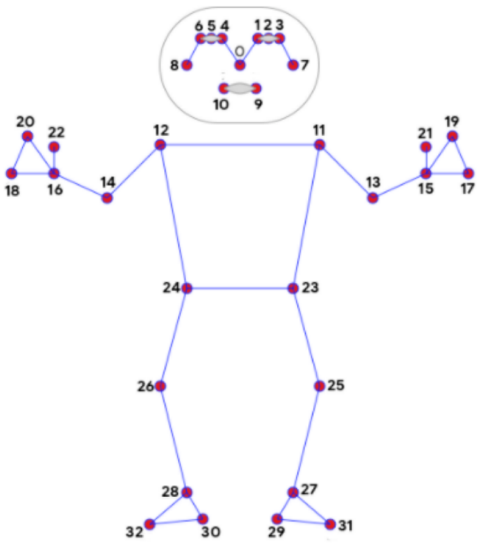
2.Posture detection

1.MediaPipe Pose

MediaPipe Pose is an ML solution for body pose tracking with high fidelity. Through BlazePose research, 33 3D coordinates and full background segmentation masks are inferred from RGB video frames. This study also provides dynamic support for the ML Kit pose detection API.

The landmark model in the MediaPipe pose predicted the positions of 33 pose coordinates (see figure below).



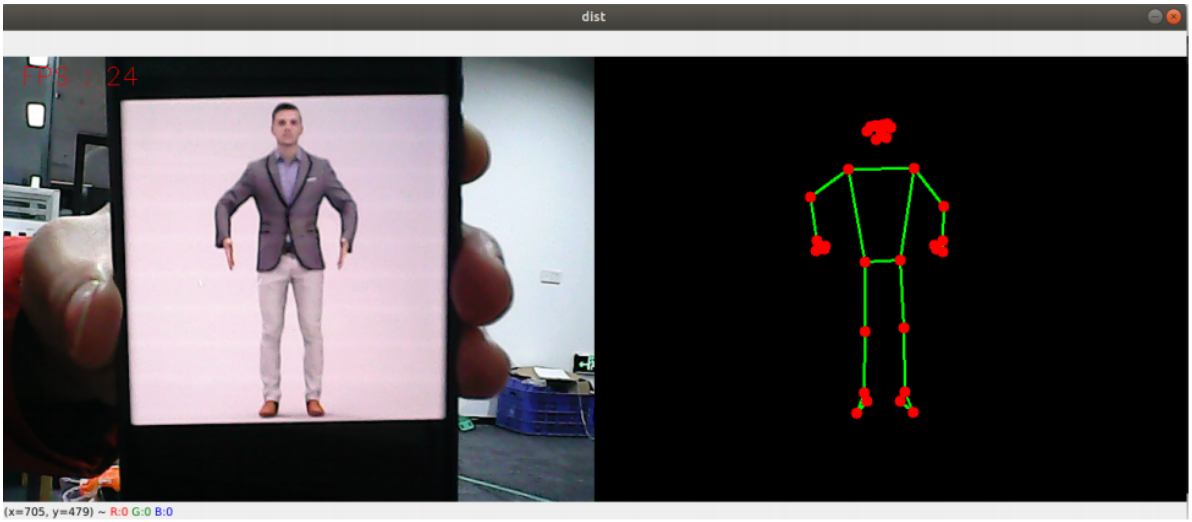
0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index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

2. Posture detection

1) Start up

Input following command:

```
ros2 run yahboomcar_mediapipe 02_PoseDetector
```



2) Code

Code path:
~/orbbec_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/02_PoseDetector.py

```
#!/usr/bin/env python3
# encoding: utf-8

#import ros lib
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Point
import mediapipe as mp
#import define msg
from yahboomcar_msgs.msg import PointArray
#import commom lib
import cv2 as cv
import numpy as np
import time
print("import done")

class PoseDetector(Node):
    def __init__(self, name,mode=False, smooth=True, detectionCon=0.5,
trackCon=0.5):
        super().__init__(name)
        self.mpPose = mp.solutions.pose
        self.mpDraw = mp.solutions.drawing_utils
        self.pose = self.mpPose.Pose(
            static_image_mode=mode,
            smooth_landmarks=smooth,
            min_detection_confidence=detectionCon,
            min_tracking_confidence=trackCon )
        self.pub_point =
self.create_publisher(PointArray,'/mediapipe/points',1000)
        self.lmDrawSpec = mp.solutions.drawing_utils.DrawingSpec(color=(0, 0,
255), thickness=-1, circle_radius=6)
        self.drawSpec = mp.solutions.drawing_utils.DrawingSpec(color=(0, 255,
0), thickness=2, circle_radius=2)

    def pubPosePoint(self, frame, draw=True):
        pointArray = PointArray()
        img = np.zeros(frame.shape, np.uint8)
        img_RGB = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
        self.results = self.pose.process(img_RGB)
        if self.results.pose_landmarks:
            if draw: self.mpDraw.draw_landmarks(frame,
self.results.pose_landmarks, self.mpPose.POSE_CONNECTIONS, self.lmDrawSpec,
self.drawSpec)
            self.mpDraw.draw_landmarks(img, self.results.pose_landmarks,
self.mpPose.POSE_CONNECTIONS, self.lmDrawSpec, self.drawSpec)
            for id, lm in enumerate(self.results.pose_landmarks.landmark):
                point = Point()
                point.x, point.y, point.z = lm.x, lm.y, lm.z
                pointArray.points.append(point)
        self.pub_point.publish(pointArray)
```

```
        return frame, img

    def frame_combine(slef,frame, src):
        if len(frame.shape) == 3:
            frameH, frameW = frame.shape[:2]
            srcH, srcW = src.shape[:2]
            dst = np.zeros((max(frameH, srcH), frameW + srcW, 3), np.uint8)
            dst[:, :frameW] = frame[:, :]
            dst[:, frameW:] = src[:, :]
        else:
            src = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
            frameH, frameW = frame.shape[:2]
            imgH, imgW = src.shape[:2]
            dst = np.zeros((frameH, frameW + imgW), np.uint8)
            dst[:, :frameW] = frame[:, :]
            dst[:, frameW:] = src[:, :]
        return dst

def main():
    print("start it")
    rclpy.init()
    pose_detector = PoseDetector('pose_detector')
    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))
    pTime = cTime = 0
    index = 3
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv.flip(frame, 1)
        frame, img = pose_detector.pubPosePoint(frame,draw=False)
        if cv.waitKey(1) & 0xFF == ord('q'): break
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        text = "FPS : " + str(int(fps))
        cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0,
255), 1)
        dist = pose_detector.frame_combine(frame, img)
        cv.imshow('dist', dist)
        # cv.imshow('frame', frame)
        # cv.imshow('img', img)
    capture.release()
    cv.destroyAllWindows()
```