Singapore Institute of Technology

BEng (Hons) Information and Communications Technology majoring in Software Engineering

INF2009 Edge Computing and Analytics

Academic Year 2024/2025 Trimester 2

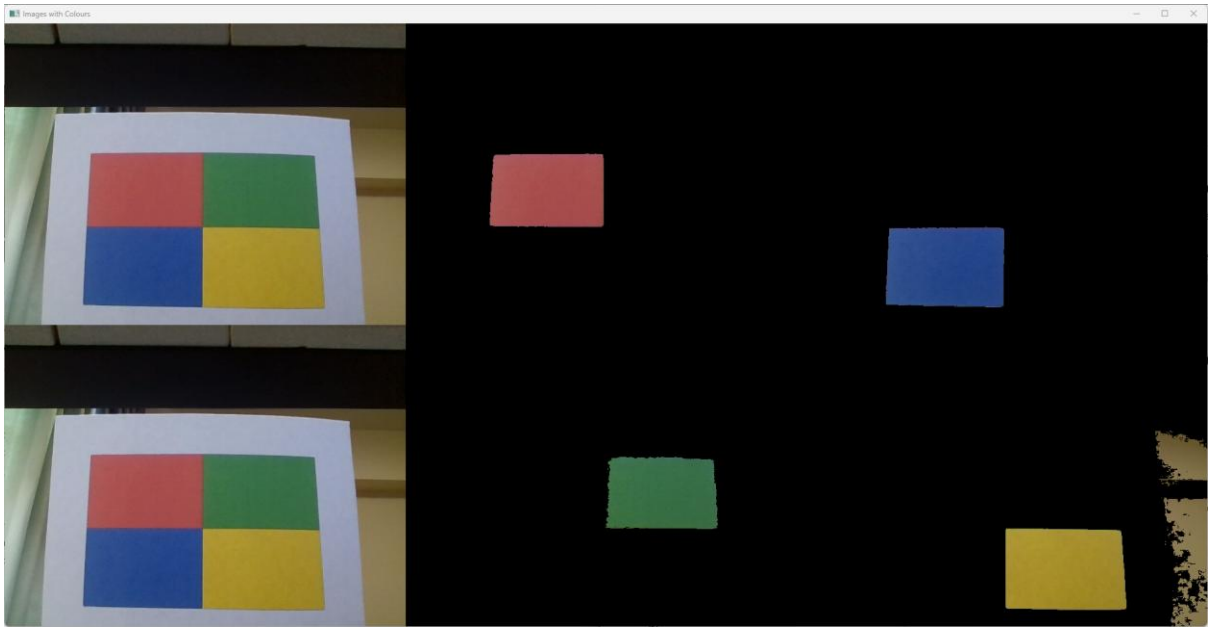
Week 3 & 4 Lab - Image and Video Analytics

Name: Lim Chee Hean

Student ID: 2201529

Image Analytics

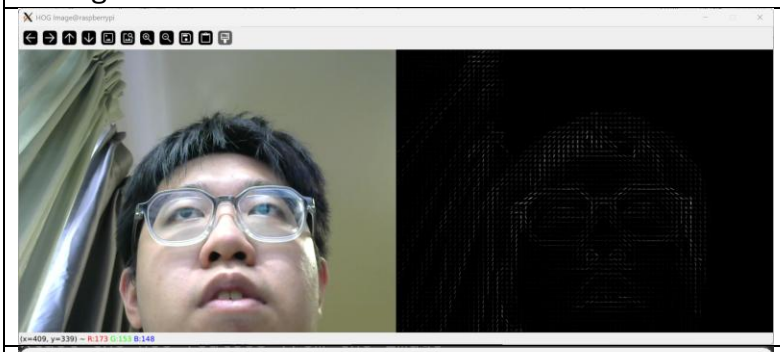
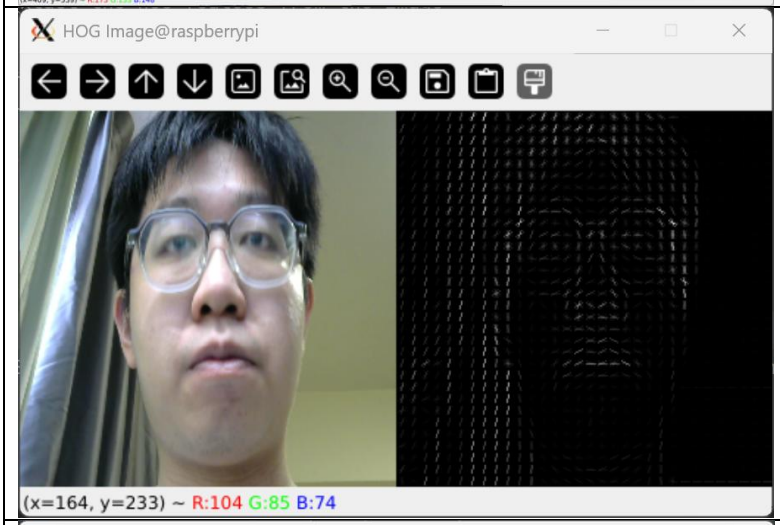
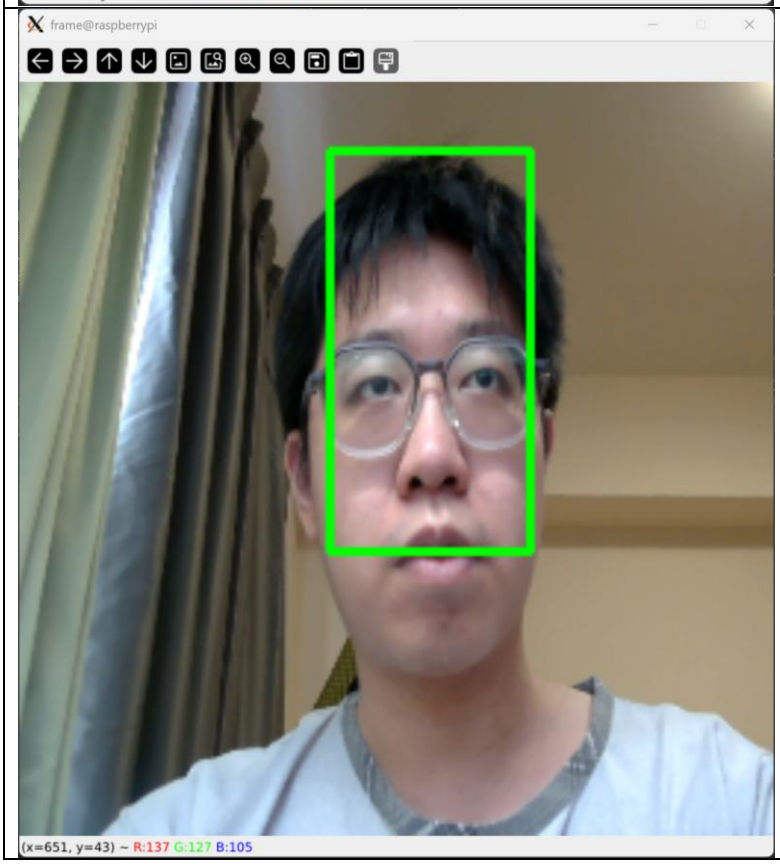
4. Introduction to Real-time Image Processing with Python



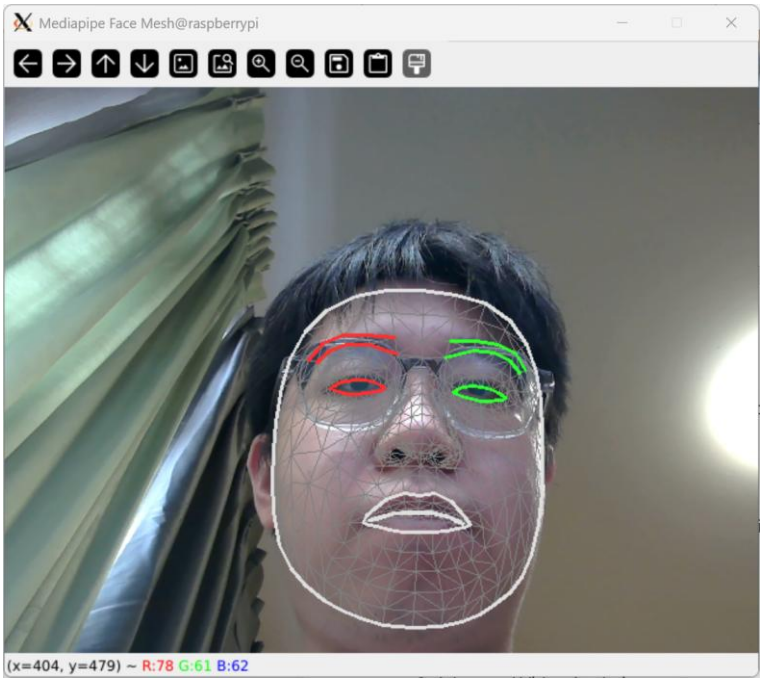
```
# Defining a list of boundaries in the HSV space
 # H (hue) defines the position of the color in the range of 0 to 180
 # S (saturation) defines the intensity of the color
# V (value) defines the brightness of the color
boundaries = {
    "red": [(0, 100, 100), (10, 255, 255)],        # Lower red
    "red2": [(160, 100, 100), (180, 255, 255)],    # Upper red (wrap-around)
    "blue": [(100, 100, 100), (140, 255, 255)],  # Blue range
    "green": [(40, 100, 100), (90, 255, 255)],     # Green range
    "yellow": [(20, 100, 100), (35, 255, 255)]   # Yellow range
}
```

HSV values were used instead to provide a more accurate classification by Hue, which ranges from 0 to 180 degrees.

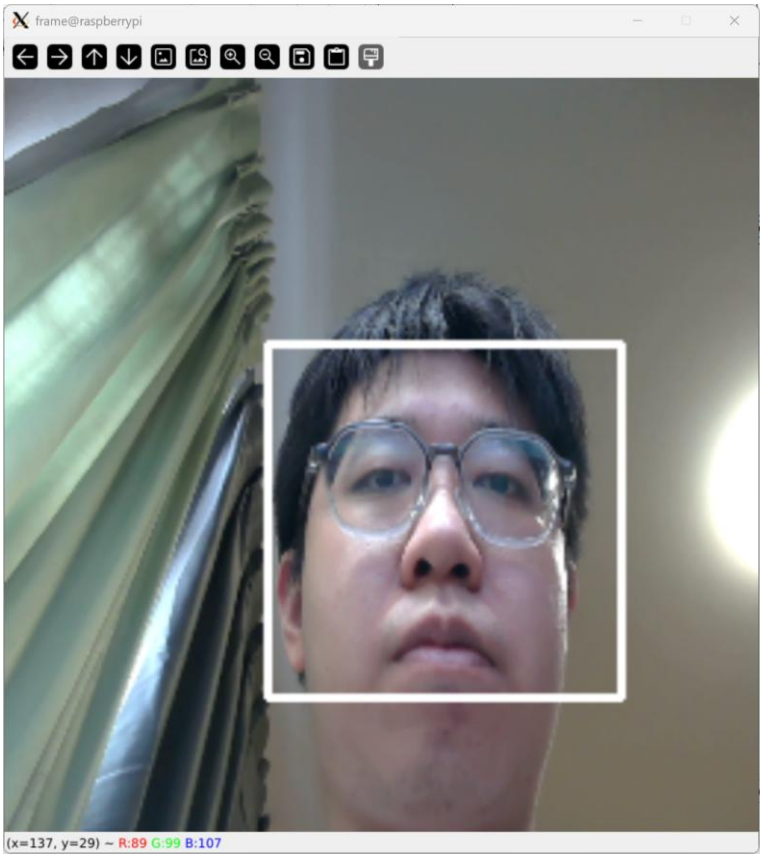
5. Real-time Image Analysis

Histogram of Gradients	
	Before resizing (poor frame rate)
	After resizing to 256 by 256 pixels (improved frame rate)
	Face detection

6. Real-time Image Feature Analysis for Face Capture and Facial Landmark Extraction



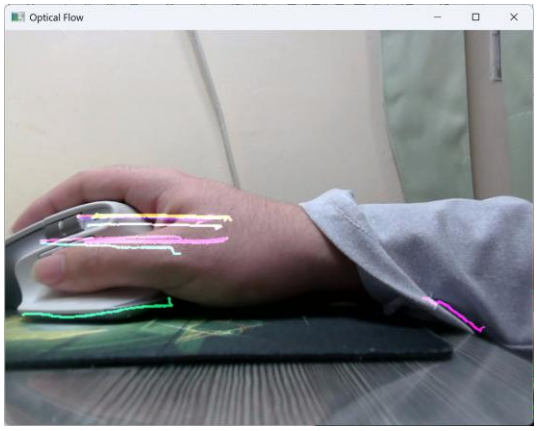
Face detection with mediapipe



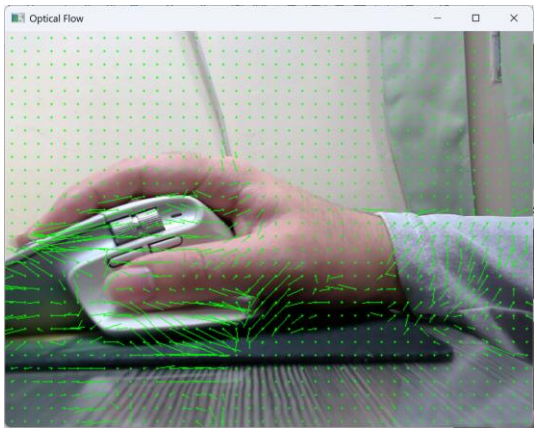
Face detection with OpenCV

Video Analytics

4. Introduction to Real-time Video Processing on Raspberry Pi



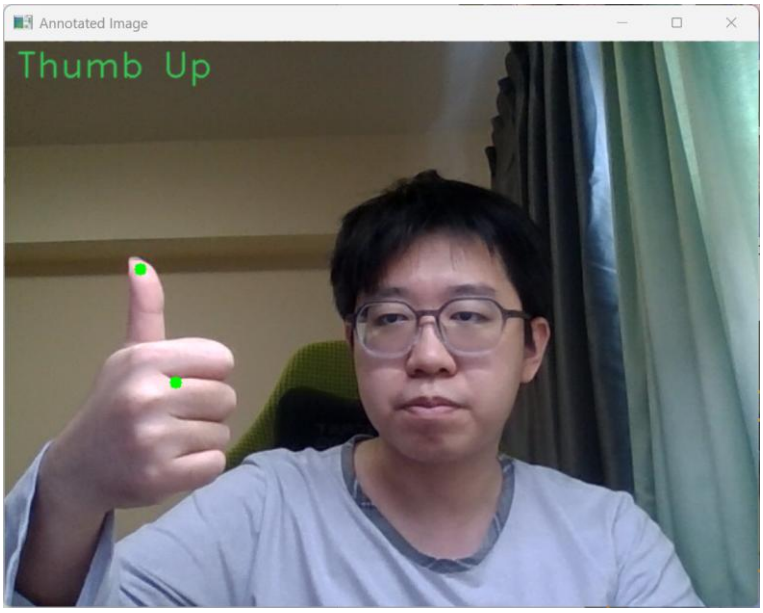
Lucas Kande Optical Flow



Dense Optical Flow by Lines

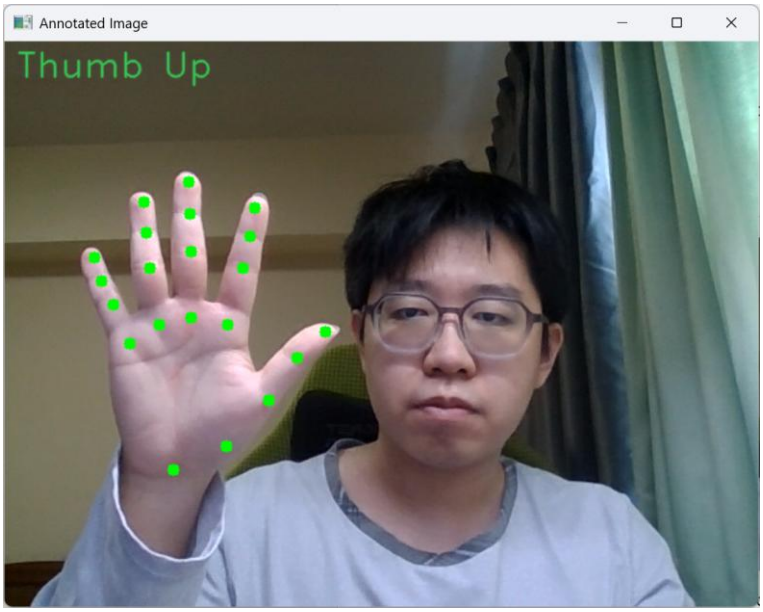
Parameter	Observation/Conclusion
maxCorners	Maximum number of points to track.
qualityLevel	The minimum quality of a point being considered a corner.
minDistance	Minimum distance between corners to prevent tracking points close to each other.
blockSize	Window size to perform corner detection.
winSize	The size of the area to track movement. A larger value can track the movement of larger objects.
maxLevel	Tracks points across different zoom levels.
criteria	The criteria to stop tracking movement, where 10 is the maximum number of iterations to track, and 0.03 is the minimum change between iterations.

5. Advanced Video Analytics

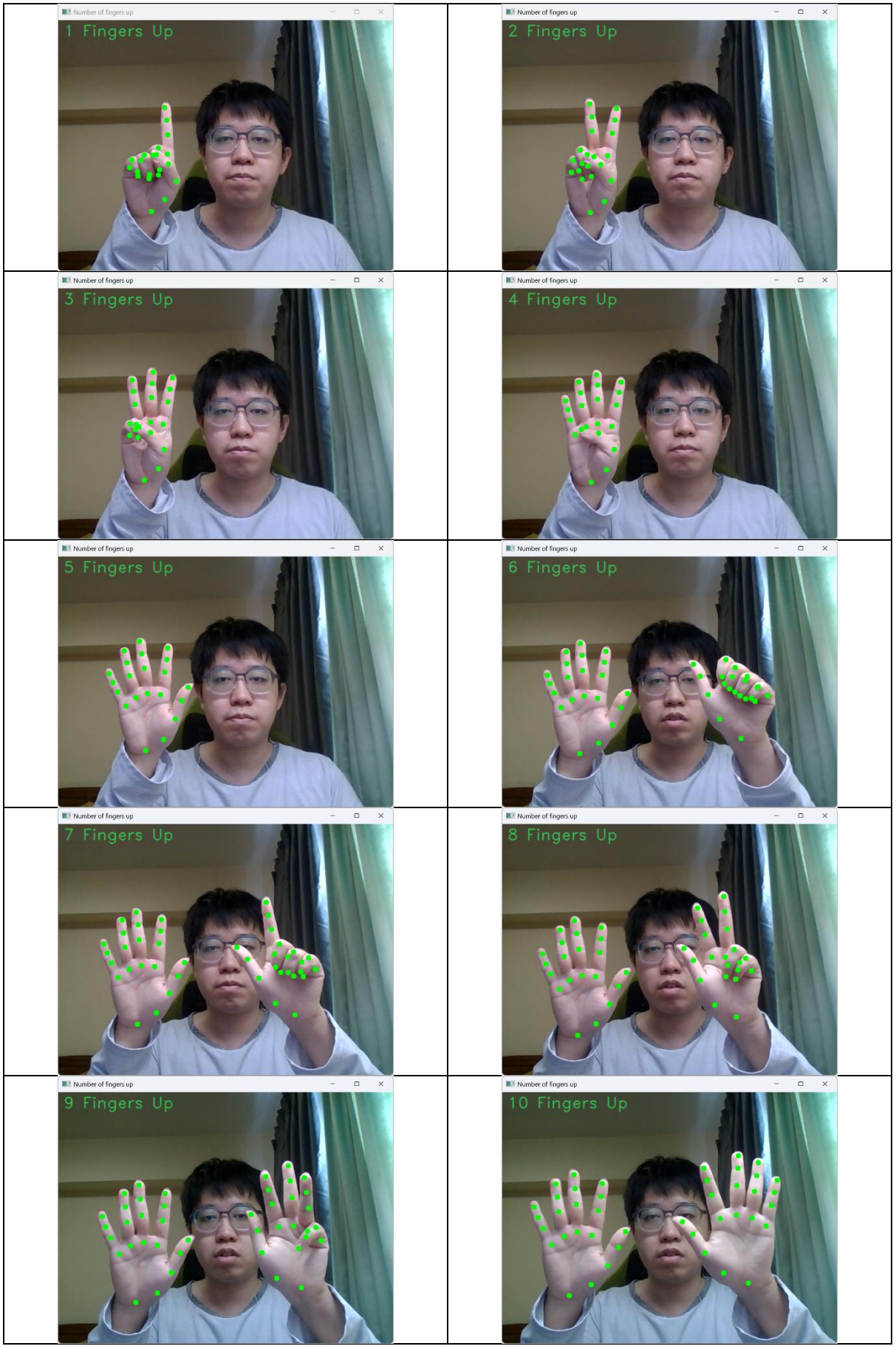


Detecting Thumbs Up

```
for _ in range(21):
    print(hand_landmarks)
    x = int(hand_landmarks[_].x * frame.shape[1])
    y = int(hand_landmarks[_].y * frame.shape[0])
    cv2.circle(frame, center: (x, y), radius: 5, color: (0, 255, 0), -1)
```



Show all 21 fingerpoints



Show number of fingers raised

```python
# y increases downwards!!!
# Check position of THUMB_CMC and THUMB_IP relative to WRIST
# If THUMB_IP is closer and lower, then finger is up
wrist, thumb_cmc, thumb_tip = hand_landmarks[0], hand_landmarks[1], hand_landmarks[4]
if abs(wrist.x - thumb_cmc.x) < abs(wrist.x - thumb_tip.x) and thumb_cmc.y > thumb_tip.y:
    fingers_up += 1

# Loop through the other fingers
for _ in range(5, 21, 4):
    finger_up = 1
    # Check if each landmark is lower than previous
    for landmark in range(_ + 1, _ + 4):
        # Finger down if landmark is higher than previous
        if hand_landmarks[landmark].y > hand_landmarks[landmark - 1].y:
            finger_up = 0
    fingers_up += finger_up
```

Implementation for counting number of fingers raised.

For thumb:

The thumb is different of other fingers such that the tip is always higher than the other landmarks even when the thumb is not raised (imagine making a fist or the number 4).

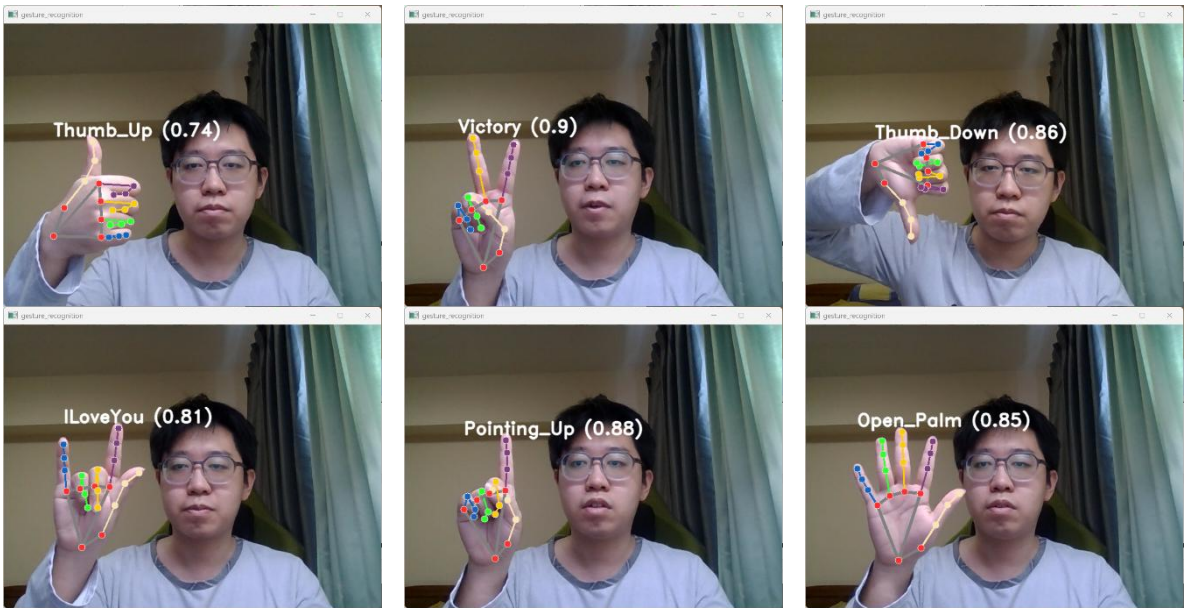
Hence, the horizontal distance from the thumb base (THUMB_CMC) and the thumb tip (THUMB_TIP) to the wrist are compared. If the thumb base is closer to the wrist, then the finger is considered down and vice versa.

To prevent the thumbs down gesture from being recognised as finger up, the thumb base is also ensured to be lower than the thumb tip.

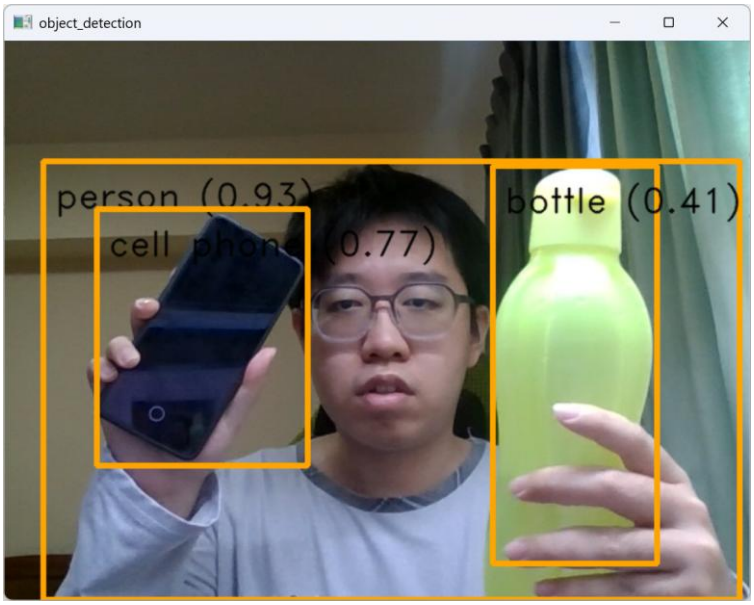
For other fingers:

If all landmarks on the same finger are higher than the previous index (based on the finger model), then the finger is considered up.

6. Advanced Video Analytics



Gesture Recognition



Object Detection

```python
import cv2
from cv2 import VideoCapture, VideoWriter, VideoWriter_fourcc, CAP_PROP_FPS, CAP_PROP_FRAME_WIDTH, CAP_PROP_FRAME_HEIGHT
from mediapipe.tasks import python
from mediapipe.tasks.python import vision
import mediapipe as mp

# Setup detector
base_options = python.BaseOptions(model_asset_path='efficientdet.tflite')
options = vision.ObjectDetectorOptions(base_options=base_options,
                                       score_threshold=0.25)
detector = vision.ObjectDetector.create_from_options(options)

# Extract frames
frames = []
capture = VideoCapture("original.mp4")
fps, width, height = capture.get(CAP_PROP_FPS), int(capture.get(CAP_PROP_FRAME_WIDTH)), int(capture.get(CAP_PROP_FRAME_HEIGHT))
success, frame = capture.read()
while success:
    frames.append(frame)
    success, frame = capture.read()
capture.release()

writer = VideoWriter("summary_.avi", VideoWriter_fourcc(*'XVID'), fps, (width, height))

for _, frame in enumerate(frames):
    print(f"\rProcessing frame {_} of {len(frames)}", end="")

    # Perform detection
    rgb_image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=rgb_image)
    categories = [category[0].category_name for category in [detection.categories for detection in detector.detect(mp_image).detections]]

    # Save frames with cell phone
    if "cell phone" in categories:
        print("\rCell Phone detected", end="")
        writer.write(frame)

writer.release()
```

Video Summarisation by Object Detection

The program takes in an input video, performs object detection for every frame, and then only saves the frame with cell phone detected.

Original vs Summarised Video Length

Original Video Link: https://youtu.be/ssUXZEsEo3w

Summarised Video Link: https://youtu.be/UJSnI6E95TE