

3. Overall detection

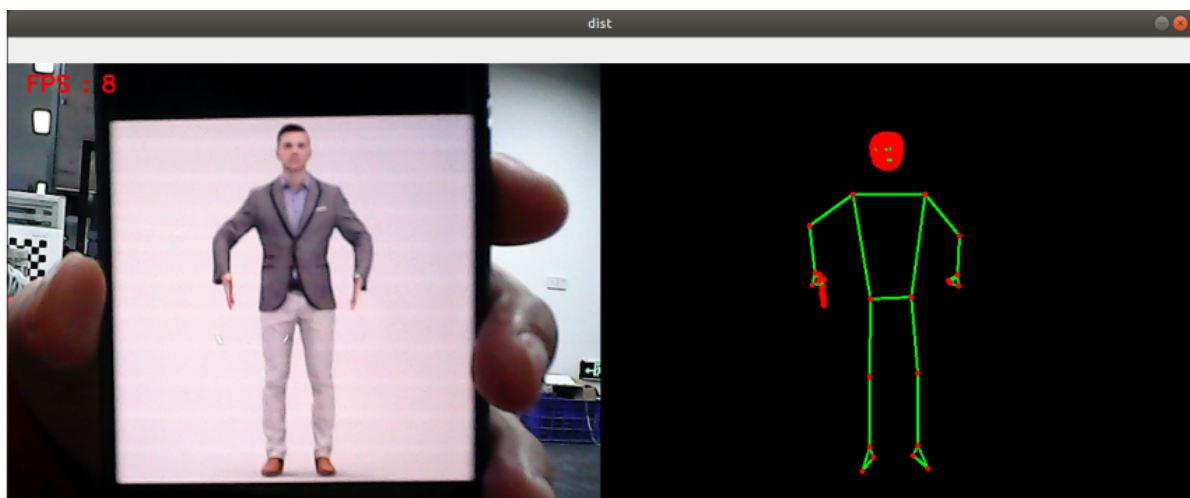
3.1 Overall detection

Combining the content of the first two sections, 1. Hand detection/2. Post detection, this routine implementation can detect both palm and human functions.

1. Start

Input following command:

```
ros2 run yahboomcar_mediapipe 03_Holistic
```



2. Code

Code path: ~/orbbec_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/03_Holistic.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 Holistic(Node):
    def __init__(self, name, staticMode=False, landmarks=True, detectionCon=0.5,
trackingCon=0.5):
        super().__init__(name)
```

```

self.mpHolistic = mp.solutions.holistic
self.mpFaceMesh = mp.solutions.face_mesh
self.mpHands = mp.solutions.hands
self.mpPose = mp.solutions.pose
self.mpDraw = mp.solutions.drawing_utils
self.mpholistic = self.mpHolistic.Holistic(
    static_image_mode=staticMode,
    smooth_landmarks=landmarks,
    min_detection_confidence=detectionCon,
    min_tracking_confidence=trackingCon)
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=3)
self.drawSpec = mp.solutions.drawing_utils.DrawingSpec(color=(0, 255,
0), thickness=2, circle_radius=2)

def findHolistic(self, frame, draw=True):
    pointArray = PointArray()
    img = np.zeros(frame.shape, np.uint8)
    img_RGB = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    self.results = self.mpholistic.process(img_RGB)
    if self.results.face_landmarks:
        if draw: self.mpDraw.draw_landmarks(frame,
self.results.face_landmarks, self.mpFaceMesh.FACEMESH_CONTOURS, self.lmDrawSpec,
self.drawSpec)
        self.mpDraw.draw_landmarks(img, self.results.face_landmarks,
self.mpFaceMesh.FACEMESH_CONTOURS, self.lmDrawSpec, self.drawSpec)
        for id, lm in enumerate(self.results.face_landmarks.landmark):
            point = Point()
            point.x, point.y, point.z = lm.x, lm.y, lm.z
            pointArray.points.append(point)
    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)
    if self.results.left_hand_landmarks:
        if draw: self.mpDraw.draw_landmarks(frame,
self.results.left_hand_landmarks, self.mpHands.HAND_CONNECTIONS,
self.lmDrawSpec, self.drawSpec)
        self.mpDraw.draw_landmarks(img, self.results.left_hand_landmarks,
self.mpHands.HAND_CONNECTIONS, self.lmDrawSpec, self.drawSpec)
        for id, lm in enumerate(self.results.left_hand_landmarks.landmark):
            point = Point()
            point.x, point.y, point.z = lm.x, lm.y, lm.z
            pointArray.points.append(point)
    if self.results.right_hand_landmarks:

```

```

        if draw: self.mpDraw.draw_landmarks(frame,
self.results.right_hand_landmarks, self.mpHands.HAND_CONNECTIONS,
self.lmDrawSpec, self.drawSpec)
        self.mpDraw.draw_landmarks(img, self.results.right_hand_landmarks,
self.mpHands.HAND_CONNECTIONS, self.lmDrawSpec, self.drawSpec)
        for id, lm in enumerate(self.results.right_hand_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(self, 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()
    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
    holistic = Holistic('holistic')
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv.flip(frame, 1)
        frame, img = holistic.findHolistic(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, 0.8, (0, 0,
255), 2)
        dist = holistic.frame_combine(frame, img)
        cv.imshow('dist', dist)
        # cv.imshow('frame', frame)
        # cv.imshow('img', img)
    capture.release()

```

```
cv.destroyAllWindows()
```