityDefects(max_contour, hull_idx)
```

Each defect corresponds to a possible finger gap.

9. Finger Gap Processing

A defect is considered valid if:

- The angle is small (finger-like)
- The depth is sufficiently large

```
if ang < DEFECT_ANGLE_THRESH and depth > area * DEFECT_DEPTH_RATIO:
    fingers_defects += 1
```

---

10. Hand State Detection (Open/Closed)

A simple rule:

```
hand_open = fingers_defects >= 2
```

- `>= 2` → Open Hand
- `< 2` → Closed Hand

Displayed on screen:

```
cv2.putText(frame, f"Hand: {final_state}", ...)
```

---

11. Actions (Customize)

You can run any function depending on the gesture:

```
def action_on(frame):
    pass

def action_off(frame):
    pass
```

These are triggered when the hand state changes.

---

12. GUI (Tkinter)

Tkinter is used to display frames inside a window:

```
video_label = tk.Label(root)
```

Frames are converted to PIL:

```
im_pil = Image.fromarray(img_rgb)
imgtk = ImageTk.PhotoImage(image=im_pil)
video_label.configure(image=imgtk)
```

📎 Additional Notes

- Optional preview windows show both the mask and the original frame.
- All drawing (contours, hull, defects) is overlaid on the main frame for easier debugging.
- Code is modular and easy to extend.

🏁 How to Run

```
python Hand_Gesture_Recognition.py
```

📜 License

Ziad Ahmed Shalaby License