MediaPipe Development

mediapipe github：<u>https://github.com/google/mediapipe</u>
mediapipe：<u>https://google.github.io/mediapipe/</u>
dlib：<u>http://dlib.net/</u>
dlib github：<u>https://github.com/davisking/dlib</u>

1.Introduction
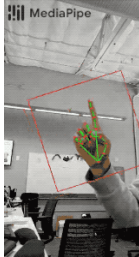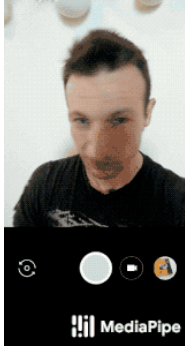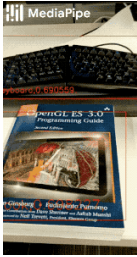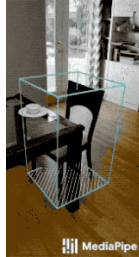
MediaPipe is a data stream processing machine learning application development framework developed and open source by Google. It is a graph based data processing pipeline used to construct data sources that utilize various forms, such as video, audio, sensor data, and any time series data. MediaPipe is cross platform and can run on embedded platforms (such as Raspberry Pi), mobile devices (iOS and Android), workstations, and servers, and supports mobile GPU acceleration. MediaPipe provides cross platform, customizable ML solutions for real-time and streaming media.

The core framework of MediaPipe is implemented in C++and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include Packets, Streams, Calculators, Graphs, and Subgraphs.

Characteristics of MediaPipe:

- End-to-end acceleration: The built-in fast ML inference and processing can be accelerated even on ordinary hardware.
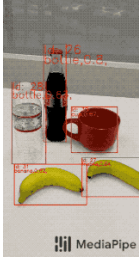- Build once, deploy anytime, anywhere: A unified solution suitable for Android, iOS, desktop/cloud, web, and IoT.
- Ready to use solution: A cutting-edge ML solution that showcases the full functionality of the framework.
- Free open source: Framework and solutions under Apache 2.0, fully scalable and customizable.

Deep learning solutions in MediaPipe

Face Detection	Face Mesh	Iris	Hands	Pose	Holistic

Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT

	Android	iOS	C++	Python	JS	Coral
Face Detection	✅	✅	✅	✅	✅	✅
Face Mesh	✅	✅	✅	✅	✅	
Iris	✅	✅	✅			
Hands	✅	✅	✅	✅	✅	
Pose	✅	✅	✅	✅	✅	
Holistic	✅	✅	✅	✅	✅	
Selfie Segmentation	✅	✅	✅	✅	✅	
Hair Segmentation	✅		✅			
Object Detection	✅	✅	✅			✅
Box Tracking	✅	✅	✅			
Instant Motion Tracking	✅					
Objectron	✅		✅	✅	✅	
KNIFT	✅					
AutoFlip			✅			
MediaSequence			✅			
YouTube 8M			✅			

2.Use

This only demonstrates the case of py files.

```
cd ~/UARTDemo/scripts
python3 06_FaceLandmarks.py
python3 07_FaceDetection.py
python3 08_Objectron.py
python3 09_VirtualPaint.py
python3 10_HandCtrl.py
python3 11_GestureRecognition.py
```

During use, attention should be paid to the following

- Hand detection, posture detection, overall detection, and facial detection all have point cloud viewing functions, taking facial detection as an example.
- All functions [q key] are set to exit.
- Overall detection: including hand, face, and body posture detection.
- 3D object recognition: recognizable objects include: ['Shoe ',' Chair ',' Cup ',' Camera '], a total of 4 categories; Click the 'f' button to switch to recognizing objects; The Jetson series does not use keyboard buttons, and switching to recognize objects requires changing the [self. index] parameter in the source code.

- Brush: When the index and middle fingers of the right hand merge, it is in the selection state, and a color selection box pops up. When the two fingertips move to the corresponding color position, select the color (black is the eraser); The index and middle fingers are in a drawing state and can be drawn on the drawing board at will.
- Finger control: Click the 'f' button to switch the recognition effect.
- Finger recognition: gesture recognition designed based on the right hand can be accurately recognized when certain conditions are met. The recognizable gestures include: [Zero, One, Two, Three, Four, Five, Six, Seven, Eight, Okay, Rock, Thumb_up (like), Thumb_down (thumb down), Heart_single (one handed comparison)], a total of 14 categories.
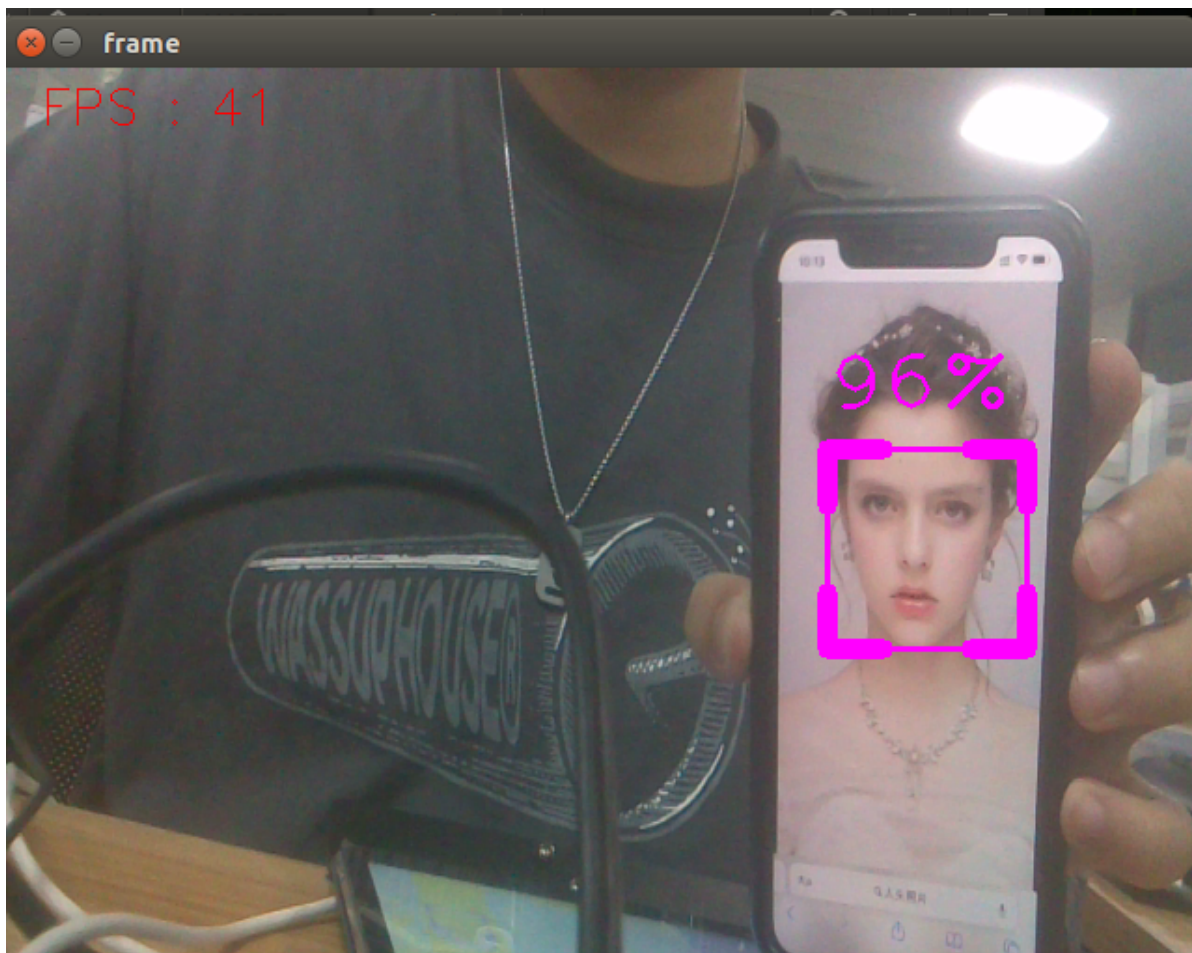
Program operation:

```
python3 06_FaceLandmarks.py
```
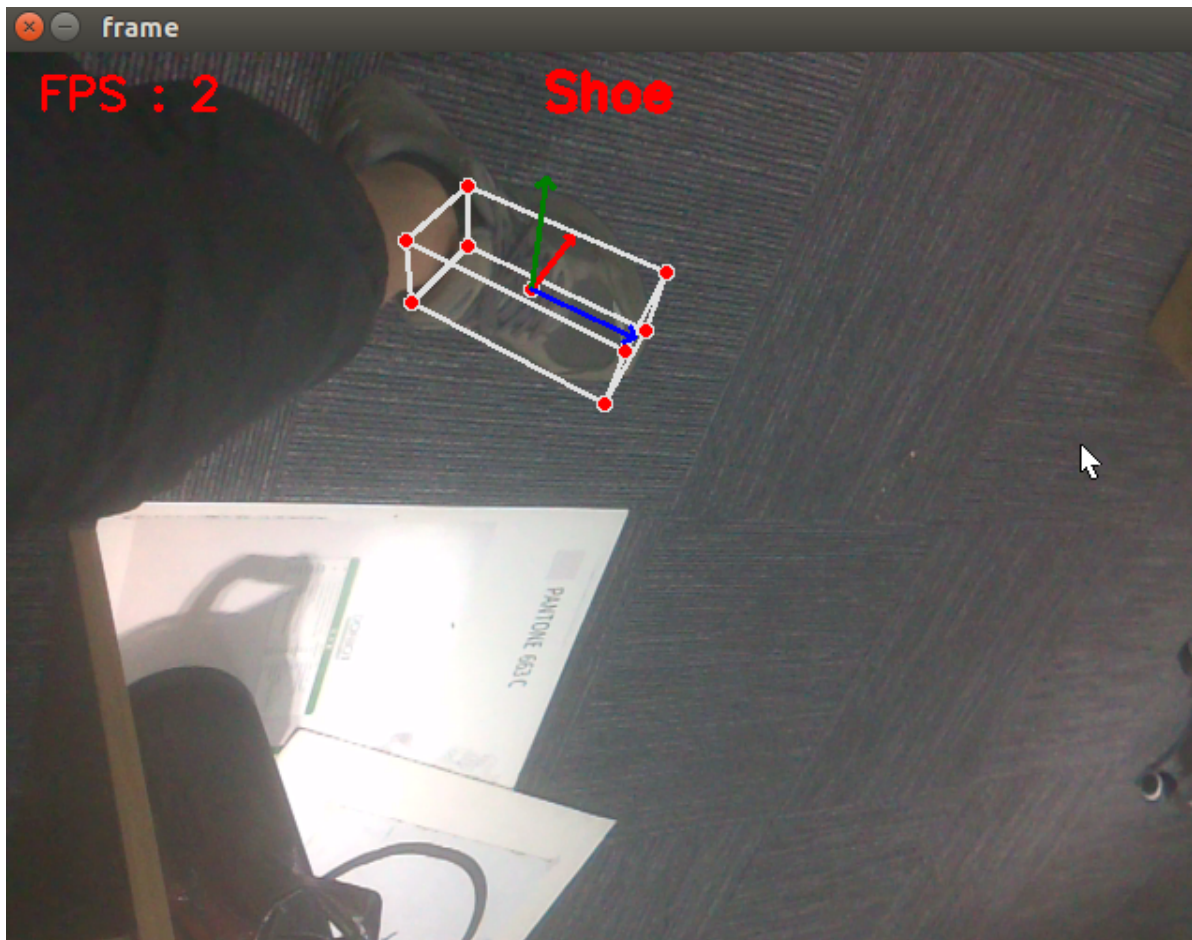
design sketch:


```
python3 07_FaceDetection.py
```
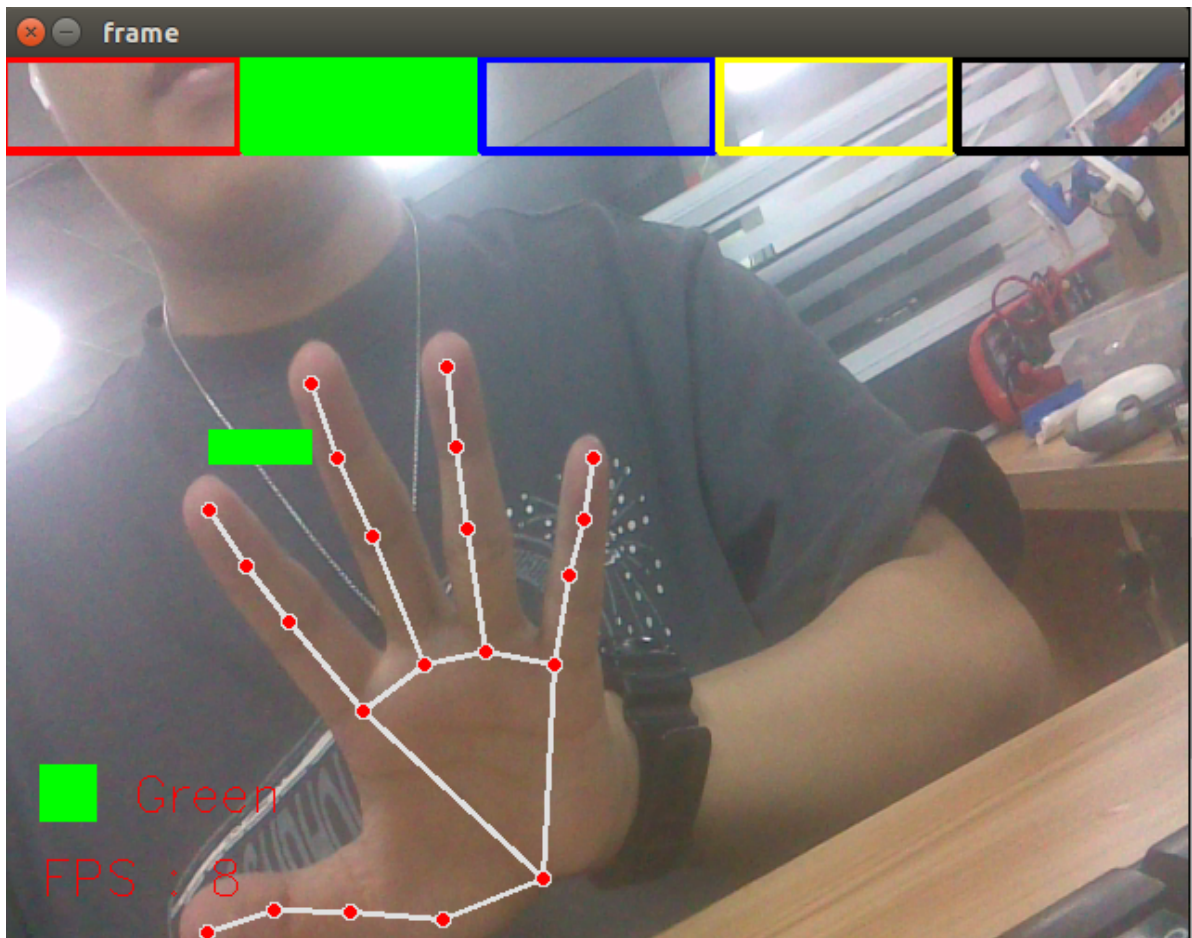
design sketch:

```
python3 08_Objectron.py
```
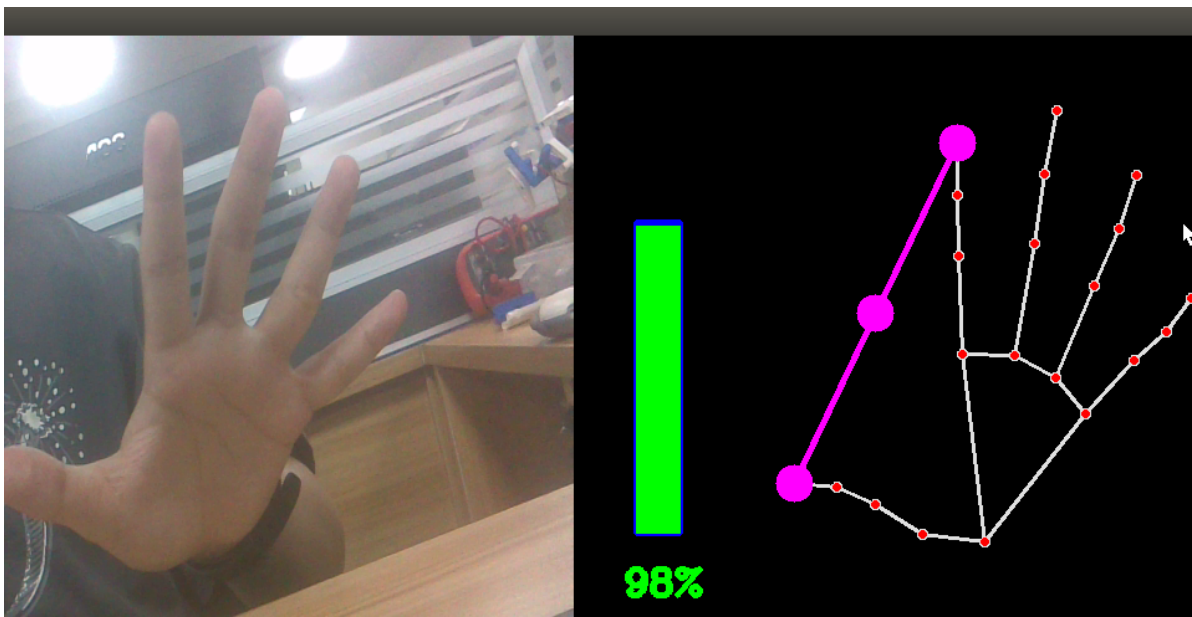
design sketch:

```
python3 09_VirtualPaint.py
```
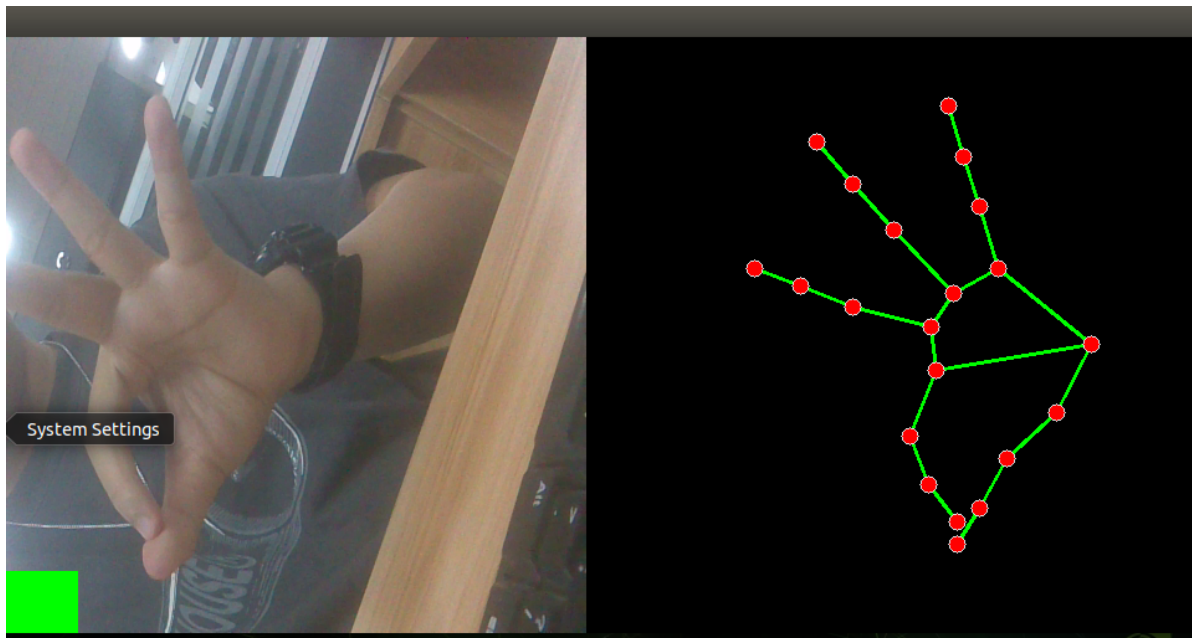
design sketch:


```
python3 10_HandCtrl.py
```

design sketch:


```
python3 11_GestureRecognition.py
```

design sketch:

Note: These cases are all used on the IMx219 onboard camera. If you want to use a USB camera, you can modify the capture in the program to look like the following.

```python
if __name__ == '__main__':
    capture = cv.VideoCapture(0)
    capture.set(6, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    print("capture get FPS : ", capture.get(cv.CAP_PROP_FPS))
    hand_detector = handDetector(detectorCon=0.85)
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv.flip(frame, 1)
        h, w, c = frame.shape
        frame,lmList = hand_detector.findHands(frame, draw=False)
        if len(lmList) != 0:
            # print(lmList)
            # tip of index and middle fingers
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]
            fingers = hand_detector.fingersUp()
            if fingers[1] and fingers[2]:
                # print("Seclection mode")
                if y1 < top_height:
                    if 0 < x1 < int(w / 5) - 1:
                        boxx = 0
                        Color = "Red"
                    if int(w / 5) < x1 < int(w * 2 / 5) - 1:
                        boxx = int(w / 5)
```

3.MediaPipe Hands

MediaPipe Hands is a high fidelity hand and finger tracking solution. It uses machine learning (ML) to infer the 3D coordinates of 21 hands from a single frame.

After palm detection of the entire image, precise key point localization of 21 3D hand joint coordinates within the detected hand area is performed through regression based on the hand marking model, which is known as direct coordinate prediction. This model learns consistent internal hand pose representations and is robust even to partially visible hands and self occlusion.

In order to obtain real ground data, approximately 30K real-world images were manually annotated using 21 3D coordinates, as shown below (obtaining Z values from the image depth map, if each corresponding coordinate has a Z value). In order to better cover possible hand postures and provide additional supervision on the properties of hand geometry, high-quality

synthetic hand models were drawn in various backgrounds and mapped to corresponding 3D coordinates.

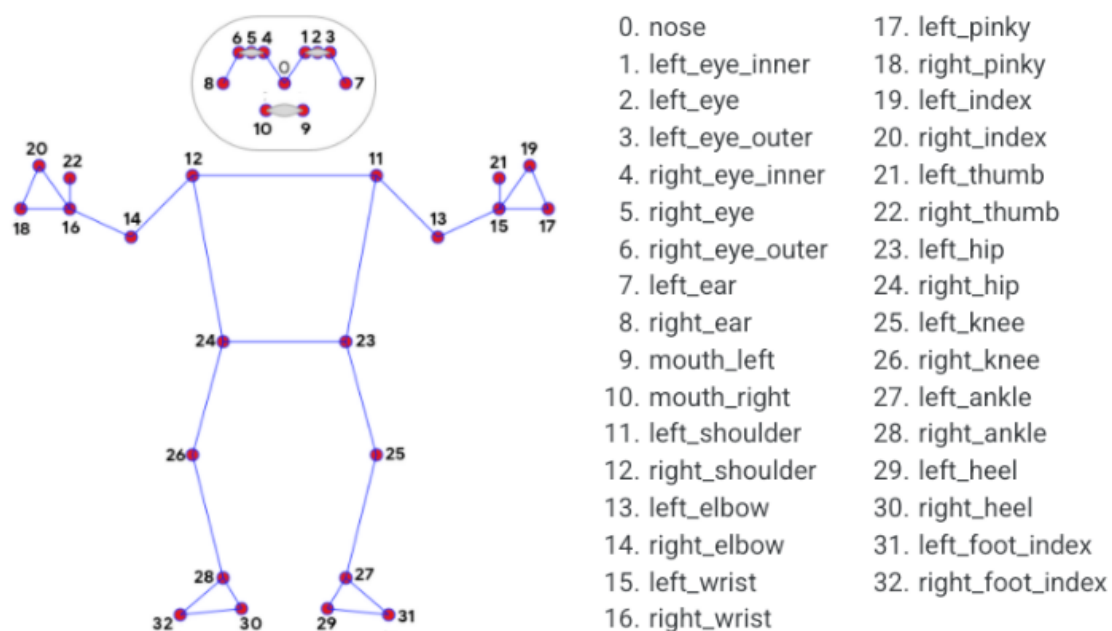|	
0. WRIST	11. MIDDLE_FINGER_DIP
1. THUMB_CMC	12. MIDDLE_FINGER_TIP
2. THUMB_MCP	13. RING_FINGER_MCP
3. THUMB_IP	14. RING_FINGER_PIP
4. THUMB_TIP	15. RING_FINGER_DIP
5. INDEX_FINGER_MCP	16. RING_FINGER_TIP
6. INDEX_FINGER_PIP	17. PINKY_MCP
7. INDEX_FINGER_DIP	18. PINKY_PIP
8. INDEX_FINGER_TIP	19. PINKY_DIP
9. MIDDLE_FINGER_MCP	20. PINKY_TIP
10. MIDDLE_FINGER_PIP	
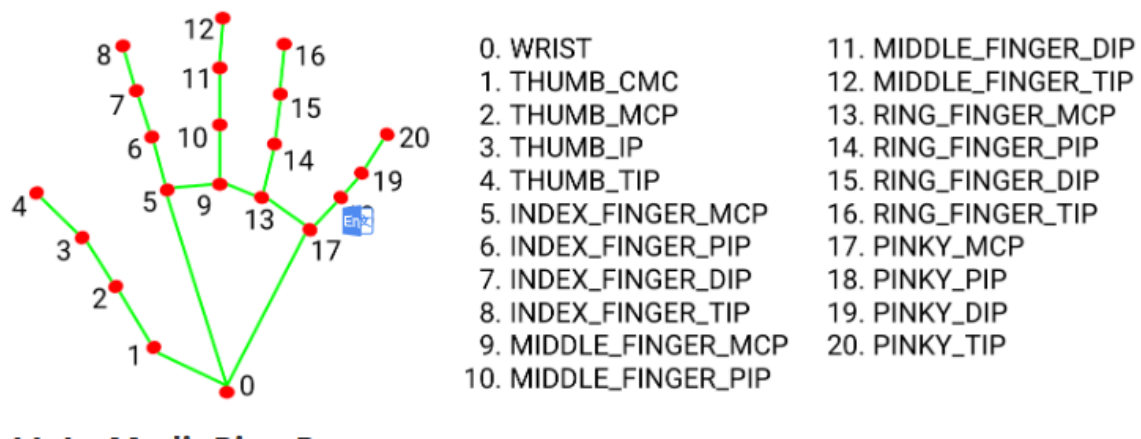
4、 MediaPipe Pose

MediaPipe Pose is an ML solution for high-fidelity body pose tracking. Using BlazePose research, 33 3D coordinates and full body background segmentation masks were inferred from RGB video frames, which also provided impetus for the ML Kit pose detection API.The landmark model in the MediaPipe pose predicted the positions of 33 pose coordinates (see figure below).

0. nose	17. left_pinky
1. left_eye_inner	18. right_pinky
2. left_eye	19. left_index
3. left_eye_outer	20. right_index
4. right_eye_inner	21. left_thumb
5. right_eye	22. right_thumb
6. right_eye_outer	23. left_hip
7. left_ear	24. right_hip
8. right_ear	25. left_knee
9. mouth_left	26. right_knee
10. mouth_right	27. left_ankle
11. left_shoulder	28. right_ankle
12. right_shoulder	29. left_heel
13. left_elbow	30. right_heel
14. right_elbow	31. left_foot_index
15. left_wrist	32. right_foot_index
16. right_wrist	

5.dlib

The corresponding case is facial effects.DLIB is a modern C++toolkit that includes machine learning algorithms and tools for creating complex software in C++to solve real-world problems. It is widely used in industries and academia in fields such as robotics, embedded devices, mobile phones, and large-scale high-performance computing environments.The dlib library uses 68 points to mark important parts of the face, such as the right eyebrow at 18-22 points and the mouth at 51-68 points. Get using the dlib library_ frontal_ face_ The detector module detects faces and uses shape_ predictor_ 68_ face_ Landmarks. dat feature data for predicting facial feature values