基于MediaPipe手势控制计算机音量设计

张　晶　杨碎明　孙攀攀
西安交通工程学院　陕西　西安　721000

摘　要：随着计算机技术的高速发展，人们的日常生活逐渐趋近于科技化，在日常工作、学习、生活中也越来越不能离开人工智能，因此计算机与人类的结合度也会越来越高，人机交互程度也会越来越深入，其中手势识别是最为突出并且很重要的一项。本设计的内容是通过手势对计算机音量的大小进行控制，用MediaPipe提取手部骨架的21个关键点，并检测得出大拇指与食指指尖的坐标，通过坐标计算出两者之间的距离，再用线性插值算法，将距离处理为参数调用PyCaw中相关的音量控制函数实现手势对音量的实时控制，利用OpenCV画出柱状图像以显示音量的实时大小，从而实现手势调节音量大小。

关键词：MediaPipe；OpenCV；手势控制

引言

随着科技的进步，利用机器设备直接检测已经不能满足于现状。早期的手势识别仅仅建立在二维层面上的，通过相关特定的设备，捕捉人体具有关键特征的关节部位之间形成的空间结构与角度关系。基于手部动作的多样性，将常见的手势通过数据手套进行数据传输，依靠有限技术，把数据手套上接收到的数据信息通过有线连接传输到电脑上，利用电脑自动的生成一个关于该用户姿势的具体的数据库，便于具体实践的时候进行比对。早期的数据手套是由多个传感器元件组成的，基于数据手套分布密集的传感器带来的优势，能够更加精准的捕捉到手部的细微动作，从而达到与参考组更加明显的对比，但其使用过程繁琐操作难度较大，且其价格昂贵，使得该数据手套无法得到广泛的应用；后期的光学标记方法取代了数据手套，它通过红外线将人手位置和手指的变化传到系统屏幕上，虽然取得了良好的效果，但自然表达却被忽略；直到现在我们根据视觉来进行手势识别，通过图像处理，将视频拍摄到的手势进行序列排序，再用图像计算机视觉技术进行处理识别。我国各大高校的相关专业的学者和研究者对手势识别已经研究了多种方法，同时他们对这些方法进行了大量的创新，并投入到了具体的应用中。例如清华大学计算机科学的一些学者就创新了手势识别的方法，提出了一种基于运动分割的帧间图像运动估计方法，指出可以通过运动、形状、颜色和纹理等特性，对手势进行精确的检测。

1 设计思路

用MediaPipe提取手部骨架的21个关键点，并检测得出大拇指与食指指尖的坐标，通过坐标计算出两者之间的距离，再用线性插值算法,将距离处理为参数调用PyCaw中相关的音量控制函数实现手势对音量的实时控制，利用OpenCV画出柱状图像以显示音量的实时大小，从而实现手势调节音量大小[1]。

2 设计与实现

2.1 手势识别

2.1.1 手势识别的具体实现步骤

本文主要是利用MediaPipe中的手部关键点检测技术来获取手部关键点信息，其具体步骤如下：

第一步，安装MediaPipe依赖库，并下载MediaPipe的预训练模型。

第二步，加载预训练模型，并初始化MediaPipe Hand对象。

```
import cv2
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
# 加载预训练模型
hands=mp_hands.Hands(static_image_mode=False, max_num_hands=2,min_detection_confidence=0.5, min_tracking_confidence=0.5)
# 初始化MediaPipe Hand对象
cap = cv2.VideoCapture(0)
while cap.isOpened():
success, image = cap.read()
if not success:
break
# 将RGB图像格式转化为MediaPipe支持的BGR图像格式
image = cv2.cvtColor(image, cv2.COLOR_
```

```
RGB2BGR)
        results = hands.process(image)
        # 对每个检测到的手，提取手部关键点
        if results.multi_hand_landmarks:
        For hand_landmarks in results.multi_hand_
landmarks:
        mp_drawing.draw_landmarks(image,hand_
landmarks, mp_hands.HAND_CONNECTIONS)
        # 展示结果
        cv2.imshow('MediaPipe Hands', image)
        cv2.destroyAllWindows()
        cap.release()
```

综上所述，MediaPipe手部关键点检测技术是一种基于深度学习的目标检测技术，通过卷积神经网络构建了有效的图像特征表示，并利用检测器定位手的位置和提取手部关键点，手部的关键点如图1所示。

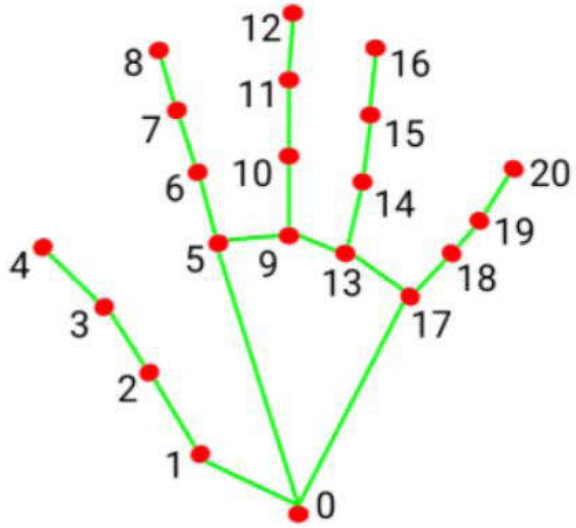

图1

第三步，对每个检测到的手，提取手部关键点。手部关键点共有21个，包括手腕、掌心、手指关节等，通过访问手部关键点的索引，可以获取相应位置的关键点坐标。

```
python
        # 对每个检测到的手，提取手部关键点
        if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_
landmarks:
        # 遍历21个手部关键点
        for idx, landmark in enumerate(hand_landmarks.
landmark):
        # 根据坐标和图像尺寸计算像素位置
        h, w, c = image.shape
        px, py = int(landmark.x * w), int(landmark.y * h)
        cv2.circle(image, (px, py), 5, (0, 255, 0), -1)
```

通过以上步骤，就可以利用MediaPipe中手部关键点检测技术，获取手部关键点信息。

2.2 利用OpenCV画音量柱状图像

利用OpenCV画音量柱状图像的原理是将声音数据转化为图像数据，并将其在画布上绘制成柱状图。具体步骤如下：

第一步，读取音频文件，将其转化为数字音频信号。

```
        import numpy as np
        import cv2
        import librosa
        # 读取音频文件
        y, sr = librosa.load('audio.wav', sr=None)
```

第二步，利用音频信号计算短时傅里叶变换（Short-time Fourier Transform, STFT），以获得音频信号每一帧的频率和强度信息。

```
        # 计算短时傅里叶变换
        stft = librosa.stft(y, n_fft=4096, hop_length=512)
        magnitude = np.abs(stft)
```

第三步，将得到的幅度谱图表示为图像，并将图像大小调整为随意的大小。

```
        # 将幅度谱图以0到255的范围映射到图像中
        img = cv2.normalize(np.log(magnitude), None, 0,
255, cv2.NORM_MINMAX, cv2.CV_8UC1)
        # 调整图像大小
        img = cv2.resize(img, (800, 600))
```

第四步，定义柱状图的参数和位置，并将其绘制在图像上。

```
        # 定义柱状图参数
        bar_width = img.shape[1] // magnitude.shape[1]
        bar_x = np.arange(0, img.shape[1], bar_width)
        bar_height = np.sum(img, axis=0) // bar_width
        bar_y = img.shape[0] - bar_height
        # 绘制柱状图
        for i in range(magnitude.shape[1]):
        cv2.rectangle(img, (bar_x[i], bar_y[i]), (bar_x[i] +
bar_width, img.shape[0]), (255, 255, 255), -1)
```

第五步，展示最终的音量柱状图像。

```
        cv2.imshow('Volume', img)
        cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

综上所述，利用OpenCV画音量柱状图像的原理是将声音信号数据在频域上的强度信息表示为图像，然后根据得到的图像信息，在画布上绘制柱状图，以呈现声音的音量变化情况。这种方式已广泛应用于音频可视化和音频视觉化领域。

2.3 pycaw与Window滑杆关联

关于滑杆控件与pycaw的关联，一般是通过将滑杆的value值与系统音量的值进行绑定，从而实现拖动滑杆调节系统音量的功能。可以使用pycaw库中提供的类来获取默认音频渲染设备的音量控制对象，并通过调整VolumeScalar值来调整系统音量。通常，该过程包括以下几个步骤：

1.导入pycaw库和其他必需的模块。

```
import pyautogui
import pycaw.pycaw as pycaw
```

2.获取默认音频渲染设备的音量控制对象。

```
volume_object = pycaw.AudioUtilities.GetSpeakers()
volume_control = volume_object.AudioVolume
```

3.根据滑杆的值调整音量控制对象的音量。

```
# 获取滑杆的值
value = slider.get()
# 计算音量百分比
volume_percentage = value / 100.0
# 设置音量
volume_control.SetMasterVolumeLevelScalar(volume_percentage, None)
```

通过将滑杆的value值与系统音量的值进行绑定，可以在拖动滑杆时实时改变系统音量，从而实现平滑且有效的音量控制[2]。

3.本文基于MediaPipe实现了手势控制音量的应用，并且对其进行了详细的设计和实现。但是，目前的应用仍存在一些不足之处，需要进一步的研究探索来进一步完善其功能。

一方面，在应用场景方面，本文只是将其应用在了音量调节上，而且仅仅是控制了音量的大小。在未来的研究中，本文可以研究更多的音频相关的功能实现，如播放、暂停、上一首、下一首等。此外，还可以将手势控制应用在其他领域，如游戏，医疗等领域进行进一步的研究[3]。

另一方面，在算法方面，本文实现了对MediaPipe模型的调用，并以此为基础实现了手势识别功能。但由于手势种类繁多，本文只实现了少量样本的手势识别，因此在实际应用中可能还存在一定的误差。因此，未来的研究可以进一步提高模型的准确率，并增加更多的手势样本，使得其更加稳定可靠。

结束语：综上所述、本项目基于MediaPipe手势控制计算机音量，通过摄像头获取手势，计算食指和大拇指之间的距离，从而控制音量的大小。随着智能化的发展，人机交互程度也会越来越深入，在人们吃东西的时候或者其他不方便用鼠标的时候，为了方便，对人机交互的要求更高。在不久的将来，手势控制音量的应用肯定会越来越广。

参考文献

[1]刘德发.基于MediaPipe的数字手势识别[J].电子制作,2022：9

[2]王如斌,窦全礼,张淇,等.基于MediaPipe的手势识别用于挖掘机遥操作控制[J].土木建筑工程信息技术,2021:8.

[3]许向前.Authorware中音量的控制[J].信息技术教育,2018:75-76.