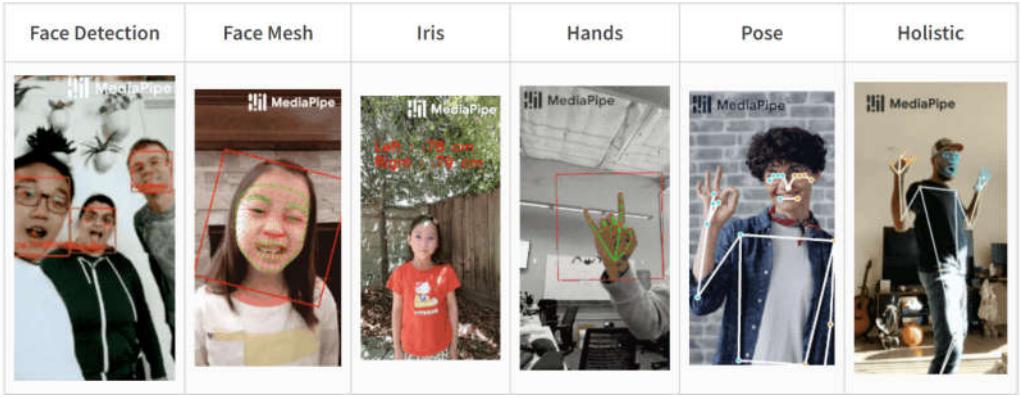
智能系统与控制

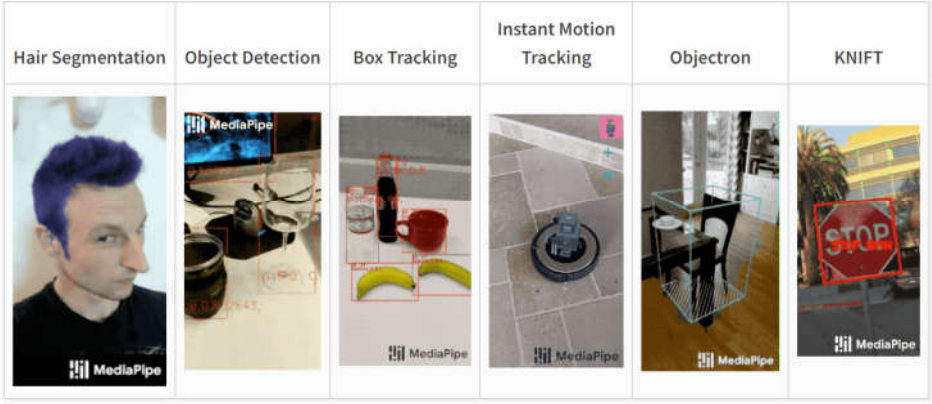
树莓派
Mediapipe-手势识别+虚拟按键

于泓
鲁东大学
信息与电气工程学院
2022.3.12

MediaPipe：Google Research 开源的跨平台多媒体机器学习模型应用框架

作为一款跨平台框架，MediaPipe 不仅可以被部署在服务器端，更可以在多个移动端（安卓和苹果 iOS）和嵌入式平台（Google Coral 和树莓派）中作为设备端机器学习推理（On-device Machine Learning Inference）框架。





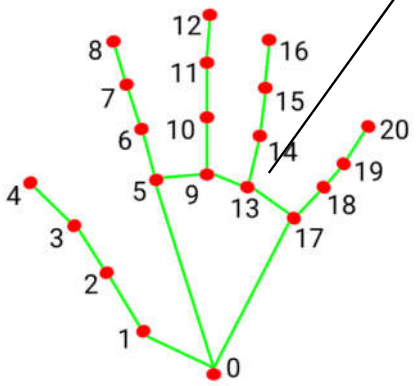
任务

- 1 利用mediapipe实现手部关键点检测
- 2 利用检测到的关键点，构造虚拟键盘，控制LED灯
- 显示不同的颜色。

(1) mediapipe手部关键点检测

```python
import mediapipe as mp
import cv2
import numpy as np


if __name__ == "__main__":


    # 打开摄像头

    cap = cv2.VideoCapture(0)
    # 定义手 检测对象
    mpHands = mp.solutions.hands
    hands = mpHands.Hands()
    mpDraw = mp.solutions.drawing_utils

    while True:

        # 读取一帧图像
        success, img = cap.read()
        if not success:
            continue

        img=cv2.flip(img, 1)
        image_height, image_width, _ = np.shape(img)
```

检测21个关键点



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

Fig 2. 21 hand landmarks.

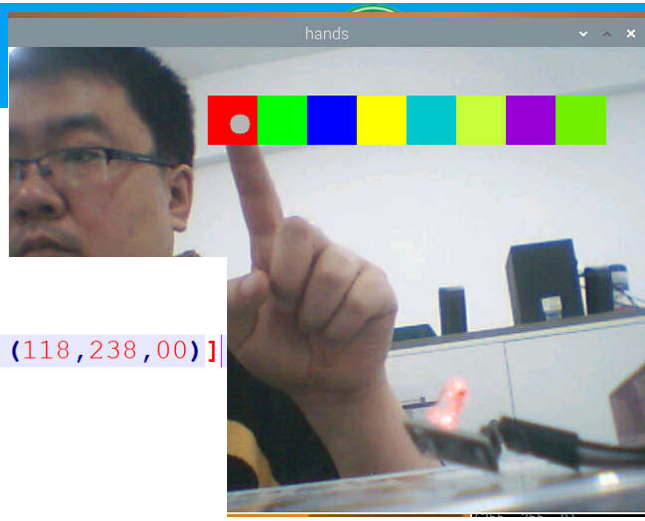
```
    # 转换为RGB
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 得到检测结果
    results = hands.process(imgRGB)

    if results.multi_hand_landmarks:
        for hand in results.multi_hand_landmarks:

            print("\r%.2f %.2f %.2f %.2f %.2f %.2f "%(hand.landmark[0].z,hand.landmark[4].z,hand.landmark[8].z,hand.landmark[12

            mpDraw.draw_landmarks(img,hand,mpHands.HAND_CONNECTIONS)

            # for i in range(21):
                # pos_x = hand.landmark[i].x*image_width
                # pos_y = hand.landmark[i].y*image_height
                # # 画点
                # cv2.circle(img, (int(pos_x),int(pos_y)), 3, (0,255,255),-1)


    cv2.imshow("hands",img)

    key =  cv2.waitKey(1) & 0xFF

    # 按键 "q" 退出
    if key ==  ord('q'):
        break
cap.release()
```

遍历所有找到的手

利用内置的绘图函数进行手部关键点的绘制

自己进行关键点的绘制

构造一排彩色的虚拟按键




```
def draw_button(img):

    colors = [(255,0,0),(0,255,0),(0,0,255),(255,255,0),(0,197,204),(192,255,62),(148,0,211),(118,238,00)]
    h = 50
    w = 200
    for i,color in enumerate(colors):

        pos1 = (w+i*50,h)
        pos2 = (w+(i+1)*50,h+50)


        cv2.rectangle(img,pos1,pos2,(color[2],color[1],color[0]),cv2.FILLED)


    return((w,h),pos2)
```

按键区域的左上角

每个按键 50×50

颜色 RGB转BGR

根据指尖的位置进行颜色拾取

```
def color_pick_up(img,pos_f,color_area):

    pos1 = color_area[0]
    pos2 = color_area[1]

    if pos_f[0]>pos1[0] and pos_f[0]<pos2[0] and pos_f[1] > pos1[1] and pos_f[1] < pos2[1]:

        color = (img[pos_f[1],pos_f[0],2],img[pos_f[1],pos_f[0],1],img[pos_f[1],pos_f[0],0])
    else:
        color = None

    return color
```

指尖不在按键区域返回None

小灯颜色控制部分，沿用PWM-RGB小灯控制程序

```
class RGB_LED(object):
    def __init__(self,pin_R,pin_G,pin_B):
        self.pins = [pin_R,pin_G,pin_B]

        # 设置为输出引脚，初始化第电平，灯灭
        for pin in self.pins:
            GPIO.setup(pin, GPIO.OUT)
            GPIO.output(pin, GPIO.LOW)

        # 设置三个引脚为pwm对象，频率2000
        self.pwm_R = GPIO.PWM(pin_R, 2000)
        self.pwm_G = GPIO.PWM(pin_G, 2000)
        self.pwm_B = GPIO.PWM(pin_B, 2000)

        # 初始占空比为0
        self.pwm_R.start(0)
        self.pwm_G.start(0)
        self.pwm_B.start(0)
```

```
def setColor(self,col):
    R_val,G_val,B_val = col

    R =self.color2ratio(R_val, 0, 255, 0, 100)
    G =self.color2ratio(G_val, 0, 255, 0, 100)
    B =self.color2ratio(B_val, 0, 255, 0, 100)
    # 改变占空比
    self.pwm_R.ChangeDutyCycle(R)
    self.pwm_G.ChangeDutyCycle(G)
    self.pwm_B.ChangeDutyCycle(B)
```

设置小灯颜色

```
def color2ratio(self,x,min_color,max_color,min_ratio,max_ratio):
    return (x - min_color) * (max_ratio - min_ratio) / (max_color - min_color) + min_ratio
```

颜色转占空比

```
def destroy(self):
    self.pwm_R.stop()
    self.pwm_G.stop()
    self.pwm_B.stop()
    for pin in self.pins:
        GPIO.output(pin, GPIO.HIGH)
    GPIO.cleanup()
```

退出

```python
import mediapipe as mp
import cv2
import numpy as np
import RPi.GPIO as GPIO
from pin_dic import pin_dic
```

```python
if __name__ == "__main__":

    # 设置引脚编号模式
    GPIO.setmode(GPIO.BOARD)

    # 定义三个引脚
    pin_R = pin_dic['G17']
    pin_G = pin_dic['G16']
    pin_B = pin_dic['G13']

    # 定义 RGB_LED 对象
    m_RGB_LED = RGB_LED(pin_R,pin_G,pin_B)

    # 打开摄像头
    cap = cv2.VideoCapture(0)

    # 定义手 检测对象
    mpHands = mp.solutions.hands
    hands = mpHands.Hands()

    color_p = (0,0,0)
```

```python
while True:

    # 读取一帧图像
    success, img = cap.read()
    if not success:
        continue

    img=cv2.flip(img, 1)
    image_height, image_width, _ = np.shape(img)

    # 转换为RGB
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 画按键
    color_area = draw_button(img)

    # 得到检测结果
    results = hands.process(imgRGB)

    if results.multi_hand_landmarks:

        hand = results.multi_hand_landmarks[0]
        pos_x = hand.landmark[8].x*image_width
        pos_y = hand.landmark[8].y*image_height
        pos_f = (int(pos_x),int(pos_y))

        color_c = color_pick_up(img,pos_f,color_area)
```

只取一只手

计算食指指尖

```
    if not color_c is None:

        if color_c != color_p:

            print(color_c)
            m_RGB_LED.setColor(color_c)
            color_p = color_c


    cv2.circle(img, pos_f, 10, (180,180,180),-1)


    cv2.imshow("hands",img)

    key =  cv2.waitKey(1) & 0xFF

    # 按键 "q" 退出
    if key ==  ord('q'):
        break
cap.release()
m_RGB_LED.destroy()
```

有颜色变化
驱动小灯

画出食指的位置

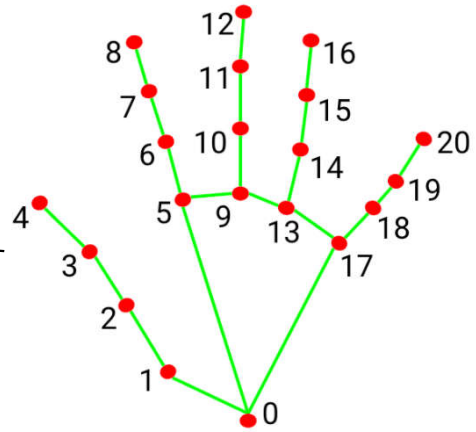
按q退出

```
import mediapipe as mp
import cv2
import numpy as np


if __name__ == "__main__":

    # 打开摄像头
    cap = cv2.VideoCapture(0)

    # 定义手 检测对象
    mpHands = mp.solutions.hands
    hands = mpHands.Hands()
    mpDraw = mp.solutions.drawing_utils
```

```
mp_hands.Hands(
  model_complexity=0,
  min_detection_confidence=0.5,
  min_tracking_confidence=0.5)
```

引入跟踪
增加速度

检测21个关键点



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

Fig 2. 21 hand landmarks.

```
while True:

    # 读取一帧图像
    success, img = cap.read()
    image_height, image_width, _ = img.shape

    # 转换为RGB
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 得到检测结果
    results = hands.process(imgRGB)

    if results.multi_hand_landmarks:
        for hand in results.multi_hand_landmarks:

            print("\r%.2f %.2f %.2f %.2f %.2f %.2f "%(hand.landmark[0].z,hand.landmark[4].z,hand.landmark[8].z,hand.landmark[12].z,hand.lar

            mpDraw.draw_landmarks(img,hand,mpHands.HAND_CONNECTIONS)

            # for i in range(21):
                # pos_x = hand.landmark[i].x*image_width
                # pos_y = hand.landmark[i].y*image_height
                # # 画点
                # cv2.circle(img, (int(pos_x),int(pos_y)), 2, (0,0,255),-1)

    cv2.imshow("hands",img)

    key =  cv2.waitKey(1) & 0xFF

    # 按键 "q" 退出
    if key ==  ord('q'):
        break
cap.release()
```

hand.landmark[i].x → 归一化
hand.landmark[i].y → 横纵坐标
hand.landmark[i].z → 对手腕的相对位置

绘图

x,y 实际坐标的计算