2022 年英特尔杯大学生电子设计竞赛嵌入式系统专题邀请赛

2022 Intel Cup Undergraduate Electronic Design Contest

- Embedded System Design Invitational Contest

作品设计报告

Final Report

Intel Cup Embedded System Design Contest

报告题目：智能康复训练辅助系统

学生姓名：张耿明 张吉哲 刘嘉颖

指导教师：刘宁艳

参赛学校：西安交通大学

2022 年英特尔杯大学生电子设计竞赛嵌入式系统专题邀请赛

参赛作品原创性声明

本人郑重声明：所呈交的参赛作品报告，是本人和队友独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果，不侵犯任何第三方的知识产权或其他权利。本人完全意识到本声明的法律结果由本人承担。

参赛队员签名：

张耿明　张吉哲　刘嘉毅

日期：2022 年 7 月 26 日

智能康复训练辅助系统

摘要

为解决国内复健医疗资源短缺、疫情形势下患者难以与康复师面对面复健等问题，团队基于 GNS-V40 边缘计算主机开发了一款智能康复训练辅助系统。该系统充分利用人工智能、边缘计算等技术，实现了康复师查看患者数据、制定训练方案、录制训练动作，患者康复动作的评价与纠正、身体状态评测、异常状态监测等功能。经测试，系统对功能的实现完整、准确，充分发挥出了 GNS-V40 的优异性能，有较高的实用价值和广阔的前景。

关键词：计算机视觉，康复训练，姿态估计，人脸识别，语音交互

The Intelligent Rehabilitation Training Auxiliary System

ABSTRACT

In order to solve the problems such as the shortage of rehabilitation medical resources in the domestic and the difficulty of Face to face communication between patients and rehabilitation physicians during the epidemic, our team developed an intelligent rehabilitation training assistance system based on Edge computing host GNS-V40. The system makes full use of artificial intelligence, edge calculation and other technologies to realize the functions such as rehabilitation physicians viewing patient data, making training programs, recording training movements, evaluating and correcting patients' rehabilitation actions, evaluating and evaluating physical condition, monitoring abnormal conditions, and so on. After testing, the system realizes the function completely and accurately, gives full play to the excellent performance of GNS-V40, and has high practical value and broad prospect.

Key words: Computer vision, rehabilitation training, pose estimation, face recognition, voice interaction

目录

第一章 绪论

1.1 项目背景和意义

当前我国复健医疗资源紧俏而复健需求人群数量庞大，康复产品市场前景广阔。本作品的设计针对出院后需要在家中进行康复训练的患者，切实解决患者康复训练的需求，具有极高的现实意义。

1.1.1 项目背景

（1）复健医疗资源紧俏而复健需求人群数量庞大

我国运动康复产业具有起步晚，机构数量少的特点，它萌芽于 2008 年，起步于 2012 年。截止到 2020 年年底，从事运动康复服务的机构门店数量仅有 370 家[1]。

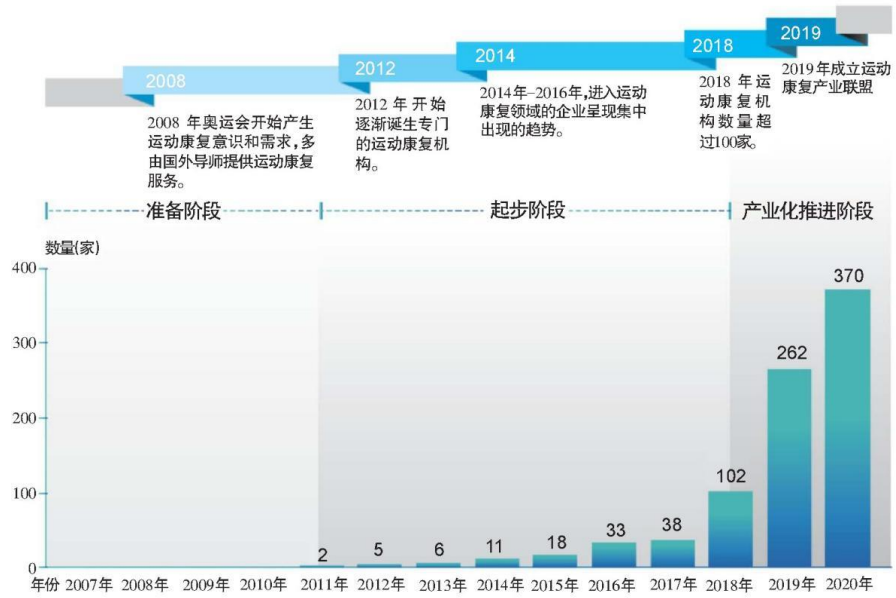


图 1-1 中国运动康复产业重要历程及门店数量发展

从国外的运动康复机构数量来看，欧美发达国家平均每 4000 人可拥有一家康复服务机构，而中国平均每 362 万人才拥有一家康复服务机构。如此紧俏的医疗资源使得运动康复的价格居高不下，一线城市普通复健运动一小时需要 800-1200 元，有些更高端更专业的复健运动甚至高达 5000 元/小时。

目前我国复健需求数量庞大，包括骨科术后康复、运动损伤康复、肌肉疼痛康复、姿势体态矫正、慢性病康复、产后康复等；复健人群构成复杂，如术后康复人群、生长发育时期的青少年、身体机能衰退的老年人、运动员等。复健医疗资源紧俏而复健需求人群数量庞大这一对矛盾亟待解决。

（2） 患者在家中复健时存在的问题

1）患者在出院后依存性下降与心理问题有待解决

以脑卒中患者为例，康复锻炼是恢复脑卒中患者日常生活能力的主要手段，也是脑卒中首选的康复疗法。国内临床发现，脑卒中患者在住院期间，依从性好的人数占 63%～82%，但出院后依从性好的只占 47.41%，严重影响脑卒中患者的康复[2] ，其依从性随时间的变换如图 1-2 所示。从图中可以看出患者初期依从性好，而中长期的依从性较差。国外也有类似

研究与发现[3]。

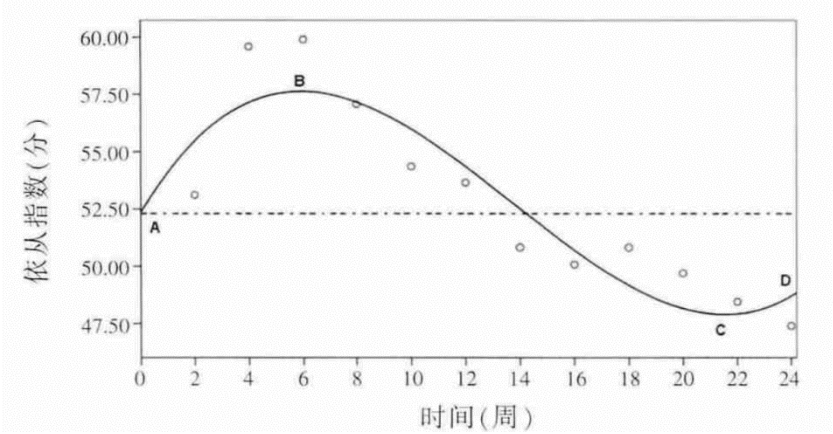


图 1-2 患者依从性随时间的变换

同时，当一个人患病时，心理会出现急躁、抑郁、恐惧、伤心、自卑、畏难退缩等不良情绪。而复健恢复是一个漫长的过程，在复健没有产生效果时，病人可能会自我否定、自卑。又或者当复健操作过于复杂时，病人可能会急躁着急。这样的消极情绪不仅会对病人本就虚弱的身体产生伤害，还可能会使病人半途而废，放弃复健。

基于患者出院后依存性变化规律，并考虑到患者的心理状况，在康复训练过程中，应当能够监督患者完成相应训练计划，在患者未达到标准时给与提醒和鼓励。

2）自行复健时难以把握运动标准和运动量

当下复健机构资源紧俏，价格昂贵，多数人难以获得复健机构资源。许多人仅在医嘱下，自行在家进行复健。而医生给的医嘱大多比较简单，不是一整套完整的康复体系。自行复健时，所做的动作比较单一枯燥，难以将肌肉与关节全部调动起来。而且这样无法确定自己做出的动作的是否标准，特别是由于需要锻炼的部位关节不灵活、肌肉无力，更容易造成动作的变形，如果不及时纠正，可能会对身体造成更大伤害。有些人也会在网络上找一些教程、视频进行学习。这些教程给出的指导的正确性难以辨别，且视频中的动作难以模仿准确。自行复健时，还很难把控运动量的大小。如果运动量过小，则起不到恢复效果，耽误了最佳恢复时机，运动量过大则有可能造成二次伤害。

3）新冠疫情下，康复师面对面指导患者变得困难

随着全球疫情的持续不断,康复师面对面指导患者进行康复训练变得困难。而且线上指导时康复师无法全面把控患者动作的细节，也难以在线上全面系统的指导多名患者。

1.1.2 项目意义

该项目旨在帮助已出院且有日常活动能力的患者能够独自在家中进行科学专业的康复训练，享受康复师的指导。实现康复师远程在线上为多位患者制定训练计划，获取多位患者训练评价结果和数据，着重关注训练状态不理想的患者并及时调整训练计划；实现系统智能监督、督促患者有效地完成康复训练计划，并及时给与患者动作纠正与正向反馈，评价患者训练情况的功能。该项目可以有效缓解复健医疗资源紧俏而复健需求人群数量庞大、患者出院后存在依存性下降、患者自身难以把握复健时的运动量、易出现心理问题、疫情形势下难以面对面辅导等情况。

1.2 国内外研究现状

随着人工智能、大数据、边缘计算等技术的发展，国内外康复训练系统研究取得许多新的突破，涌现出了一批新型康复系统，它们各有优缺点。了解国内外研究现状与现有的康复系统的特点可以使我们在设计复健系统时提供参考与帮助。

1.2.1 国外研究现状

目前国外已经使用的康复训练方式有很多种，例如将人体数据信息传感器置于患者的待测肢体上，就能获取患者的肢体信息数据[4][5]。而非专业人士难以独立正确佩戴这些设备，佩戴的方向或位置可能出现偏差，无法达到精细化评估的要求。

瓦拉多利德大学将十字轴结构运用于 E2 Rebot 机器人，并采用被动引导和主动辅助两种模式对患者进行康复训练[6]。但由于其机械结构限制，患者运动会受到一定程度上的约束，仅能实现小部分康复方案。同时，由机器人辅助康复训练的仪器较为昂贵，一般家庭难以负担。

美国加利福尼亚大学和芝加哥康复研究中心共同研究开发了一种面向康复初期、运动能力未完全恢复的患者的设备。它可以将患者的实际训练情况传输至医院，再由医师对患者做出指导[7]。这种设备无法对患者康复情况进行实时分析，但是这种非接触的康复训练模式给我们了一些启发。

随着计算机视觉和深度学习领域的飞速发展，有研究人员提出了一种基于计算机视觉的患者术后康复训练系统，如 Lin 等人开发了一个基于 Kinect 的康复系统，它利用“坐式太极”练习来帮助运动障碍患者[8]。与此同时，越来越多的开发者使用基于深度学习的方法来研究人体姿态，并实现了快速人体姿态的估计。

1.2.2 国内研究现状

国内对康复训练系统的研究起步较晚。之前一般使用中医疗法对患者进行术后康复治疗，包括针灸、推拿、按摩等[9]，随着国内对康复系统的研究越发重视，许多机构、企业和各大高校、实验室都加入了研发的行列。他们研究出的康复训练系统不仅可以有效辅助患者进行康复训练，还能节约人力物力等资源。

湖南理工大学的刘林等研究了一款针对轻度运动障碍人群的康复训练系统[10]。该系统采用了 5DT 数据手套作为采集数据信息的传感器，通过操控虚拟人物完成相应动作来进行训练；昆明理工大学研究开发了一套基于双目平行视觉技术的训练系统[11]。该系统将双目平行视觉技术和辅助康复设备进行了融合。它可以识别训练者身份，自动选择患者对应的康复训练方案。

沈天毓等人，用基于运动传感器单元的可穿戴设备，让帕金森患者进行指定动作训练，并根据患者运动过程中具体部位的加速度和角速度的详细信息为其运动功能评分[12]。

此外，有些系统利用虚拟现实游戏为患者提供舒适的康复训练虚拟环境[13][14]，通过摄像头检测康复人员的情绪,通过分析用户的不同情绪来动态调整康复训练的难度[15]。

随着人工智能等技术的发展，国内外都已经开发出一大批具有不同特色的提供康复训练运动的产品。但是目前现有产品对不同家居环境和用户个性化需求针对性不高，应用场景定义仍需明确，对于患者的心理需求关注度不够，与用户的期望存在一定差距。同时，这类智能康复训练产品未考虑患者与系统进行交互的困难，系统在设计过程中应该着重考虑病人的特点，从用户的角度出发，提供友好的交互方式。

1.3 技术路线

团队通过查阅文献和去康复院考察的方法进行市场调研，了解市场现有产品特点、当前用户的痛点，明确项目选题与目标人群。针对康复师和患者这两大类目标人群，团队确定了面向两大类用户的功能：康复师可以查看患者数据、制定训练方案、录制康复训练动作，患者可以使用该系统进行康复动作评价与纠正、身体状态评测、异常状态监测。基于功能定义，团队从算法、软件层、硬件层设计系统，完成系统的构建。通过多次系统评估与调试优化最终完成系统设计。项目技术路线如图 1-3 所示。

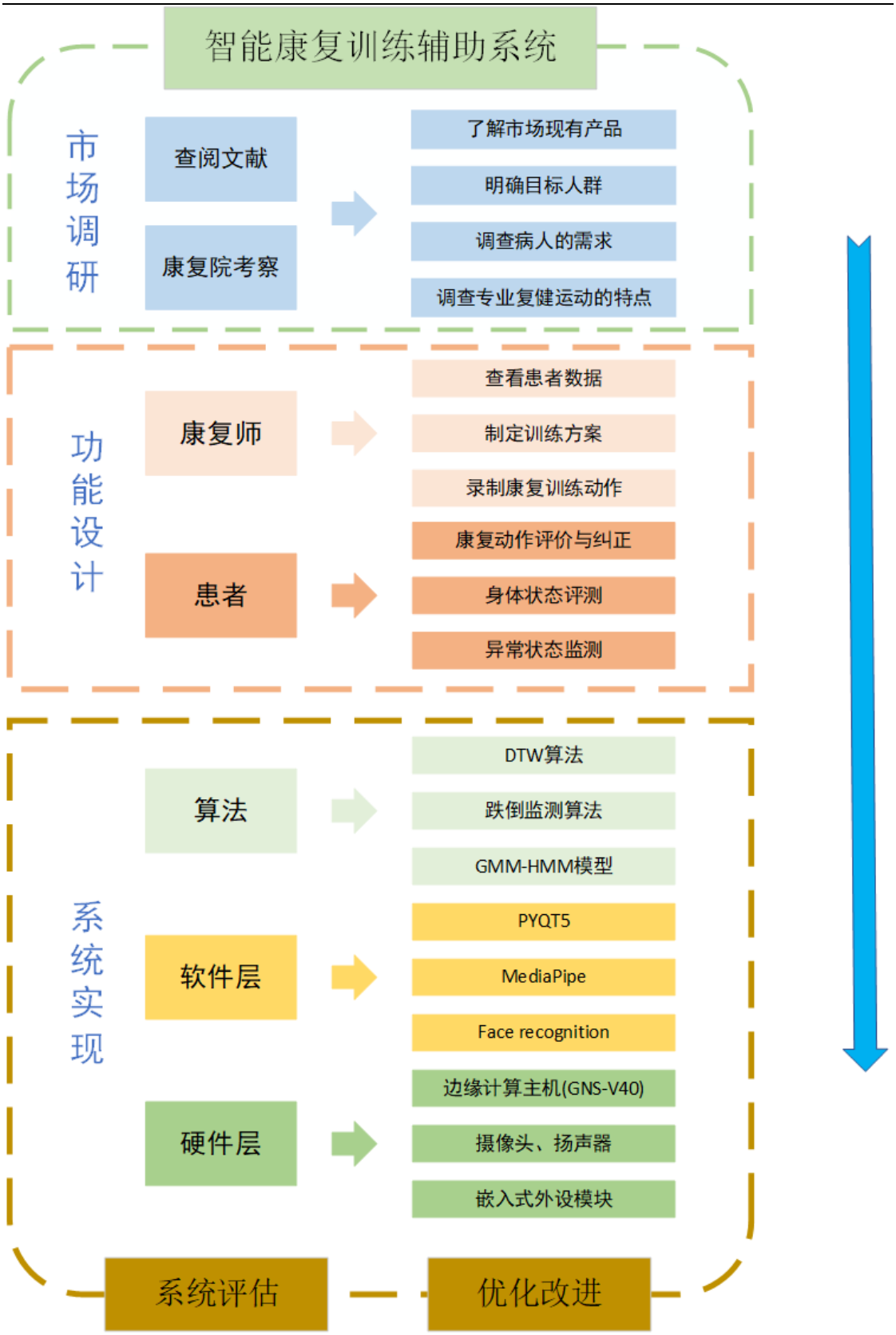

智能康复训练辅助系统
市场调研
查阅文献
康复院考察
了解市场现有产品
明确目标人群
调查病人的需求
调查专业复健运动的特点
功能设计
康复师
患者
查看患者数据
制定训练方案
录制康复训练动作
康复动作评价与纠正
身体状态评测
异常状态监测
系统实现
算法
软件层
硬件层
DTW算法
跌倒监测算法
GMM-HMM模型
PYQT5
MediaPipe
Face recognition
边缘计算主机(GNS-V40)
摄像头、扬声器
嵌入式外设模块
系统评估
优化改进

图1-3 技术路线

第二章 系统方案

针对复健人群在康复训练时的生理心理上的实际需求，本团队旨在设计一款能够从生理、心理多维帮助复健人群康复训练的智能化程度高，融合人脸识别、姿态识别、语音识别为一体的科学专业的智能康复训练辅助系统。

面向人群：该系统面向的用户师康复训练师和达到三级康复水平的患者。达到三级康复水平患者已经能够独立完成日常活动，适合返回社区、家中进行康复训练。但他们仍然需要进行社区康复医师在二级康复的基础上制定的康复计划。

使用场景：患者家中或医院。

系统评测：以达到三级康复水平的脑卒中患者为例，针对适合脑卒中患者的锻炼的动作进行评测。

2.1 系统功能

本系统从用户角度出发，基于康复师与病人的实际需求，对康复师提供了查看患者数据、患者康复动作与方案的制定的功能；对病人提供了身体状态评测、康复动作的评价与纠正、异常状态检测。另外根据不同场景、不同人员的需求提供了语音、触屏等多种交互方式，便于用户操控。其功能框图如图 2-1 所示。

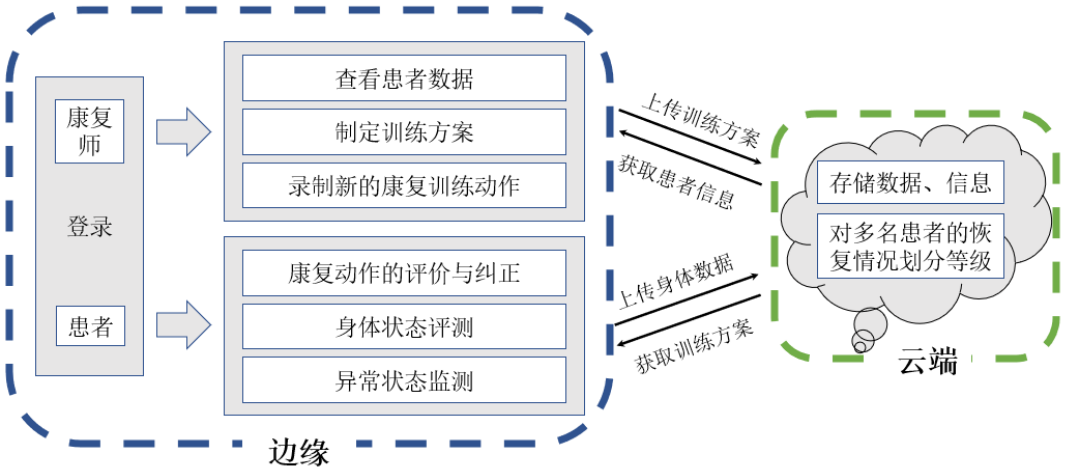


图2-1 功能框图

2.1.1 康复师：查看患者数据，制定训练方案

康复动作并不具有普适性，不同患者、同一患者的不同恢复时期都需要不同的康复运动方案。这对本系统康复动作的可扩展性提出了要求。

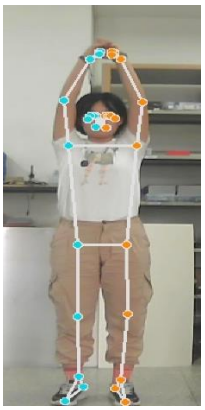
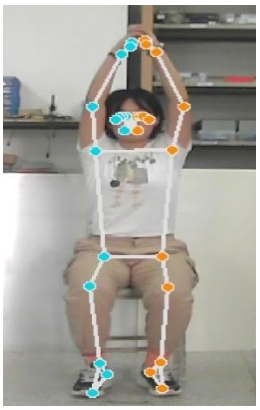
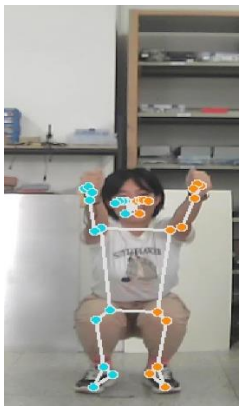
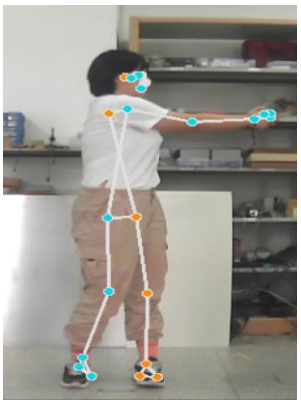
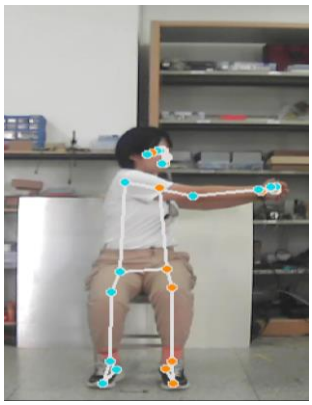
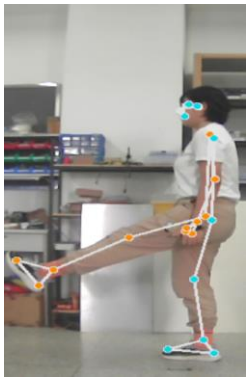
康复师可以自行录制新的康复动作并选择该动作需要着重关注的身体部位。系统将识别、分析该动作特征。康复师可以随时查看患者训练数据和身体机能情况，如患者四肢恢复程度、动作幅度、锻炼次数、3D 骨架图、心率、血压等数据以及系统对患者每个动作的评价，并可根据患者的数据动态调整康复训练方案，同时对患者进行留言鼓励。

2.1.2 患者：康复动作的评价与纠正

基于患者在家独立完成康复训练存在依存性下降，无人督促，无法及时得到动作反馈等情况。本系统可以按照康复师指定的康复计划，对患者的动作完成度进行定量定性的评估。及时向患者反馈其动作完成的质量，提醒督促患者努力达到康复师所设定的专业标准，并在过程中持续鼓励患者，使患者保持较高的积极性。

系统以达到三级康复水平的脑卒中患者为例，设置了站位扣手上举、坐位扣手上举、深蹲、站位扣手左/右平举、坐位扣手左/右平举、左/右踢腿等动作及其相应标准[16]。如表 2-1 所示。当患者在做出标准动作后，系统会将其计入有效动作；当患者做出不标准动作时，系统会将不标准部位的角度与对应标准角度的通过语言反馈给患者。

表 2-1 脑卒中患者的康复训练动作

动作示意图			
名称	站位扣手上举	坐位扣手上举	深蹲
描述	站位状态双手交叉扣手，双臂伸直自腹部向头部运动	坐位状态双手交叉扣手，双臂伸直自腹部向头部运动	双脚分开与肩同宽，双脚脚尖冲外，膝盖不要超过脚尖
目的	患侧肩关节前屈	患侧肩关节前屈	下肢肌群锻炼
关键肢体	上臂 下臂	上臂 下臂	上臂、下臂 大腿、小腿
动作示意图			
名称	站位扣手左/右平举	坐位扣手左/右平举	左/右踢腿
描述	站位双手交叉扣手，双臂伸直后向左/右运动。	坐位双手交叉扣手，双臂伸直后向左/右运动。	左/右脚向前至腿伸直
目的	患侧肩关节内收、外展	患侧肩关节内收、外展	患腿屈伸膝控制
关键肢体	右/左上臂	右/左上臂 右/左上臂	左/右大腿 左/右小腿

2.1.3 患者：身体状态评测

本系统设置了身体状态评测功能，能够在每次训练时对患者的血压、心率等能够反映患者身体状况的生理数据进行采集。

同时，系统还可以对患者四肢的活动范围进行评判。患者根据系统指示分别做出：左/右上臂前举、左/右下臂弯曲、左/右大腿前踢、左/右小腿弯曲的动作，系统会计算出患者肢体活动的角度。夹角选取如表 2-2 所示。患者的身体状态数据会上传至云端供康复师参考。

表 2-2 四肢活动范围夹角选取

动作	选取的夹角
左/右上臂前举	大臂与竖直线的夹角
左/右下臂弯曲	小臂与大臂夹角的补角
左/右大腿前踢	大腿与竖直线的夹角
左/右小腿弯曲	小腿与大腿夹角的补角

2.1.4 异常状态监测

为保障患者训练时的安全，系统监测患者的跌倒行为以及患者的血压、心率等生理数据。系统监测到患者跌倒或者患者的生理数据处于异常范围时立刻向家人或者社区医疗服务人员发送短信呼救，为患者的安全提供保障。

2.1.5 人机交互

本系统采用刷卡或人脸识别登录，面向康复师和患者分别设计了对应的 UI 界面，如图 2-2 所示。用户可通过触摸屏或者语音操控系统。

图 2-2 系统 UI 界面

2.2 软件流程

系统开始运行时，康复师或者病人采用刷卡或人脸识别进行注册与登录。

当系统检测到用户是康复师时，进入康复师操作界面。康复师可以根据不同患者的不同需要，自行录制新的康复动作并选择该动作需要着重关注的身体部位，系统将识别、分析该动作特征。同时，康复师可以随时查看患者训练数据、身体机能情况以及系统对患者每个动作的完成情况，动态调整患者的康复训练方案，并对患者进行留言鼓励。

当系统检测到用户是患者时，进入患者操作界面。患者通过健康数据测量来检测自己的心率、血压等生理数据情况；通过选择身体状态评测来测量自己的四肢与躯体的活动角度；

通过选择训练方案进行锻炼。锻炼过程中，患者可以根据系统的反馈来实现动作的纠正。同时，系统实时监测患者是否摔倒，患者一旦摔倒系统将立即向家人或社区医疗人员报警。患者的身体数据和锻炼数据可以上传至云端，经过汇总处理后反馈给康复师，供康复师根据患者情况调整锻炼方案。软件流程如图 2-3 所示。

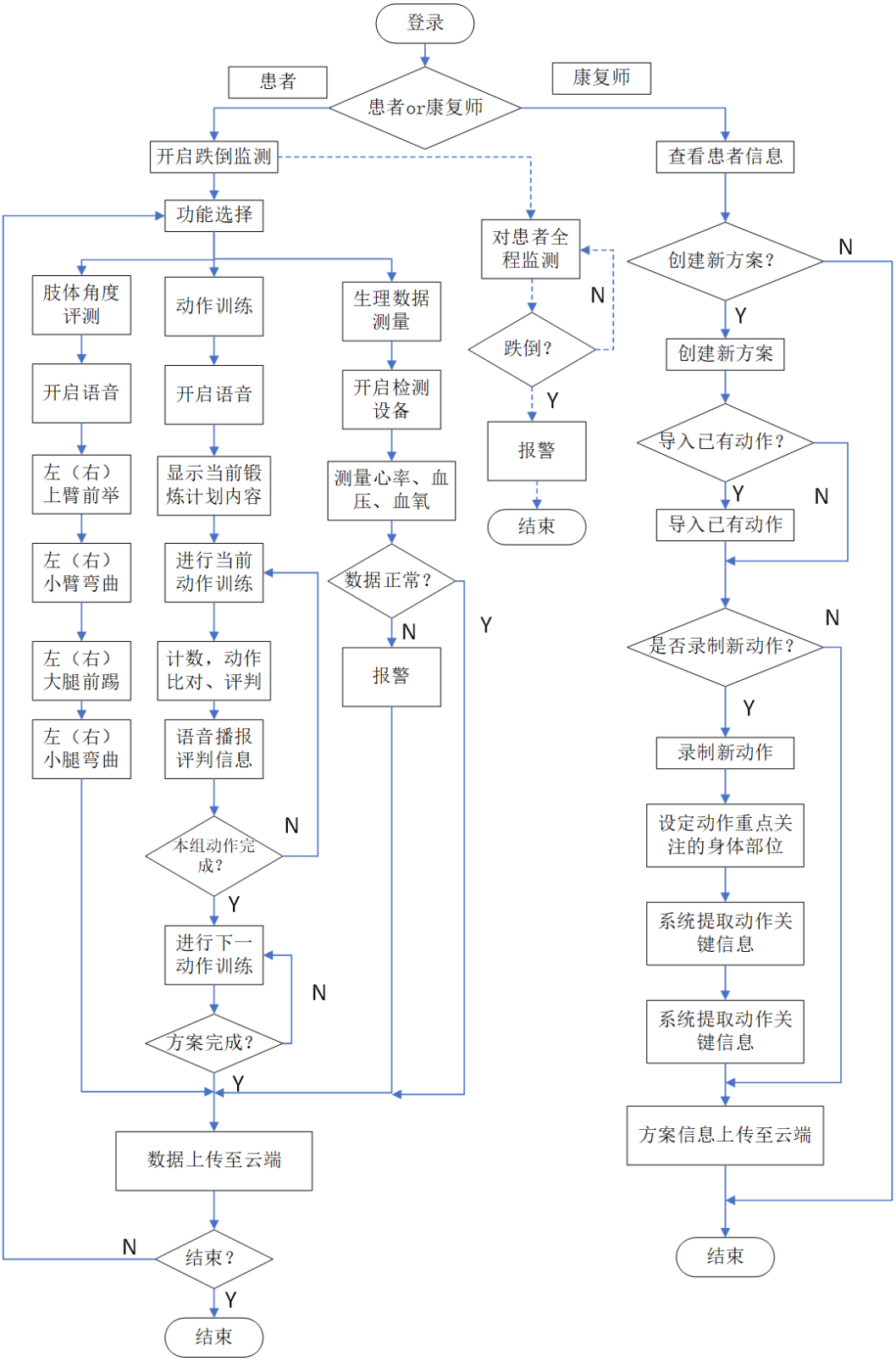


图 2-3 系统软件流程图

2.3 硬件组成与硬件框图

系统以第 11 代英特尔酷睿处理器的边缘计算主机（GNS-V40）为核心。通过外接数字血压检测模块，实现对患者生理数据的采集并传输至 GNS-V40 进行处理。系统通过摄像头实现康复动作识别、跌倒检测与人脸识别。系统另设一个触摸显示屏、音响设备、语音识别模块，进行人机交互。实物图如图 2-4 所示。硬件框图如图 2-5 所示。

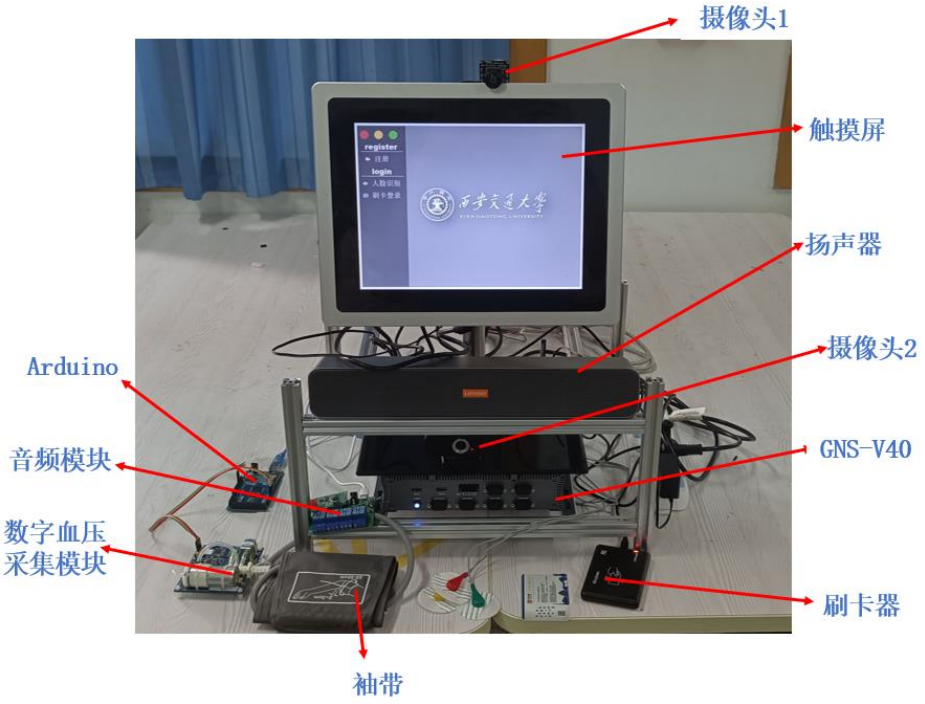


图 2-4 系统实物图

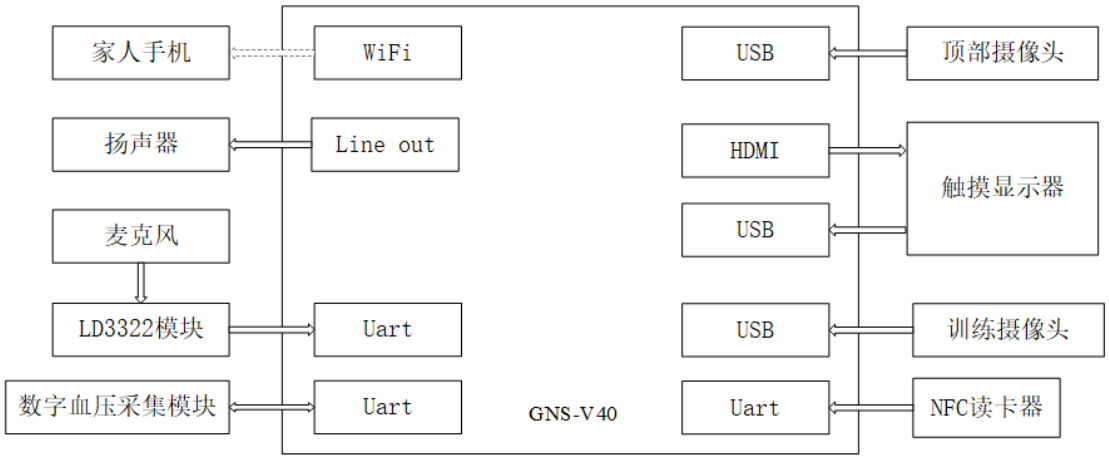


图 2-5 硬件框图

第三章 系统设计原理

系统在进行康复动作姿势纠正时，采用 DTW 算法将患者动作与标准动作对齐在进行比对，通过二者的比对来评判患者动作的标准性；采用 Face_recognition 库实现人脸识别；采用 UNV-LD3322 嵌入式音频模块，实现系统与患者或康复师的语音交互；通过分析摔倒特征判断患者是否摔倒；采用数字血压模块实现患者生理数据的测量。具体原理如下：

3.1 康复动作姿势纠正

系统选用 MediaPipe 库获取 33 个人体关键点数据，舍弃冗余点后，转化为 13 个骨架向量。获取患者视频流数据后，采用 DTW 算法与康复师提供的标准动作对齐。通过比较二者关键帧的人体骨架向量的差异完成对病人所做动作的评价与纠正。

3.1.1MediaPipe 介绍

MediaPipe 是一款由 Google 开发并开源的数据流处理机器学习应用开发框架。它是一个基于图的数据处理管线，用于构建使用了多种形式的数据源，如视频、音频、传感器数据以及任何时间序列数据。MediaPipe 可跨平台，并支持移动端 GPU 加速。它能够将机器学习任务构建为一个图形的模块表示的数据流管道，也包括推理模型和流媒体处理功能。它可以实现物体检测、自拍分割、头发分割、人脸检测、手部检测、运动追踪等许多功能。

MediaPipe 库基于 BlazePose 进行人体姿态估计[17]。BlazePose 的模型架构分为推理管道（Inference pipeline）、人体检测器（Person detector）、拓扑结构（Topology）、神经网络架构构（Neural network architecture）和对齐与遮挡增强（Alignment and occlusions augmentation）。

推理管道中采用了一个检测-跟踪装置,如图 3-1。跟踪器可以预测关键点坐标、当前帧上的人的存在状态以及当前帧的姿态兴趣区域（ROI）。当跟踪器指示没有人存在时，该装置会在下一帧上重新运行检测器网络。

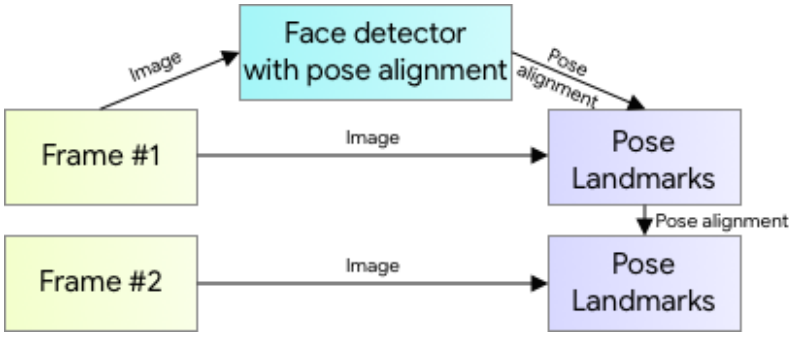


图 3-1 检测-跟踪装置

人体检测器选用面部检测作为人体检测的特征，因为在多数情况下神经网络中有关躯干位置的最强信号是具有高对比度和外观变化较少的人的脸部。该面部检测器可预测其他特定的人的对齐参数：人臀部的中点、整个人的外接圆大小、人体倾斜程度。

拓扑结构基于 BlazeFace、BlazePalm 和 Coco 使用的数据集，在人体上使用 33 个点，如图 3-2 所示。与 OpenPose 和 Kinect 拓扑相比，其只在面部、手和脚上使用了最少数量的关键点来估计后续模型目标区域的旋转、大小和位置。

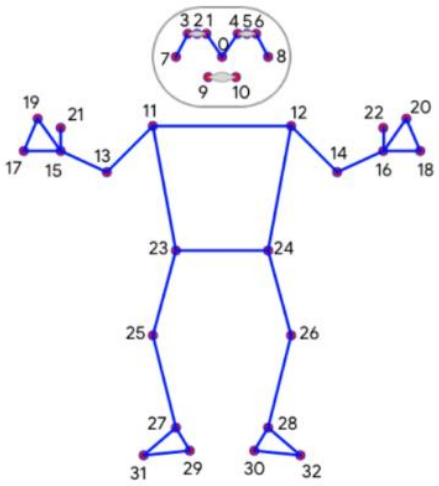


图 3-2 拓扑结构

神经网络架构采用热图、偏移量和回归相结合的方法，如图 3-3。神经网络在训练阶段使用热图和偏移损失，并在运行推理之前从模型中删除相应的输出层。利用网络中所有阶段之间的跳连接来实现高级功能和低级功能之间的平衡。来自回归编码器的梯度不会传播回受热图训练的特征。这样不仅改善了热图预测，而且还大大提高了坐标回归精度。

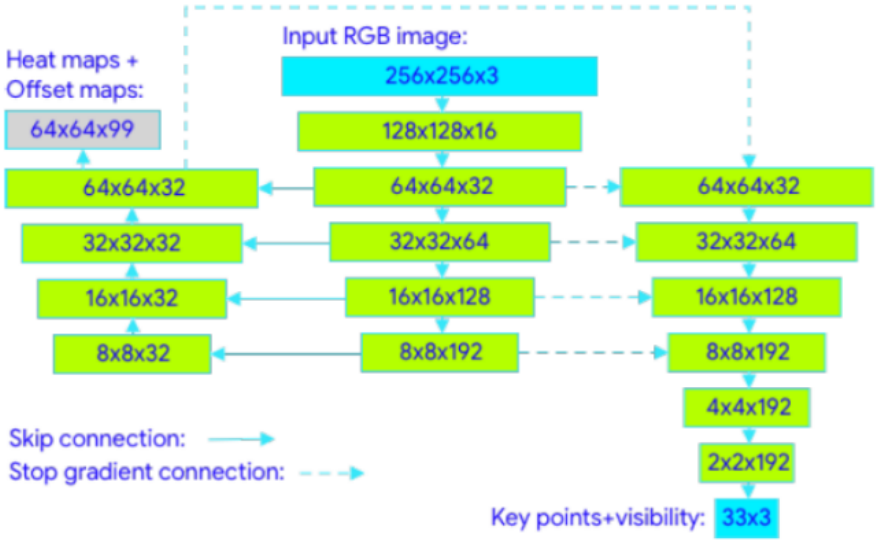


图 3-3 神经网络架构

在对齐与遮挡增强中，为了支持对不可见点的预测，BlazePose 在训练过程中通过填充各种颜色的随机矩形来模拟遮挡，并引入了每点可见度分类器。该分类器可以判断是否遮挡了特定点以及区分位置预测是否错误。通过以上处理，即使在发生严重遮挡的情况下（如仅上身），该模型也可以持续跟踪人。

3.1.2 数据采集与处理

系统基于 MediaPipe 库得到每帧图像中人体的 33 个关键点的三维坐标。为了减少冗余特征的干扰，使关键点坐标更能反映出人体运动的显著特征，本团队去除了左右眼、左右耳、嘴巴、左右手、左右脚等共 19 个点，保留了 14 个点的坐标。取同一帧中相邻两点两点坐标之差为骨骼向量，骨骼向量可以反映出每个肢体活动的角度和幅度信息，更精确的刻画出康复训练动作。人体三维骨骼向量如图 3-4 所示，本系统返回的人体骨骼向量如图 3-5 所示。

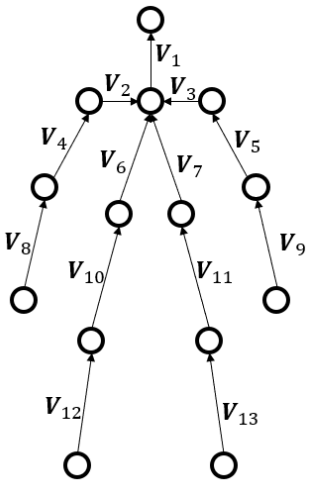


图 3-4 人体骨骼向量

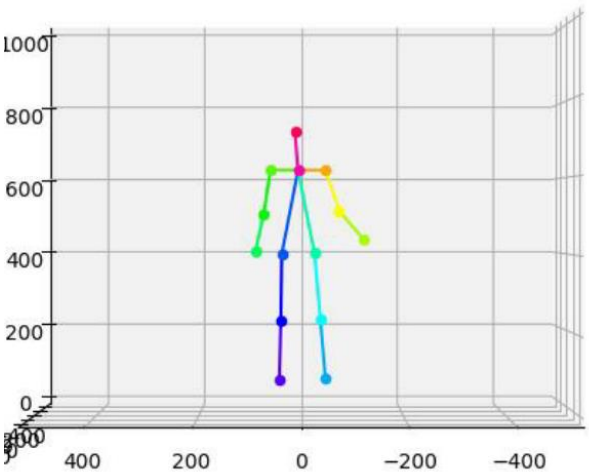


图 3-5 系统返回的骨骼向量

3.1.3 DTW 算法

人体在做动作时，每个关键点的变化具有时序关系。为了评测患者做出的动作与标准动作的区别，系统中采用动态时间规整算法（DTW）来衡量这两个长度不一致的时间序列的相似度[18]。

首先假设有两条序列 Q 和 C，他们的长度分别是 n 和 m，即：

$$Q = q_1, q_2, \cdots, q_n \ ; \ C = c_1, c_2, \cdots, c_m$$

采用一个矩阵A_{n*m}来对比两个序列，warping 路径的第 k 个元素表示为$w_k = (i,j)_k$，i 与 j 表示两个序列中对齐的两个点q_i与c_j。

Warping 路径的约束条件：

1）边界条件：$w_1 = (1,1)$和$w_k = (m,n)$,即两条序列首尾必须匹配。

2）连续性：如果$w_k = (a,b)$且$w_{k-1} = (a',b')$，则必须满足$a - a' \le 1$且$b - b' \le 1$。即在匹配过程中多对一和一对多的情况只能匹配周围一个时间步的情况，不能跨过某个点去匹配，只能和自己相邻的点对齐。

3）单调性：如果$w_k = (a,b)$且$w_{k-1} = (a',b')$，则必须满足$a - a' \ge 0$且$b - b' \ge 0$。即 warping 路径随时间单调递增。

Warping 路径的目标为：

$$DTW(Q,C) = min \sqrt{\sum_{k=1}^{K} w_k} \Big/ K \tag{1}$$

为了得到上述最短距离，定义累积距离：

$$\gamma(i,\ j) = d(q_i,\ c_j) + min\{\gamma(i-1,\ j-1),\ \gamma(i-1,\ j),\gamma(i,\ j-1)\} \tag{2}$$

这条路径可以通过动态规划算法得到。

3.1.4 比对标准姿势与患者姿势

系统应用 DTW 算法，将标准动作与患者动作对齐后，分析康复师指定的某些关键帧中标准动作和患者动作形成的人体骨骼向量所成夹角的差别。并向患者做出反馈。整体流程如图 3-6 所示。

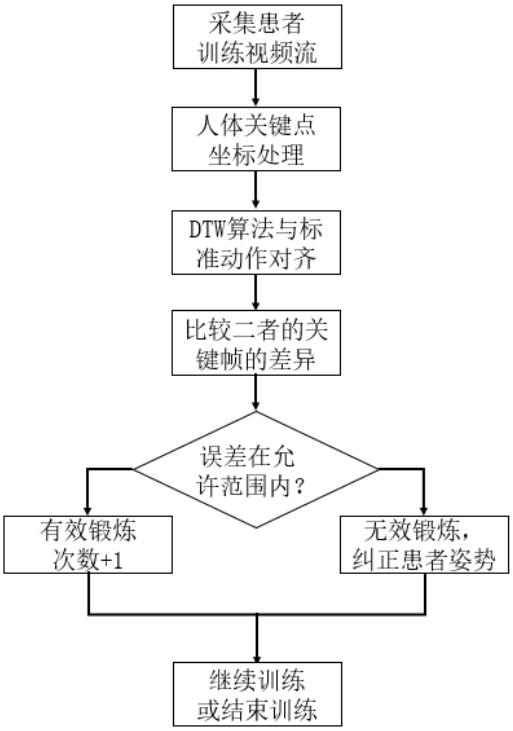


图 3-6　患者姿势纠正流程

3.2 人脸识别

系统利用 face_recognition 库实现人脸识别。Face_recognition 项目是世界上最简洁的人脸识别库，可以使用 Python 和命令行工具提取、识别、操作人脸。它是基于业内领先的 C++开源库 dlib 中的深度学习模型，用 Labeled Faces in the Wild 人脸数据集进行测试，有高达 99.38%的准确率[19]。具体实现过程如下：

（1）人脸检测

通过方向梯度直方图(HOG)来检测人脸位置。首先将图片灰度化，因为色彩对于找到人脸位置并无明显作用，接着计算图像中各像素的梯度。从而提取图像的特征，获取人脸位置，将人脸部分的图像切割出来。

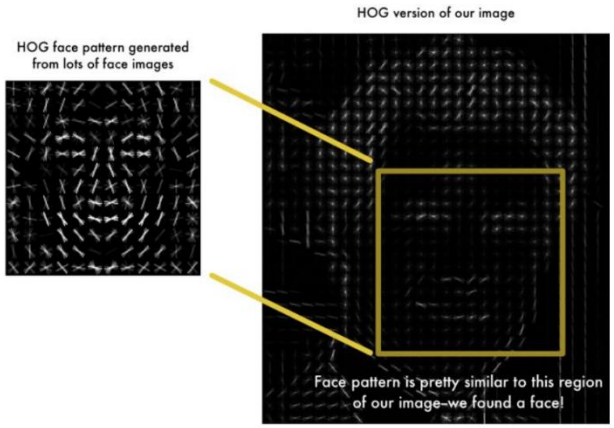


图 3-7　人脸方向梯度直方图

（2）人脸对齐。为了方便给人脸编码，需要将人脸对齐成同一种标准的形状。Dlib 有专门的函数和模型，能够实现人脸 68 个特征点的定位。通过 Dlib 找到特征点后就可以

通过图像的几何变换（仿射、旋转、缩放），使各个特征点对齐（将眼睛、嘴等部位移到相同位置）。

（3）人脸编码。基于深度 CNN 网络，不断迭代训练，减小类内距离，增大类间距离。通过该神经网络将输入的脸部图像生成为 128 维的预测值。

（4）识别身份。预先将所有人的连放入人脸库中，全部用上述的神经网络编码为 128 维并保存。识别时，将人脸预测为 128 维的向量后，采用 SVM 或 KNN 分类器与人脸库中的数据进行比对，从而生成人的身份。

3.3 语音交互

康复师录制动作或患者进行训练时，系统采用按键、触摸屏等方式收发指令很不方便。语音交互在跨空间场景下更为方便快捷高效。因此系统中选用 UNV-LD3322 嵌入式音频模块，实现系统与患者或康复师的语音交互，方便完成动作录制和训练。

UNV-LD3322 主控板集成了功率放大器、麦克风等模块，为客户提供超低成本的离线语音识别方案，可满足智能穿戴设备、蓝牙室内导航、运动健身、智能工业、信息安全、零售支付、数据传输、楼宇自动化、安防、智能家居等物联网应用等多种场景需求。实物图如图 3-8 所示。

图 3-8 UNV-LD3322 嵌入式音频模块

从经过预处理的声音信号提取声音特征，选用 GMM-HMM 构架对声音进行训练，再采用 N 元语言模型找出声音中词与词间的关系，最终通过解码完成声音识别。流程如图 3-9 所示。

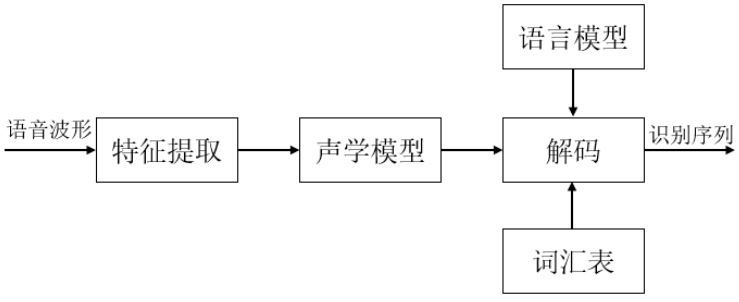


图 3-9　语音识别整体流程

语音识别的具体原理[20]为：

（1）声学信号预处理。首先对采集信号进行滤波与平滑操作，通过分帧加窗操作将连续信号用不同长度的采集窗口分成独立的频域稳定的部分以便于分析。再通过短时能量（同一帧内信号变化的幅度）与短时平均过零率（同一帧内采样信号经过 0 的次数）来对输入信号的起止点进行判断。

（2）声学特征提取。将经过预处理的声学信号进行频域变换后提取特征参数用于语音识别，参数选取应满足易计算、代表性强、参数分量的耦合小。目前主流研究最常用到的特征参数有：线性预测倒谱系数(LPCC)和 Mel 倒谱系数(MFCC)。两种特征参数在倒谱域上对语音信号进行操作，前者以发声模型作为出发点，利用 LPC 技术求倒谱系数。后者则模拟听觉模型，把语音经过滤波器组模型的输出做为声学特征，然后利用离散傅里叶变换(DFT)进行变换。

（3）声学模型：GMM-HMM 构架

声学模型选用隐马尔科夫模型（HMM）进行训练，采用连续的高斯混合模型（GMM）来对状态输出密度函数进行刻画，因此构成了 GMM-HMM 构架。

高斯混合模型（GMM）是单一高斯概率密度函数的延伸，能够平滑地近似任意形状的密度分布。它最明显的性质是它的多模态。这中性质使得混合高斯模型可以描述很多显示出多模态性质的物理数据，如语音数据。一个连续随机变量 X 的混合高斯分布的概率密度函数为：

$$f(x)=\sum_{m=1}^{M}\frac{c_m}{\sqrt{2\pi\sigma_m}}e^{-\frac{1}{2}\left(\frac{x-\mu_m}{\sigma_m}\right)}=\sum_{m=1}^{M}c_m N(x;\mu_m,\sigma_m^2),\ -\infty<x<+\infty;\sigma_m>0;c_m>0 \quad (3)$$

将上面的公式推广到多变量的多远混合高斯分布，就得到了语音识别上使用的混合高斯模型，其联合概率密度函数的形式如下：

$$\mathrm{f(x)}=\sum_{\mathrm{m=1}}^{\mathrm{M}}\frac{\mathrm{c_m}}{(2\pi)^{\mathrm{D/2}}|\Sigma_{\mathrm{m}}|^{1/2}}\exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{m}})^{\mathrm{T}}\sum_{\mathrm{m}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{m}})]=\sum_{\mathbf{m=1}}^{\mathbf{M}}\mathbf{c_m}\mathbf{N}(\mathbf{x};\boldsymbol{\mu}_{\mathbf{m}},\boldsymbol{\Sigma}_{\mathbf{m}}) \quad (\mathbf{4})$$

在得到混合高斯模型的形式后，采用最大期望值算法（EM）进行参数估计，公式如下：

$$c_m^{(j+1)}=\frac{1}{N}\sum_{t=1}^{N}h_m^{(j)}(t) \quad (5)$$

$$\boldsymbol{\mu}_m^{(j+1)}=\frac{\sum_{t=1}^{N}h_m^{(j)}(t)\boldsymbol{x}^{(t)}}{\sum_{t=1}^{N}h_m^{(j)}(t)} \quad (6)$$

$$\Sigma_m^{(j+1)}=\frac{\sum_{t=1}^{N}h_m^{(j)}\left[\boldsymbol{x}^{(t)}-\boldsymbol{\mu}_m^{(j)}\right]\left[\boldsymbol{x}^{(t)}-\boldsymbol{\mu}_m^{(j)}\right]^T}{\sum_{t=1}^{N}h_m^{(j)}(t)} \quad (7)$$

其中 j 是当前迭代轮数。$X^{(t)}$为 t 时刻的特征向量。

GMM 参数通过 EM 算法进行估计，可以使其在训练数据上生成语音观察特征的概率最大化。

隐马尔可夫模型（HMM）在马尔科夫链的基础上进行了扩展，用一个观测的概率分布与马尔可夫链上的每个状态进行对应，引入了双重随机性。它能够描述语音信号中不平稳但有规律可学习的空间变量，具有顺序排列的马尔可夫状态，使得模型能够分段处理短时平稳的语音特征，并以此来逼近全局非平稳的语音特征序列。

隐马尔可夫模型主要又三部分组成。对于状态序列q_1，q_2，…，q_{T}，：

（1）转移概率矩阵$A=[a_{ij}]$，i，$j\in[1,N]$,描述马尔可夫链状态间的跳转概率：$a_{ij}=P(q_t=j|q_{t-1}=i)$；

（2）马尔可夫链的初始概率：$\pi=[\pi_i]$，$i\epsilon[1,N]$，其中$\pi_i=P$（$q_1=i$）；

（3）每个状态的观察概率分布$b_i(o_t)=P(o_t|q_t=i)$，采用 GMM 模型来描述状态的观察概率分布。此时公式可以表述为：

$$b_i(\boldsymbol{o_t})=\sum_{m=1}^{M}\frac{c_{i,m}}{(2\pi)^{D/2}\left|\boldsymbol{\Sigma}_{i,m}\right|^{1/2}}exp\left[-\frac{1}{2}\left(\boldsymbol{o_t}-\boldsymbol{\mu}_{i,m}\right)^T\boldsymbol{\Sigma}_{i,m}^{-1}\left(\boldsymbol{o_t}-\boldsymbol{\mu}_{i,m}\right)\right] \tag{8}$$

隐马尔可夫模型的参数通过 Baum-Welch 算法（在 HMM 上 EM 算法的推广）进行估计。

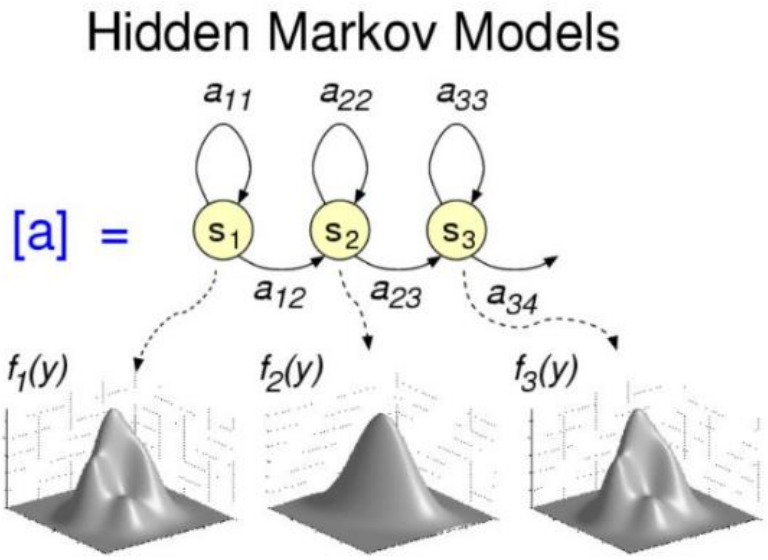


图 3-10　隐马尔可夫模型

（4）语言模型

语言模型着重描述词与词在排列结构上的内在联系。本系统采用了 N 元语言模型（N-gram Language Model），它假设当前在给定上文环境下，当前此的概率只与前 N-1 个词相关。因此词序列w_1，…，w_m的概率$P(w_1,\cdots,w_m)$可以表示为

$$P\left(w_i|w_{i-(n-1)},\cdots,w_{i-1}\right)=\frac{count\left(w_{i-(n-1)},\cdots,w_{i-1},w_i\right)}{count\left(w_{i-(n-1)},\cdots,w_{i-1}\right)} \tag{9}$$

（5）解码

解码器一定程度上使用预生成的有限状态变换器(Finite State Transducer, FST)作为预加载的静态解码图。采用语言模型(G)，词汇表(L)，上下文相关信息(C)，隐马尔可夫模型(H)四个部分分别构建为标准的有限状态变换器，再通过标准的有限状态变换器操作将他们组合起来，构建一个从上下文相关音素子状态到词的变换器。同时，可以对预先构建的有限状态变换器进行预优化，合并和剪掉不必要的部分，使得搜索空间变得更加合理。

3.4 跌倒检测

系统基于 MediaPipe 库提取人体骨骼关键点的位置，通过分析关键点的坐标设计跌倒监测算法，当检测到病人在锻炼过程中跌倒后，即时向社区医疗人员或家人远程报警，对病人进行及时的救护，保障病人的安全。流程包括数据采集与分析、判断是否满足摔倒特征、询问患者是否摔倒、判定患者是否摔倒，如图 3-11 所示：

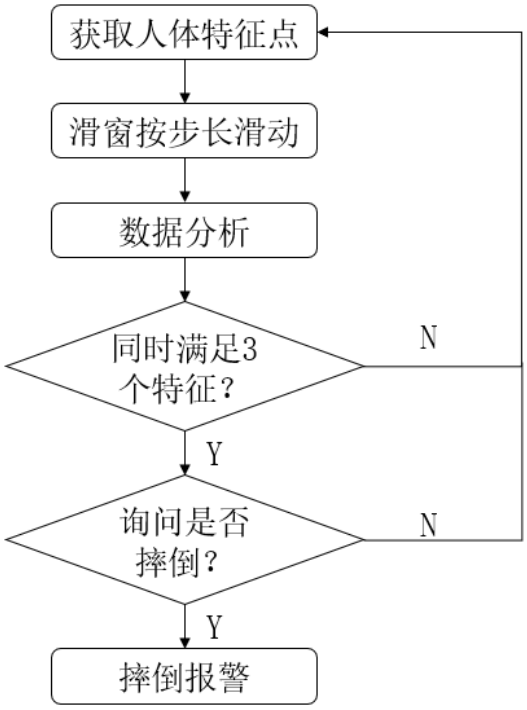


图 3-11 跌倒检测流程图

3.4.1 处理视频流

系统利用 MediaPipe 库逐帧返回人体骨骼 33 个关键点，点的坐标形式采用像素坐标系。滑窗从初始帧开始向后滑动，滑窗大小为 25 帧，滑窗滑动的步长为 5 帧。

3.4.2 设置特征条件

对摔倒、躺下、下蹲、站立等动作的坐标特征进行分析与区分。如果滑窗中的坐标同时满足如下特征，则判断病人可能摔倒。

（1）特征一：人体关键点y坐标比较大

由于采用像素坐标系，因此越接近地面，y 坐标就越大。当病人摔倒时，人体贴近地面，y 坐标较大。在具体判断时，若滑窗中有多于 10 帧的人体关键点的 y 坐标的平均值 $\bar{y} = \sum_{i=0}^{32} y_i > y_threshold$时，则认为满足该条件。

（2）特征二：头部下降速度比较大

对滑窗中取每一帧的头部 12 个点 y 坐标的平均值$\overline{y_head}$，利用最小二乘法求出滑窗中$\left(t, \overline{y_head}_t\right)$（1≤t≤25）的斜率 v，把斜率 v 当作速度。当速度 v>v_threshold 时，则认为满足该条件。

（3）特征三：人体x坐标范围过大

病人摔倒在地时，关键点的 x 坐标范围比较大。若满足滑窗中有多于 15 帧的 x 坐标的方差 x_var>x_var_threshold，且有多于 5 帧的 x_range=x_max-x_min>x_range_threshold, 则认为满足该条件。

3.4.3 阈值选取

为了选取合理的 y_threshold、v_threshold、x_var_threshold 和 x_range_threshold，我们对人在各种状态下$\bar{y}$、v、x_var 和 x_range 的范围进行了测试，数据见表 6-2。因此，我们取 y_threshold=335，v_threshold=20，x_var_threshold=3500，x_range_threshold=170。

表 3-1 各种状态下各个特征的范围

活动类型	$\bar{y}$	v	x_var	x_range
日常站立	255	0	160	55
下蹲	315	23	300	62
躺	340	16	3800	186
摔倒	344	22	3950	189
阈值	335	20	3500	170

3.4.4 判定是否摔倒

当姿态特征满足上述跌倒检测特征时，判断病人可能跌倒。此时通过语音询问病人是否摔倒，如果接收到“是”的回应，或一分钟内病人无应答，则自动向社区医疗人员或家人远程报警。

3.5 数字血压检测模块

数字血压检测模块是基于示波法降压测量原理开发而成的多参数输出模块。其主要由血压模块、气泵、电磁阀、放气阀、气路管件和袖带等组成，如图 3-12 所示

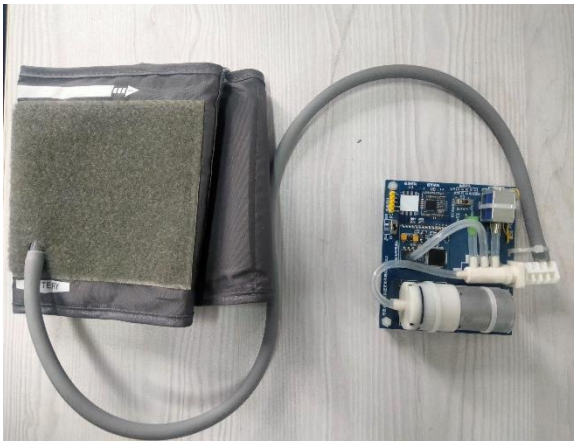


图 3-12 数字血压检测模块实物图

3.7.1 技术参数

（1）工作条件

·温度：5℃~40℃

·相对湿度：18%~80%

·大气压力：80kPa~105kPa

·电源电压：DC 6V；3.7V 锂电池（充满电在 3.7V~4.2V 区间使用）

（2）量程/采集范围：0~39.9kPa（299mmHg）

（3）最大功耗：不大于 4.5VA

（4）压力传感器准确性：误差为±0.4kPa（3mmHg）

（5）分辨率：0.1kPa（1mmHg）

3.7.2 工作原理

示波法降压测量法原理如下：血压计使用气泵对袖带进行充气加压，利用充气袖带压迫动脉血管，使动脉血管处于完全闭阻状态。随后开启放气阀，使袖带内压力缓慢下降。随着袖带内压力的下降，动脉血管呈完全阻闭—渐开—全开的变化过程。降压过程中，动脉内压

力振幅大小变化趋势如图 3-13 所示。

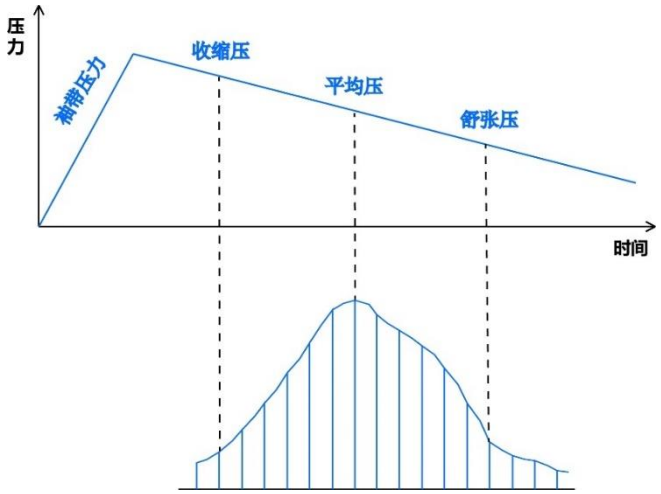


图 3-13 动脉内压力振幅大小变化趋势

压力传感器采集连续变化的袖带内压力，将其转化为数字信号送入 MCU，通过软件辨别动脉血流受阻过程中相应压力点，根据经验累积的软件算法得出人体的舒张压、收缩压和平均压。

第四章 系统测试

测试环境:西安交通大学工程坊

测试设备：智能康复训练辅助系统（本作品）

测试项目：（1）血压、心率测试；（2）计算肢体角度测试；（3）锻炼动作反馈情况测试（4）语音口令测试（5）跌倒检测测试（6）其他

4.1 血压、心率采集测试

对三位测试者各进行 3 次健康数据采集，测试数据如表 4-1 所示。

表 4-1 血压、心率采集数据统计表

测试对象	第一次			第二次			第三次		
	高压	低压	心率	高压	低压	心率	高压	低压	心率
1	109	78	67	108	80	74	116	78	70
2	113	76	78	117	74	69	108	70	71
3	120	84	71	114	71	67	118	69	65

4.2 患者肢体极限活动角度测试

测试人员跟随系统的指令分别对左右手臂、左右腿的角度进行测试。统计测试者姿势的真实角度与系统计算得出的角度的差异。测试数据如表 4-2 所示。

表 4-2 肢体活动角度测试数据

测量部位	实际角度	测量角度							平均误差
		1	**2**	**3**	**4**	**5**	**6**	**7**	
左上臂前举	90	91	90	92	82	88	87	88	2.6
	45	45	36	37	42	44	46	47	3.4
	120	124	121	125	108	115	124	119	4.6
右上臂前举	90	97	84	84	89	90	82	88	4.3
	45	45	44	34	46	47	36	39	4.3
	120	116	117	118	135	120	114	119	4.4
左下臂弯曲	90	92	97	84	93	94	85	89	4.0
	120	126	130	124	114	126	123	115	5.7
	60	63	53	55	58	54	50	67	5.7
右下臂弯曲	90	92	87	84	85	90	85	84	3.9
	120	126	130	124	125	112	115	110	6.9
	60	53	50	53	55	51	62	52	6.9
左大腿前踢	0	1	3	4	0	4	6	5	3.3
	30	35	38	32	25	22	26	24	5.4
	60	66	62	50	51	54	58	67	6.0

续表 4-2

右大腿前踢	0	3	5	6	4	1	0	6	3.6
	30	35	40	33	33	26	21	30	4.9
	60	63	68	61	62	70	51	52	5.6
左小腿弯曲	0	5	3	3	8	7	5	3	3.3
	60	54	67	62	66	66	52	53	6.0
	90	95	96	96	98	81	90	84	5.7
右小腿弯曲	0	8	2	2	3	6	9	5	5.0
	60	64	62	60	52	58	54	56	3.7
	90	84	87	89	88	85	95	96	4.0

4.3 患者锻炼动作反馈情况测试

测试人员作为患者跟随系统指令运动，分别做出标准姿势的动作和不标准的动作，检查系统的反馈情况。对每个动作均进行了 60 次测试，其中标准动作样本与不标准动作样本各为 30 次。所得数据中，除深蹲外系统反馈准确率均在 95%以上，深蹲动作对患者肢体角度精确性要求较高，包含 10 个角度限制条件，不排除测试正确样本过程中测试人员动作不达标，导致系统判定为错误动作的可能。统计情况如表 4-3 所示。

表 4-3　锻炼动作系统反馈情况

动作	总样本数	真阳样本	真阴样本	准确率/%
站位扣手上举	60	30	30	100
坐位扣手上举	60	29	30	98.3
深蹲	60	27	29	93.3
站位扣手左平举	60	28	29	95
站位扣手右平举	60	29	29	96.7
坐位扣手左平举	60	30	30	100
坐位扣手右平举	60	30	29	98.3
左踢腿	60	28	30	96.7
右踢腿	60	29	30	98.3

4.4 语音口令测试

三名测试人员分别对系统设定的口令测试 20 次，统计系统正确和错误识别次数。测试口令包括“开始录制”，“重新录制”，“重新播放”，“开始测量”，“继续测量”，“重新测量”，“开始训练”，“继续训练”，“下一个动作”，“救命”。测试数据如表 4-4 所示。

表 4-4　语音口令测试数据

测试口令	开始录制	重新录制	重新播放	开始测量	继续测量	重新测量	开始训练	继续训练	下一动作	救命
测试次数	60	60	60	60	60	60	60	60	60	60
正确识别	60	58	55	60	59	55	59	59	58	59
比例	100	96.7	91.7	100	98.3	91.7	98.3	98.3	96.7	98.3

4.5 跌倒检测测试

三位测试者分别做出横向跌倒、纵向跌倒、深蹲、弯腰、缓慢躺下动作各 20 次，统计系统正确与错误识别次数。测试数据如表 4-5 所示。

表 4-5　跌倒检测数据统计表

测试动作	深蹲	弯腰	缓慢躺下	横向跌倒	纵向跌倒
测试次数	60	60	60	60	60
识别为跌倒的次数	0	0	3	56	55
误识率	0	0	5%	6.67%	8.3%

4.6 其他功能测试

表 4-6　其他功能测试表

功能	录制动作	查看患者信息	人脸识别	刷卡器
状态	正常	正常	正常	正常

第五章 总结与展望

5.1 系统特色

（1）关注运动复健人群，缓解复健资源短缺情况

目前我国复健医疗资源短缺，疫情形势使得复健更为困难。系统从患者角度出发，为患者提供了在家进行康复动作锻炼的机会。同时也方便康复师即时获取患者数据，为每位患者量身打造特定锻炼方案。

（2）系统可扩展性强

目前已有的产品大多只能提供特定的锻炼动作供相应需求的病人使用。而本系统提供了康复师自定义动作的功能。康复师录制标准动作后只需要根据系统指引设定相应的动作标准，便可将该动作提供给相应病人进行参考与训练。动作扩展流程方便快捷，系统对扩展后的动作的指导与纠正清晰准确。

（3）采用边缘计算模式

本系统将患者数据采集、姿势纠正、语音识别、人脸检测、跌倒检测等算法依托性能优异的第 11 代英特尔酷睿处理器的边缘计算主机（GNS-V40）上实现。云端上负责多个病人的数据存储与分析。边缘计算缩短了信息传递时间并使得 GNS-V40 的资源得到了充分利用。

（4）友好的人机交互

系统提供了界面美观，操控方便的 GUI 交互界面，并根据不同人群、场景的需求辅以语音交互与人脸识别。标准动作视频框及实时锻炼视频流反馈框的设置使患者在锻炼过程能通过视觉感知更直观的进行动作完成及规范。用户操作流程简便、人机交互友好使得用户更容易接受本产品。

5.2 前景展望

本系统依托性能优异的第 11 代英特尔酷睿处理器的边缘计算主机（GNS-V40），运用深度学习、人工智能和边缘计算等先进的技术与概念，深度融入时代发展潮流。能够准确地捕捉、刻画人体动作，即时纠正患者动作不规范的地方。病人异常情况检测等辅助功能使得本系统功能更加完善，患者的居家安全更加有保障。该系统在复健医疗资源短缺、疫情影响下将会使得更加病患者受益，发展前景广阔。

同时，该系统目前只能做到对包含单项动作视频流的处理。难以对包含多个动作、多种动作的视频流进行处理。后续我们会针对这一问题对系统进一步优化，使得患者的训练过程更加流畅。

参考文献

[1]孟涵,孙薇婷,陈家瑞,蒋沛,李云霞,马原,任松青,宋朋,田璐,徐克拉,夏沫,谭华峰,宗润薇,周山山,张颖,陈世益,康绍勇,孟坡.2020-2021 年中国运动康复产业白皮书[J].中国运动医学杂志,2021,40(09):749-756.DOI:10.16038/j.1000-6710.2021.09.016.

[2]盛晗,邵圣文,王惠琴,姚梅琪,陈金花.脑卒中患者康复锻炼依从性动态变化的研究[J].中华护理杂志,2016,51(06):712-715.

[3] Touillet A,Guesdon H,Bosser G,et al.Assessment of compliance with prescribed activity by hemiplegic stroke patients after an exercise programme and physical activity education[J].Ann Phys Rehabil Med,2010,53(4):250,257-265.

[4]H. Yu, S. Cang and Y. Wang, "A review of sensor selection, sensor devices and sensor deployment for wearable sensor-based human activity recognition systems," 2016 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA), 2016, pp. 250-257, doi: 10.1109/SKIMA.2016.7916228.

[5]Saisakul Chernbumroong,Shuang Cang,Hongnian Yu. A practical multi-sensor activity recognition system for home-based care[J]. Decision Support Systems,2014,66.

[6]Juan C Fraile,Javier Pérez-Turiel,Enrique Baeyens,Pablo Viñas,Rubén Alonso,Alejandro Cuadrado,Manuel Franco-Martín,Esther Parra,Laureano Ayuso,Francisco García-Bravo,Félix Nieto,Lipsa Laurentiu. E2Rebot: A robotic platform for upper limb rehabilitation in patients with neuromotor disability[J]. Advances in Mechanical Engineering,2016,8(8).

[7]K. Li,Y. Hao,X. Hu,D. Xie,X. Li,H. Zheng,Y. Fu,Y. Chen,Y. Zheng. The effect of sensorimotor training performed by carers on home-based rehabilitation in stroke patients[J]. Physiotherapy,2015,101.

[8]T. -Y. Lin, C. -H. Hsieh and J. -D. Lee, "A Kinect-Based System for Physical Rehabilitation: Utilizing Tai Chi Exercises to Improve Movement Disorders in Patients with Balance Ability," 2013 7th Asia Modelling Symposium, 2013, pp. 149-153, doi: 10.1109/AMS.2013.29.

[9]崔银洁.临床康复学多元化教学方法总结与策略分析[J].按摩与康复医学,2018,9(17):82-84.DOI:10.19787/j.issn.1008-1879.2018.17.043.

[10]刘林,伍平平,熊巍.虚拟超市认知康复训练系统开发关键技术研究[J].图学学报,2014,35(01):105-109.

[11]李海军,潘晓露,李一民,吴刚,罗明刚.平行双目视觉系统中深度图像的生成与分析[J].计算机与数字工程,2006(02):50-51+56.

[12]沈天毓,王计平,郭立泉,白启帆,张惠钧,王守岩,熊大曦.利用可穿戴设备对帕金森病患者运动功能进行量化评估[J].生物医学工程学杂志,2018,35(02):206-213.

[13]周鑫. 下肢康复机器人的训练规划与康复效果评估[D].哈尔滨工程大学,2011.

[14]李晋,谢晓添,姬庆庆,韩子鹤,闻文,马硕.基于虚拟现实技术的手功能康复训练系统设计与实现[J].电脑与信息技术,2017,25(01):35-37.DOI:10.19414/j.cnki.1005-1228.2017.01.012.

[15]王清波,虞成,陈天笑,杨攀,高云,袁杰.基于 OpenVINO 的情绪识别反馈康复训练系统设计[J].中国医学装备,2021,18(01):102-105.

[16] 闫航,陈刚,佟瑶,姬波,胡北辰.基于姿态估计与 GRU 网络的人体康复动作识别[J].计算机工程,2021,47(01):12-20.DOI:10.19678/j.issn.1000-3428.0058201.

[17] https://arxiv.org/abs/2006.10204

[18] 杨一鸣,潘嵘,潘嘉林,杨强,李磊.时间序列分类问题的算法比较[J].计算机学报,2007(08):1259-1266.

[19] https://zhuanlan.zhihu.com/p/99927894
[20] https://xueqiu.com/3075605687/130174699

附录

main.py

```
from gui_module.main_ui import run_ui
from pose_module.human_pose_estimation import fall_down_main
from speech_module import receive_word
import sys
from multiprocessing import Pool
if __name__ == '__main__':
    # 配置进程池
    pool = Pool(processes=2)
    # 开启语音识别（子进程）
    pool.apply_async(receive_word, (1,))
    # 开启跌倒检测（子进程）
    pool.apply_async(fall_down_main, (2,))
    # 开启交互界面
    run_ui()
    sys.exit(0)
```

human_pose_estimation.py

```
from speech_module import wait_word, wait_anything

import cv2
from fastdtw import fastdtw
import mediapipe as mp
from matplotlib import pyplot as plt
import math
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import os
from scipy.spatial.distance import euclidean
import time
from win32com.client import Dispatch
from winsound import Beep

# directory where model will be downloaded
baseDir = os.path.relpath(os.path.dirname(os.path.abspath(__file__)), start=os.getcwd())

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_holistic = mp.solutions.holistic
```

```
capNum = 1
colors = ((255, 0, 85), (255, 0, 170), (85, 255, 0), (255, 170, 0), (0, 255, 0), (255, 255, 0), (0, 255, 85),
          (170, 255, 0), (0, 85, 255), (0, 255, 170), (0, 0, 255), (0, 255, 255), (85, 0, 255), (0, 170, 255))
default_skeleton = ((0, 1), (1, 2), (1, 3), (1, 8), (1, 9), (2, 4), (3, 5),
                    (4, 6), (5, 7), (8, 10), (9, 11), (10, 12), (11, 13))

vector_skeleton = ((0, 1), (2, 3), (2, 4), (3, 5), (4, 6), (5, 7),
                   (8, 9), (8, 10), (9, 11), (10, 12), (11, 13))
angle_skeleton = ((0, 2), (1, 2), (0, 3), (1, 3), (2, 4), (3, 5),
                  (0, 7), (6, 7), (0, 8), (6, 8), (7, 9), (8, 10))
angles_name = ['左上臂前举', '左上臂侧举', '右上臂前举', '右上臂侧举', '左下臂弯曲', '右下臂弯曲',
               '左大腿前踢', '左大腿侧踢', '右大腿前踢', '右大腿侧踢', '左小腿弯曲', '右小腿弯曲']


def get_world_positions(pose_3d):
    if pose_3d.any():
        [x_w, y_w, z_w] = pose_3d
        x_mean = np.mean([x for x in x_w if abs(x) > 1])
        y_mean = np.mean([y for y in y_w if abs(y) > 1])
        x_w = [x - x_mean if x else 0 for x in x_w]
        y_w = [(y_mean - y) / 4 if y else 0 for y in y_w]
        z_w = [1000 - z if z else 0 for z in z_w]
        pose_3d = np.array([x_w, y_w, z_w])
    return pose_3d


def run_pose_estimation(_image, _holistic, threshold=0.5, use_show2d=False):
    _image.flags.writeable = False
    _image = cv2.cvtColor(_image, cv2.COLOR_BGR2RGB)
    results = _holistic.process(_image)
    # print(results.pose_landmarks)
    #  画图
    if use_show2d:
        _image.flags.writeable = True
        _image = cv2.cvtColor(_image, cv2.COLOR_RGB2BGR)
        mp_drawing.draw_landmarks(_image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,

landmark_drawing_spec=mp_drawing_styles.get_default_pose_landmarks_style())
        cv2.namedWindow('MediaPipe Holistic', cv2.WINDOW_NORMAL)
```

```
            cv2.setWindowProperty('MediaPipe Holistic', cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
            cv2.imshow('MediaPipe Holistic', _image)
            cv2.waitKey(1)
        x_list = []
        y_list = []
        z_list = []
        if results.pose_landmarks:
            landmarks = results.pose_landmarks.landmark
            pose_3d = landmarks[:1] + landmarks[11:17] + landmarks[23:29]
            for landmark in pose_3d:
                if landmark.visibility > threshold:
                    x_list.append(landmark.x)
                    y_list.append(landmark.y)
                    z_list.append(-landmark.z)
                else:
                    x_list.append(0)
                    y_list.append(0)
                    z_list.append(0)
            shoulder_points_num = len(np.array(x_list[1:3]).nonzero()[0])
            if shoulder_points_num != 0:
                x_list.insert(1, (x_list[1] + x_list[2]) / shoulder_points_num)
                y_list.insert(1, (y_list[1] + y_list[2]) / shoulder_points_num)
                z_list.insert(1, (z_list[1] + z_list[2]) / shoulder_points_num)
            else:
                x_list.insert(1, 0)
                y_list.insert(1, 0)
                z_list.insert(1, 0)
        pose_3d = np.array([x_list, z_list, y_list]) * 1000
        pose_3d = get_world_positions(pose_3d)
        return pose_3d


def show_pose_3d(pose_3d, s='title', use_show_3d=True):
        if use_show_3d:
            plt.ion()
            plt.clf()
            _fig = plt.gcf()
            _axes = _fig.gca(projection='3d')
            _axes.text2D(0.5, 1, s, transform=_axes.transAxes)

            # 设置画布大小
            _fig.set_size_inches(8, 6)
            _fig.set_tight_layout(True)
```

```
        # 设置视角
        _axes.view_init(0, 0)
        # 设置坐标轴刻度
        _axes.set_xlim3d(xmin=-500, xmax=500)
        _axes.set_ylim3d(ymin=-500, ymax=500)
        _axes.set_zlim3d(zmin=0, zmax=1000)
        if pose_3d.any():
            # Draw joints.
            for i in range(len(pose_3d[2])):
                if pose_3d[2][i] != 0:
                    _axes.scatter3D(pose_3d[0][i], pose_3d[1][i], pose_3d[2][i],
color=np.array(colors[i]) / 256)
            # Draw limbs.
            for i, j in default_skeleton:
                if pose_3d[2][i] * pose_3d[2][j] != 0:
                    _axes.plot3D(np.array([pose_3d[0][i], pose_3d[0][j]]),
                                 np.array([pose_3d[1][i], pose_3d[1][j]]),
                                 np.array([pose_3d[2][i], pose_3d[2][j]]),
                                 color=np.array(colors[j]) / 256)
        plt.pause(0.001)
        plt.ioff()


# 获取滑窗
def get_slide_window(_image, _holistic, slide_window, time_step, use_show2d=False,
use_show_3d=False):
    pose_3d = run_pose_estimation(_image, _holistic, threshold=0.5,
use_show2d=use_show2d)
    show_pose_3d(pose_3d, use_show_3d=use_show_3d)
    if pose_3d.any():
        pose_3d = pose_3d.T.reshape(-1)
        if len(slide_window) == 0:
            slide_window = pose_3d[np.newaxis, :]
        elif len(slide_window) < time_step:
            slide_window = np.r_[slide_window, pose_3d[np.newaxis, :]]
        else:
            slide_window = np.r_[slide_window[1:], pose_3d[np.newaxis, :]]
    return slide_window


# 写入关键点信息
def write_positions_file(pose_name,
                         interval_frames_num=3,  # 间隔帧数
                         time_step=20,  # 时间步长
```

```
                                time_width=5,  # 平滑区间
                                pose_library_path='pose_library',
                                use_show_2d=False,
                                use_show_3d=True):
    slide_window = np.array([])

    # 等待开始采集指令
    wait_word('start_record')

    # 开启摄像头
    _cap = cv2.VideoCapture(capNum)

    with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5)
as _holistic:
        Dispatch("SAPI.SpVoice").Speak(f'正在录制{pose_name}')
        Beep(800, 500)
        print("****************************** 开 始 录 制
******************************")
        start_time = time.time()
        count = 0
        if not os.path.exists(f'{baseDir}/{pose_library_path}'):
            os.makedirs(f'{baseDir}/{pose_library_path}')
        positions_file = open(f'{baseDir}/{pose_library_path}/{pose_name}.txt', 'w')

        while _cap.isOpened():
            _suc, _image = _cap.read()
            if not _suc:
                print("Ignoring empty camera frame.")
                continue

            slide_window = get_slide_window(_image, _holistic, slide_window, time_width,
use_show2d=use_show_2d)
            cv2.waitKey(1)

            if count >= time_width:
                print(count - time_width)
                # 每列非零值平均
                pose_3d = np.array([int(x / y) if y else 0 for x, y in
                                    zip(slide_window.sum(axis=0), (slide_window !=
0).sum(axis=0))])
                if (count - time_width) % interval_frames_num == 0:
                    _content = ' '.join(str(point) for point in pose_3d.reshape(-1))
                    positions_file.write(_content + '\n')
            count += 1
```

```
                if count >= time_step * interval_frames_num + time_width:
                    print(time.time() - start_time)
                    Dispatch("SAPI.SpVoice").Speak('结束')
                    print("****************************** 结 束 录 制
******************************")
                    positions_file.close()
                    if use_show_2d:
                        cv2.destroyWindow('MediaPipe Holistic')
                    positions_file = open(f'{baseDir}/{pose_library_path}/{pose_name}.txt',
'r')
                    while True:
                        # 播放动作
                        read_positions_file(pose_name)
                        plt.close()
                        print('重新播放，重新录制')
                        while True:
                            with open(f'speech_flag.txt', 'r') as speech_flag:
                                flag = speech_flag.readline()
                            with open(f'speech_flag.txt', 'w') as speech_flag:
                                speech_flag.write('___')
                            if flag == '___':
                                time.sleep(1)
                                continue
                            if flag == 'end' or 'again_record' or 'replay_record':
                                break

                        if flag == 'end':
                            break
                        elif flag == 'again_record':
                            positions_file =
open(f'{baseDir}/{pose_library_path}/{pose_name}.txt', 'w')
                            count = 0
                            Dispatch("SAPI.SpVoice").Speak(f'正在录制{pose_name}')
                            Beep(800, 500)
                            print("****************************** 开 始 录 制
******************************")
                            start_time = time.time()
                            break
                        else:
                            continue
                    if flag == 'end':
                        break

            _cap.release()
```

```
    return flag


# 读取关键的信息
def read_positions_file(pose_name, pose_library_path='pose_library'):
    print(f'正在播放{pose_name}')
    with open(f'{baseDir}/{pose_library_path}/{pose_name}.txt', 'r') as positions_file:
        positions_file.seek(0)
        for index, _position in enumerate(positions_file):
            _position = np.array(list(map(int, _position.rstrip().split(' ')))).reshape(14, -1).T
            show_pose_3d(_position, s=str(index), use_show_3d=True)
            time.sleep(0.05)
        Beep(800, 1000)
        plt.close()


# 制作数据库
def make_dataset(interval_frames_num=1, time_step=30, time_width=5, use_show_2d=False,
use_show_3d=False):
    while True:
        _plan = input("请输入动作名，建议使用（输入空格退出）：")
        if _plan == ' ':
            break
        flag = write_positions_file(_plan, interval_frames_num, time_step, time_width,
                                    use_show_2d=use_show_2d,
use_show_3d=use_show_3d)
        if flag == 'end':
            break


# 获取骨架向量
def get_skeleton_vector(_positions):
    _positions = _positions.reshape([-1, 14, 3])
    vector = np.array([])
    for _skeleton in vector_skeleton:
        temp = _positions[:, _skeleton[0]] - _positions[:, _skeleton[1]]
        if not vector.any():
            vector = temp[np.newaxis, :]
        else:
            vector = np.row_stack((vector, temp[np.newaxis, :]))
    vector_skeletons = vector.transpose((1, 0, 2))
    return vector_skeletons
```

```
# 计算骨架角度
def get_skeleton_angle(vector_frame, angle_num):
    _skl1, _skl2 = angle_skeleton[angle_num]
    a, b = vector_frame[_skl1], vector_frame[_skl2]
    if _skl1 == 0:
        a = np.array([0, 0, 1])
    if a.any() and b.any():
        _angle = np.arccos(a.dot(b) / (np.linalg.norm(a) * np.linalg.norm(b))) * 180 / np.pi
        if angle_num == 1 or angle_num == 3 or angle_num == 7 or angle_num == 9:
            _angle = abs(_angle - 90)
    else:
        _angle = None
    return _angle


# 运行训练辅助
def run_pose_evaluate(pose_name, max_count, template_frame=15, angles_num=None,
max_error=None, flag=None,
                      pose_library_path='pose_library', time_interval=4, time_step=160,
use_show_2d=False,
                      use_show_3d=False):
    count = 0
    valid_count = 0

    if flag == 'end':
        return flag, f'{valid_count}/{count}/{max_count}'

    # 开启摄像头
    _cap = cv2.VideoCapture(capNum)

    # 获取动作模板
    if not os.path.exists(f'{baseDir}/{pose_library_path}/{pose_name}.txt'):
        print(f'找不到动作[{pose_name}]')
        return None
    _template = np.loadtxt(f'{baseDir}/{pose_library_path}/{pose_name}.txt', dtype=int)
    vector_template = get_skeleton_vector(_template)

    while True:
        slide_window = np.array([])
        _valid = np.array([True] * 12)

        Dispatch("SAPI.SpVoice").Speak(f'{pose_name}第{count + 1}次训练')
        Beep(800, 500)
        print("******************************          开          始
```

```
**********")

        start_time = time.time()
        with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as _holistic:
            while _cap.isOpened() and time.time() - start_time < time_interval:
                _suc, _image = _cap.read()
                if not _suc:
                    print("Ignoring empty camera frame.")
                    continue
                slide_window = get_slide_window(_image, _holistic, slide_window, time_step,
                                                use_show2d=use_show_2d, use_show_3d=use_show_3d)
                if not slide_window.any():
                    start_time = time.time()
        if use_show_2d:
            cv2.destroyWindow('MediaPipe Holistic')
        if use_show_3d:
            plt.close()

        if not slide_window.any():
            continue
        print("********** 结 束 **********")
        print(time.time() - start_time)
        Beep(800, 500)

        # DTW 计算距离,还应加上判断动作是否标准的部分及记录关键部位角度的部分
        vector_slide = get_skeleton_vector(slide_window)
        _distance, _map = fastdtw(vector_template.reshape(-1, 11 * 3), vector_slide.reshape(-1, 11 * 3), dist=euclidean)
        slide_frame = int(np.array([y for (x, y) in _map if x == template_frame]).mean())
        for index, angle_num in enumerate(angles_num):
            slide_angle = get_skeleton_angle(vector_slide[slide_frame], angle_num)
            template_angle = get_skeleton_angle(vector_template[template_frame], angle_num)
            print(angles_name[angle_num], slide_angle, template_angle)
            if not slide_angle:
                Dispatch("SAPI.SpVoice").Speak(f'未检测到{angles_name[angle_num]}角度')
                _valid[angle_num] = False
            elif not template_angle:
                print(f'模板中未检测测到{angles_name[angle_num]}角度')
```

```
                elif abs(slide_angle - template_angle) > max_error[index]:
                    Dispatch("SAPI.SpVoice").Speak(
                        f'{angles_name[angle_num]}角度为{int(slide_angle)}度,与标准角度{int(template_angle)}偏离')
                    _valid[angle_num] = False

            for index, pose_3d in enumerate(slide_window):
                show_pose_3d(pose_3d.reshape(-1, 3).T, s=str(index), use_show_3d=True)
                time.sleep(0.01)
            plt.close()

            flag = wait_anything()
            if flag == 'next_train' or flag == 'end':
                # 下一动作或结束训练
                count += 1
                # 如果动作合格
                if _valid.all():
                    valid_count += 1
                break
            elif flag == 'continue':
                # 继续本动作的下一次
                count += 1
                # 如果动作合格
                if _valid.all():
                    valid_count += 1
                if count >= max_count:
                    Dispatch("SAPI.SpVoice").Speak(f'{pose_name}已到达{max_count}次训练')
                    flag = 'next_train'
                    break
                continue
            else:
                # 重新本动作的这一次
                continue

        # 关闭摄像头
        _cap.release()

        return flag, f'{valid_count}/{count}/{max_count}'


def run_train_plan(plan_name, plans_dir='./pose_module/plans', use_show_2d=False):
    out = None
    flag = None
```

```
    if not os.path.exists(f'{plans_dir}/{plan_name}'):
        print(f'找不到方案{plan_name[:-4]}')
        return -1

    # 等待训练开始指令
    wait_word('start_train')

    with open(f'{plans_dir}/{plan_name}', 'r') as plan_file:
        for line in plan_file:
            words = line.rstrip().split(' ')
            flag, ret = run_pose_evaluate(words[0], int(words[1]), int(words[2]),
                                          list(map(lambda x: ord(x) - ord('a'),
words[3])), list(map(int, words[4:])),
                                          flag=flag, use_show_2d=use_show_2d)
            if not out:
                out = ret
            else:
                out = f'{out} {ret}'
    return out


# 采集人体各部位极限角度
def collect_pose_limitation(angles_num=None, time_width=5):
    if not angles_num:
        angles_num = [0, 2, 4, 5, 6, 8, 10, 11]

    limitation_angles = np.array([])

    # 开启摄像头
    _cap = cv2.VideoCapture(capNum)

    for angle_num in angles_num:
        while True:
            slide_window = np.array([])

            Dispatch("SAPI.SpVoice").Speak(f'请将{angles_name[angle_num]}移至极限
位置')
            # 等待测量开始指令
            wait_word('start_measure')
            Beep(800, 500)
            print("******************************** 开 始
********************************")
            with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as _holistic:
```

```
                while _cap.isOpened():
                    _suc, _image = _cap.read()
                    if not _suc:
                        print("Ignoring empty camera frame.")
                        continue
                    print(_suc)
                    slide_window = get_slide_window(_image, _holistic, slide_window,
time_width)
                    if len(slide_window) >= time_width:
                        break
            slide_mean = np.array(
                [int(x / y) if y else 0 for x, y in zip(slide_window.sum(axis=0),
(slide_window != 0).sum(axis=0))])[
                                 np.newaxis, :]
            slide_vector = get_skeleton_vector(slide_mean)
            limitation_angle = get_skeleton_angle(slide_vector.reshape(-1, 3), angle_num)
            # if angle_num <= 1:
            #     limitation_angle[0] = 90 - limitation_angle
            print(limitation_angle)
            if limitation_angle:
                Dispatch("SAPI.SpVoice").Speak(f'{angles_name[angle_num]} 角 度 为
{int(limitation_angle)}度')
            else:
                Dispatch("SAPI.SpVoice").Speak(f'未识别到{angles_name[angle_num]}
角度，请重测')

            show_pose_3d(slide_mean.reshape(-1, 3).T)
            plt.pause(2)
            plt.close()

            flag = wait_anything()
            if flag == 'again_measure':
                continue
            elif flag == 'continue' or flag == 'end':
                break
        limitation_angles = np.append(limitation_angles, [limitation_angle])
        if flag == 'end':
            break
    print(limitation_angles)
    return limitation_angles


def fall_down_check(slide_window):
    x_var_threshold = 1000
```

```
    v_threshold = 7.7
    average_threshold = 300
    y = slide_window[:, 1:2].reshape(25, 33)
    x = slide_window[:, 0:1].reshape(25, 33)
    count_y_zero_index = np.where(~y.any(axis=1))[0]  # 一行全为零的下标
    count_y_zero = len(count_y_zero_index)  # 滑窗中行全为零的数目
    x_var = np.var(x, axis=1)
    y_ave = np.mean(y, axis=1)  # 均值
    y_head = y[:, 0:13]  # 取头部 12 个点的 y 坐标
    y_head_ave = np.mean(y_head, axis=1)  # 均值
    x_range = np.max(x, axis=1) - np.min(x, axis=1)  # 横坐标的范围
    if 0 < count_y_zero <= 5:
        not_zero = np.where(y.any(axis=1))[0][0]  # 取第一个非零行的下标
        for i in count_y_zero_index:
            x_var[i] = x_var[not_zero]
            y_ave[i] = y_ave[not_zero]
            y_head_ave[i] = y_head_ave[not_zero]
            x_range[i] = x_range[not_zero]
    b, v = fall_velocity(y_head_ave)  # 最小二乘求斜率当作速度
    if v != "#" and count_y_zero <= 5:
        count_y_ave = np.count_nonzero(y_ave > average_threshold)  # 滑窗中大于某个值的 y 平均重心数
        count_x_var = np.count_nonzero(x_var > x_var_threshold)  # 滑窗中大于某个值的 x 坐标方差数
        count_x_range = np.count_nonzero(x_range > 150)  # 滑窗中大于某个值的 x 坐标的范围数
        if count_y_ave >= 10 and v > v_threshold and count_x_var >= 15 and count_x_range >= 5:  # 判断是否满足摔倒条件
            print("摔倒了")
        else:
            print("没摔倒")


def fall_velocity(y):  # 求速度，最小二乘求斜率
    try:
        N = len(y)
        y_index = np.array(range(0, N))
        sumx = sum(y_index)
        sumy = sum(y)
        sumx2 = sum(y_index ** 2)
        sumxy = sum(y_index * y)
        A = np.mat([[N, sumx], [sumx, sumx2]])
        b = np.array([sumy, sumxy])
        return np.linalg.solve(A, b)  # 依次返回截距、斜率
```

```
    except:
        return "#", "#"


def fall_down_main(x):
    print(1)
    mpPose = mp.solutions.pose  # 姿态识别方法
    pose = mpPose.Pose(static_image_mode=False,  # 静态图模式，False 代表置信度高时继续跟踪，True 代表实时跟踪检测新的结果
                       smooth_landmarks=True,  # 平滑，一般为 True
                       min_detection_confidence=0.5,  # 检测置信度
                       min_tracking_confidence=0.5)  # 跟踪置信度
    mpDraw = mp.solutions.drawing_utils
    cap = cv2.VideoCapture(0)
    slide_stride = 5  # 滑窗步长
    slide_window = np.zeros((825, 2))  # 初始化滑窗，可采集连续 25 帧，25*33 个点=825
    stride_cache = np.zeros((slide_stride * 33, 2))  # 将步长长度的人体关键点存入 stride_cache
    flag = 0
    while True:
        pose_2d = np.zeros((33, 2))  # 存放每一帧人体关键点信息
        success, img = cap.read()
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        results = pose.process(imgRGB)
        if results.pose_landmarks:
            mpDraw.draw_landmarks(img, results.pose_landmarks, mpPose.POSE_CONNECTIONS)
            for index, lm in enumerate(results.pose_landmarks.landmark):
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                pose_2d[index, 0] = cx
                pose_2d[index, 1] = cy
                cv2.circle(img, (cx, cy), 3, (0, 255, 0), cv2.FILLED)
            if flag < slide_stride:  # 将一个步长的坐标存入 stride_cache
                stride_cache[flag * 33:(flag + 1) * 33, :] = pose_2d
                flag += 1
            else:  # 收集到一个步长的数据后，将一个步长的数据推入滑窗
                slide_window[0:825 - slide_stride * 33, :] = slide_window[slide_stride * 33:, :]
                slide_window[825 - slide_stride * 33:, :] = stride_cache
                flag = 0
                fall_down_check(slide_window)
        # cv2.imshow('image', img)
        if cv2.waitKey(10) & 0xFF == 27:
```

```
                break
        cap.release()
        cv2.destroyAllWindows()
```

speech_module.py

```
import os
from pynput import mouse
import serial
import time

baseDir = os.path.relpath(os.path.dirname(os.path.abspath(__file__)), start=os.getcwd())


def receive_word(x=None):
    # 开启长按屏幕监听
    mouse.Listener(on_click=on_click).start()

    # 打开串口
    _ser = serial.Serial(port='COM10', baudrate=9600, bytesize=8, parity='N', stopbits=1, timeout=5)

    if _ser.isOpen():
        print('语音已打开')

    with open(f'speech_flag.txt', 'w', encoding='utf-8') as speech_flag:
        speech_flag.write('___')
    while True:
        word = b''
        _bit = _ser.read()
        if _bit == b'A':
            while True:
                _bit = _ser.read()
                if _bit == b'Z':
                    with open(f'{baseDir}/speech_flag.txt', 'w', encoding='utf-8') as speech_flag:
                        speech_flag.write(word.decode('ascii'))
                    print(word.decode('ascii'))
                    break
                word += _bit

        with open(f'{baseDir}/speech_flag.txt', 'r', encoding='utf-8') as speech_flag:
            word = speech_flag.readline()
            if word == 'close':
                break
```

```
    _ser.close()    # 关闭串口


def wait_anything():
    while True:
        with open(f'speech_flag.txt', 'r') as speech_flag:
            flag = speech_flag.readline()
        with open(f'speech_flag.txt', 'w') as speech_flag:
            speech_flag.write('___')
        if flag == '___':
            time.sleep(1)
            continue
        return flag


def wait_word(_word):
    while True:
        if wait_anything() == _word:
            break


def on_click(x, y, button, pressed):
    if str(button) == 'Button.right' and pressed:
        with open(f'{baseDir}/speech_flag.txt', 'w') as speech_flag:
            print('end')
            speech_flag.write('end')
```

main_ui.py

```
import sys    #
import face_recognition
import cv2
import numpy as np
import win32com.client
import serial
from os import listdir
from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox
from PyQt5.QtCore import pyqtSignal, Qt
from PyQt5.QtGui import *
from datetime import datetime
from time import time
import time
from PIL import Image
from gui_module.usb_reader import *
from gui_module.addr_txt import *
```

```
import pymysql
from gui_module.first_ui import Ui_MainWindow
from gui_module.register_ui import Ui_register
from gui_module.trainer_ui import Ui_MainWindow2_1
from gui_module.doctor_ui import Ui_MainWindow2_2
from gui_module.plan_ui import Ui_MainWindow_plan
from gui_module.information_ui import Ui_MainWindow_information
from pose_module.human_pose_estimation import read_positions_file, write_positions_file, run_train_plan, \
    collect_pose_limitation

os.environ["QT_IM_MODULE"] = "qtvirtualkeyboard"
path_gui = './gui_module/'
connect_mysql = pymysql.connect("此处为数据库基础设置")
arduino_serialPort = "COM11"
arduino_baudRate = 9600


def speak(s):
    print("-->" + s)
    win32com.client.Dispatch("SAPI.SpVoice").Speak(s)


class Arduino:
    def __init__(self):
        self.serialPort = arduino_serialPort
        self.baudRate = arduino_baudRate
        self.ser = serial.Serial(self.serialPort, self.baudRate, timeout=0.01)
        self.ser.set_buffer_size(rx_size=1, tx_size=1)

    def get_hall(self):
        string = ''
        while len(string) == 0:
            str_lines = self.ser.readlines()
            string = self.ser.readline().decode('utf-8')
            try:
                if string == "":
                    string = str_lines[-1].decode('utf-8')
            except:
                pass
        return string[:-2]

    def send(self, send_data):
        self.ser.write(send_data.encode('utf-8'))  # utf-8 编码发送
```

```
    def close_ser(self):
        self.ser.close()


class MyMainWindow(QMainWindow, Ui_MainWindow):  # 继承了 QMainWindow,Ui_MainWindow 的方法和属性
    signal1 = pyqtSignal()
    signal2_1 = pyqtSignal()  # 患者界面
    signal2_2 = pyqtSignal()  # 医生界面

    def __init__(self, parent=None):
        super(MyMainWindow, self).__init__(parent)
        self.setupUi(self)
        self.user_name = ''
        self.user_identity = ''
        self.source = 0
        self.cap = cv2.VideoCapture()
        self.need_record_name1 = ([])  # 识别到的所有人脸
        # 点击按钮跳转函数
        self.left_close.clicked.connect(self.close_window)  # self.left_close = QtWidgets.QPushButton("") # 关闭按钮
        self.left_mini.clicked.connect(self.showMinimized)
        self.left_visit.clicked.connect(self.showMaximized)

        self.pushButton_l2.clicked.connect(self.btn_open_cam_click)
        self.pushButton_l1.clicked.connect(self.signal1_register)  # 跳转到 signal1 信号发出函数
        self.pushButton_l3.clicked.connect(self.login_ic)

        self.show()

    def close_window(self):
        flag2 = self.cap.isOpened()
        if flag2:
            QMessageBox.information(self, 'warning', '请先关闭人脸识别再退出')
        else:
            self.close()

    def btn_open_cam_click(self):  # 人脸识别，打开摄像头 按钮函数
        flag2 = self.cap.isOpened()  # 存储摄像头是否打开
        if not flag2:
            self.cap.open(self.source)
            try:
```

```
                self.pushButton_l2.setText(u' 关闭识别')
                self.show_camera()
            except:
                QMessageBox.about(self, 'warning', '人脸不能正常被打开')
        else:
            self.pushButton_l2.setText(u' 人脸识别')
            self.user_name = ''
            self.clear_all()
            self.cap.release()

    def show_camera(self):
        # 这段代码是 获取 photo 文件夹中 人的信息
        filepath = path_gui + 'photo'
        filename_list = listdir(filepath)
        known_face_names = []
        known_face_encodings = []
        a = 0
        for filename in filename_list:
            a += 1
            QApplication.processEvents()
            if filename.endswith('jpg'):
                known_face_names.append(filename[:-4])  # 把文件名字的后四位.jpg
去掉获取人名
                file_str = path_gui + 'photo' + '/' + filename
                a_images = face_recognition.load_image_file(file_str)
                a_face_encoding = face_recognition.face_encodings(a_images)[0]
                known_face_encodings.append(a_face_encoding)
        process_this_frame = True
        while self.cap.isOpened():
            ret, frame = self.cap.read()
            image = frame[0:200, 200: 400]
            QApplication.processEvents()
            # 改变摄像头图像的大小，图像小，所做的计算就少
            small_frame = cv2.resize(image, (0, 0), fx=SET_SIZE, fy=SET_SIZE)
            # opencv 的图像是 BGR 格式的，而我们需要是的 RGB 格式的，因此需要
进行一个转换。
            rgb_small_frame = small_frame[:, :, ::-1]
            if process_this_frame:
                QApplication.processEvents()
                face_locations = face_recognition.face_locations(rgb_small_frame)
                face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
                face_names = []
                for face_encoding in face_encodings:
```

```
                matches = face_recognition.compare_faces(known_face_encodings, face_encoding,

tolerance=TOLERANCE)
                # 阈值太低容易造成无法成功识别人脸，太高容易造成人脸识别混淆 默认阈值 tolerance 为 0.6
                name = ''
                if True in matches:
                    first_match_index = matches.index(True)
                    name = known_face_names[first_match_index]
                    self.user_name = name
                face_names.append(name)
            cv2.rectangle(frame, (200, 0), (400, 200), (255, 0, 0), 2)
            # cv2.rectangle(frame, (left, top), (right, bottom), (60, 20, 220), 3)
            cv2img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            pilimg = Image.fromarray(cv2img)
            frame = cv2.cvtColor(np.array(pilimg), cv2.COLOR_RGB2BGR)
            if self.cap.isOpened() == False:  # 再次检查，确定到这里摄像头没打开，没有这行的话有可能会跑下面的程序
                break

            show_video = cv2.resize(frame, (800, 494))
            show_video = cv2.cvtColor(show_video, cv2.COLOR_BGR2RGB)  # 这里指的是显示原图
            #opencv 读取图片的样式，不能通过 Qlabel 进行显示，需要转换为 Qimage QImage(uchar * data, int width,
            self.showImage = QImage(show_video.data, show_video.shape[1], show_video.shape[0], QImage.Format_RGB888)
            self.right_label_1.setPixmap(QPixmap.fromImage(self.showImage))
            if self.user_name != '':
                self.incert_information()
                self.signal2_emit()
                break

    def incert_information(self):  # 将登录信息写入 mysql 数据库，并且在屏幕下方显示信息
        con = connect_mysql
        cur = con.cursor()
        sql = 'select * from user where name="%s"' % self.user_name
        res = cur.execute(sql)
        results = cur.fetchall()
        self.user_identity = results[0][2]
        str_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        sql = 'insert into login(name,login_time,identity) values("%s","%s","%s")' % (
```

```
                self.user_name, str_time, self.user_identity)
            cur.execute(sql)
            speak("你好，" + self.user_name)

    def judge_identity(self, judge_name):  # 储存登录人信息到数据库并且确定并返回登录人身份
            con = connect_mysql
            cur = con.cursor()
            sql = 'select identity from user where name="%s"' % judge_name
            cur.execute(sql)
            results = cur.fetchall()

            str_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            sql = 'insert into login(name,login_time,identity) values(%s,%s,%s)'
            cur.execute(sql, (judge_name, str_time, results[0][0]))
            con.commit()
            return results[0][2]

    def login_ic(self):
            write_ic_card()
            self.ic = read_ic_card()
            con = connect_mysql
            cursor = con.cursor()
            sql = 'select * from user where id="%s"' % self.ic
            res = cursor.execute(sql)
            if res == False:
                QMessageBox.about(self, 'warning', '您还未注册，请先注册')
                QMessageBox.about(self, 'warning', '您还未注册，请先注册')
                return
            else:
                results = cursor.fetchall()
                self.user_name = results[0][1]
                self.user_identity = results[0][2]
                str_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

                sql = 'insert into login(name,login_time,identity) values("%s","%s","%s")' % (
                    self.user_name, str_time, self.user_identity)
                cursor.execute(sql)
                speak("你好，" + self.user_name)
                con.commit()
                text_create('user_name', self.user_name)
                self.user_name = ''
                if results[0][2] == '患者':
                    self.signal2_1.emit()  # 患者界面
```

```
                else:
                    self.signal2_2.emit()
                self.pushButton_l2.setText(u'  人脸识别')
                self.clear_all()
                self.cap.release()
                self.close_window()

    def signal1_register(self):
        self.signal1.emit()

    def signal2_emit(self):   #
        # global user_name
        # user_name = self.user_name
        text_create('user_name', self.user_name)
        self.user_name = ''
        if self.user_identity == '患者':
            self.signal2_1.emit()   # 患者界面
        else:
            self.signal2_2.emit()   # 医生界面
        self.pushButton_l2.setText(u'  人脸识别')
        self.clear_all()
        self.cap.release()
        self.close_window()

    def clear_all(self):
        self.right_label_1.setPixmap(QPixmap(""))
        self.user_name = ''
        self.user_identity = ''
        self.left_label_3.setText("")


class MineWindow_register(QMainWindow, Ui_register):
    def __init__(self, parent=None):
        # super(MineWindow, self).__init__(None, Qt.FramelessWindowHint)   # 这句和普通的不一样 因为可以实现无边框
        super(MineWindow_register, self).__init__(parent)
        self.setupUi(self)
        self.pushButton_5.clicked.connect(self.close_clear)
        self.pushButton_25.clicked.connect(self.close_clear)
        self.pushButton_6.clicked.connect(self.showMinimized)
        self.pushButton_24.clicked.connect(self.new_register)
        self.pushButton_ic.clicked.connect(self.ic_reader)
        self.pushButton_face2.clicked.connect(self.photo_face)
```

```
        self.ic = ''

    def close_clear(self):
        linetext = [self.lineEdit_13, self.lineEdit_14, self.lineEdit_15]
        for lineedit in linetext:
            lineedit.clear()  # 首先清空输入
            lineedit.setPlaceholderText("请输入信息")
            lineedit.setAlignment(Qt.AlignCenter)  # 输入的字放到中间
        self.ic = ''
        self.close()

    def ic_reader(self):
        write_ic_card()
        self.ic = read_ic_card()
        self.search_ic()

    def search_ic(self):
        con = connect_mysql
        cursor = con.cursor()
        sql = 'select * from user where id="%s"' % self.ic
        con.commit()
        res = cursor.execute(sql)
        if res != 0:
            QMessageBox.about(self, 'warning', '您已注册，请登录')
            QMessageBox.about(self, 'warning', '您已注册，请登录')
        else:
            pass

    def new_register(self):
        con = connect_mysql
        button = 0  # 当都输入正确的时候写入 配置文件
        name = self.lineEdit_15.text()
        age = self.lineEdit_13.text()
        sex = self.lineEdit_14.text()
        if self.ic != '':
            if name != '':  # name 有输入
                if not (str.isdigit(age) or len(age) != 11):  # 判断是否为数字
                    button = 1
                    QMessageBox.about(self, 'warning', '请输入正确的电话号码')
                if sex != '男' and sex != '女':
                    button = 1
                    QMessageBox.about(self, 'warning', '性别请输入正确')
                if button == 0:
                    cur = con.cursor()
```

```
                    sql = 'insert into user(name,age,sex,identity,id) values("%s","%s","%s","%s","%s")' % (
                        name, int(age), sex, "患者", self.ic)
                    cur.execute(sql)
                    con.commit()
                    sql = "CREATE TABLE IF NOT EXISTS %s(" \
                          "train_time VARCHAR(20)," \
                          "heart_rate VARCHAR(20)," \
                          "O2 VARCHAR(20)," \
                          "blood_pressure VARCHAR(20)," \
                          "muscle_strength VARCHAR(20)," \
                          "train_plan VARCHAR(20)," \
                          "limitation_angles VARCHAR(100)," \
                          "train_out VARCHAR(100)" \
                          ")DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;" % name
                    cursor = con.cursor()
                    cursor.execute(sql)
                    con.commit()

                    QMessageBox.about(self, ' ', '您已成功注册')
                else:
                    QMessageBox.about(self, 'warning', '姓名未输入')
            else:
                QMessageBox.about(self, 'warning', 'ic 卡未输入')

        def photo_face(self):
            self.re_name = self.lineEdit_15.text()
            if self.re_name != '' and self.ic != '':
                video_capture = cv2.VideoCapture(0)
                img_path = path_gui + "./photo/" + self.re_name + ".jpg"
                t4 = time.time()
                self.right_label.setText("还有" + str(t4) + "秒准备开始拍照")
                while True:
                    ret, image = video_capture.read()
                    QApplication.processEvents()
                    image = image[0:400, 250: 750]
                    img_rgb = image[0:200, 65: 265]
                    cv2.rectangle(image, (65, 0), (265, 200), (255, 0, 0), 1)
                    face_location = face_recognition.face_locations(img_rgb)
                    video = cv2.resize(image, (500, 400))
                    video = cv2.cvtColor(video, cv2.COLOR_BGR2RGB)  # 这里指的是显示原图
                    self.video = QImage(video.data, video.shape[1], video.shape[0],
```

```
QImage.Format_RGB888)
                    self.photo_label.setPixmap(QPixmap.fromImage(self.video))
                    now_time = 7 - time.time() + t4
                    if now_time <= 0:
                        self.right_label.setText("已经开始拍照")
                    else:
                        self.right_label.setText("还有" + str(int(now_time)) + "秒准备开始拍照")
                    if face_location != [] and (time.time() - t4) > 7:
                        image = image[1:200, 66: 265]
                        break
                cv2.imencode('.jpg', image)[1].tofile(img_path)
                self.right_label.setText("拍照结束！ ")
                video_capture.release()
                cv2.destroyAllWindows()
                show_video = cv2.resize(image, (400, 400))
                show_video = cv2.cvtColor(show_video, cv2.COLOR_BGR2RGB)  # 这里指的是显示原图
                self.showImage = QImage(show_video.data, show_video.shape[1], show_video.shape[0], QImage.Format_RGB888)
                self.photo_label.setPixmap(QPixmap.fromImage(self.showImage))
            else:
                QMessageBox.about(self, 'warning', '请先输入 ic 卡和姓名')


class MineWindow2_trainer(QMainWindow, Ui_MainWindow2_1):  # 使用信号
    signal_exit = pyqtSignal()

    def __init__(self, parent=None):
        super(MineWindow2_trainer, self).__init__(parent)
        self.setupUi(self)

        # 参数
        self.con = connect_mysql
        self.cur = self.con.cursor()
        self.flag = 0

        # 显示当前用户
        self.user_name = text_read('user_name')
        self.left_label_2.setText(self.user_name)

        self.time = datetime.now().strftime("%Y.%m.%d %H 点")
        self.user_train_plan = ''
        self.train_out = ''
```

```
            self.limitation = ''
            self.O2 = ''
            self.heart_rate = ''
            self.muscle_strength = ''
            self.blood_pressure = ''

            # 各按钮功能
            self.left_close.clicked.connect(self.close_commit)
            self.left_mini.clicked.connect(self.showMinimized)
            self.left_visit.clicked.connect(self.showMaximized)
            self.pushButton_l1.clicked.connect(self.run_train)
            self.pushButton_l2.clicked.connect(self.btn_gbi)
            self.pushButton_l9.clicked.connect(self.return_to_login)
            self.pushButton_l4.clicked.connect(self.pose_limitation)

        def close_commit(self):
            if self.train_out == '' and self.limitation == '' and self.O2 == '' and self.heart_rate == ''
\
                        and self.muscle_strength == '' and self.blood_pressure == '':
                pass
            else:
                sql                                                    =                                                'insert
into %s(train_time,heart_rate,O2,blood_pressure,muscle_strength,train_plan,' \
                          'limitation_angles,train_out)
values("%s","%s","%s","%s","%s","%s","%s","%s")' % (
                                self.user_name,          self.time,          self.heart_rate,          self.O2,
self.blood_pressure, self.muscle_strength,
                                self.user_train_plan, self.limitation, self.train_out)
                self.cur.execute(sql)
                self.con.commit()
            self.close()

        def return_to_login(self):
            if self.train_out == '' and self.limitation == '' and self.O2 == '' and self.heart_rate == ''
\
                        and self.muscle_strength == '' and self.blood_pressure == '':
                pass
            else:
                sql                                                    =                                                'insert
into %s(train_time,heart_rate,O2,blood_pressure,muscle_strength,train_plan,' \
                          'limitation_angles,train_out)
values("%s","%s","%s","%s","%s","%s","%s","%s")' % (
                                self.user_name,          self.time,          self.heart_rate,          self.O2,
self.blood_pressure, self.muscle_strength,
```

```
                    self.user_train_plan, self.limitation, self.train_out)
            self.cur.execute(sql)
            self.con.commit()
        self.close()
        self.signal_exit.emit()  # 登录界面
        self.user_name = ''
        self.left_label_2.setText('')
        self.close()  # 加清屏

    def btn_gbi(self):
        if self.flag == 0:
            try:
                self.ser = Arduino()
                self.right_label_1.setText(u"血压心率测量")
                self.pushButton_l2.setText(u"停止测量")
                self.get_body_information()
            except:
                QMessageBox.about(self, "warning", '测量设备未连接')
        else:
            self.pushButton_l2.setText(u"测量血压心率")
            self.right_label_2.setText("")
            self.right_label_3.setText("")
            self.right_label_1.setText("")
            self.right_label_4.setText("")
            self.right_label_5.setText("")
            self.right_label_6.setText("")
            self.ser.close_ser()
            self.flag = 0
        pass

    def get_body_information(self):
        # 接收 Arduino 测量的身体数据并传回云端
        self.flag = 1
        time.sleep(2)
        self.ser.send('AT+ST:1\r\n')
        QApplication.processEvents()
        while self.flag == 1:
            QApplication.processEvents()
            sleep_time = np.random.random()
            time.sleep(sleep_time)
            string = self.ser.get_hall()
            if string[1] == '0':
                self.right_label_2.setText("实时压力值:")
                self.right_label_3.setText(string[3:-3])
```

```
                elif string[1] == '1':
                    self.right_label_4.setText('收缩压:' + string[3:6])
                    self.right_label_5.setText('舒张压:' + string[7:10])
                    self.right_label_6.setText('心率:' + string[11:-3])
                    self.heart_rate = string[11:-3]
                    self.blood_pressure = string[3:6] + '/' + string[7:10]
                    break


    def pose_limitation(self):
        self.limitation = collect_pose_limitation()


    def run_train(self):
        sql = 'select trainplan from user where name="%s"' % self.user_name
        res = self.cur.execute(sql)
        results = self.cur.fetchall()
        self.user_train_plan = results[0][0]
        self.con.commit()
        if str(self.user_train_plan) == 'None':
            QMessageBox.about(self, 'warning', '未设置训练方案，请联系康复师定制')
        else:
            self.train_out = run_train_plan(self.user_train_plan + '.txt', use_show_2d=True)
            self.time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            # sql = 'insert into %s(train_time,train_plan,train_out) values("%s","%s","%s")' % (
            # self.user_name, self.time, self.user_train_plan, self.train_out)
            # self.cur.execute(sql)
            self.con.commit()

            labels_poses = [self.mid_label_2, self.mid_label_3, self.mid_label_4, self.mid_label_5,
                            self.mid_label_6, self.mid_label_7, self.mid_label_8, self.mid_label_9, self.mid_label_10,
                            self.mid_label_11, self.mid_label_12, self.mid_label_13]
            labels_out = [self.mid_label_15, self.mid_label_16, self.mid_label_17, self.mid_label_18,
                          self.mid_label_19, self.mid_label_20, self.mid_label_21, self.mid_label_22, self.mid_label_23,
                          self.mid_label_24, self.mid_label_25, self.mid_label_26]
            file = open('./pose_module/plans/' + self.user_train_plan + '.txt', 'r')
            i = 0
            s = file.read()
            counts_poses = s.count("\n")
```

```
                labels_poses = labels_poses[:counts_poses]
                labels_out = labels_out[:counts_poses]
                while i < counts_poses:
                    labels_poses[i].setText(s.split('\n')[i].split(' ')[0])
                    labels_out[i].setText(self.train_out.split(' ')[i])
                    i = i + 1
                file.close()


class MineWindow2_doctor(QMainWindow, Ui_MainWindow2_2):
    signal_exit = pyqtSignal()
    signal_plan = pyqtSignal()
    signal_trainer_infor = pyqtSignal()

    def __init__(self, parent=None):
        # super(MineWindow, self).__init__(None, Qt.FramelessWindowHint)  # 这句和普通的不一样 因为可以实现无边框
        super(MineWindow2_doctor, self).__init__(parent)
        # 初始化 ui
        self.setupUi(self)
        # 设定有关训练模型参数
        self.interval_frames_num = 3
        self.time_step = 30
        self.time_width = 5
        self.use_show_2d = True
        self.use_show_3d = True

        # 设定数据库参数
        self.con = connect_mysql
        self.cur = self.con.cursor()

        # 设定其它参数
        self.user_name = text_read('user_name')
        self.left_label_2.setText(self.user_name)
        self.print_position()  # 进入页面首先打印动作库
        self.action_label = ''
        self.action_name = ''

        # 设定按钮传递函数
        self.left_close.clicked.connect(self.close)
        self.left_mini.clicked.connect(self.showMinimized)
        self.left_visit.clicked.connect(self.showMaximized)

        self.pushButton_11.clicked.connect(self.show_new_action)  # 创建新动作
```

```
        self.pushButton_l3.clicked.connect(self.creat_plan)
        self.pushButton_l4.clicked.connect(self.signal_trainer_emit)
        self.right_label_2.clicked.connect(self.start_recording)  # 确定新方案
        self.right_label_3.clicked.connect(self.delete_pose)  # 删除
        self.right_label_4.clicked.connect(self.read_pose)  # 展示详情
        self.pushButton_l9.clicked.connect(self.return_to_login)  # 返回登录界面

        self.right_label_17.clicked.connect(self.commit_action)

    def clear_mid(self):
        buttons = [self.mid_label_1, self.mid_label_2, self.mid_label_3, self.mid_label_4,
self.mid_label_5, self.mid_label_6,
                   self.mid_label_7, self.mid_label_8, self.mid_label_9, self.mid_label_10,
self.mid_label_11,
                   self.mid_label_12, self.mid_label_13]
        for button in buttons:
            button.clicked.connect(self.clear_button)
            button.disconnect()
            button.setText('')

    def clear_right(self):
        buttons = [self.right_label_1, self.right_label_2, self.right_label_3, self.right_label_4,
self.right_label_5,
                   self.right_label_6,self.right_label_7, self.right_label_8,
self.right_label_9, self.right_label_10,
                   self.right_label_11,self.right_label_12, self.right_label_13,
self.right_label_14,
                   self.right_label_15, self.right_label_16, self.right_label_17, self.name]
        linetext = [self.lineedit_1, self.lineedit_2, self.lineedit_3, self.lineedit_4,
self.lineedit_5,
                    self.lineedit_6, self.lineedit_7, self.lineedit_8, self.lineedit_l1,
self.lineedit_l2]
        for button in buttons:
            button.setText('')
        for lineedit in linetext:
            lineedit.setStyleSheet('''QLineEdit{
                                      font-size:20px;font-weight:700;
                                      font-family: "Helvetica Neue";
                                      border-radius:12px;
                                      height:25px;
                                      color:white;
                                      border-radius:15px;
                                      background:#444444;
                                          }''')
```

```
                lineedit.setPlaceholderText("")
                lineedit.setAlignment(Qt.AlignCenter)
    def clear_button(self):
        pass

    def creat_plan(self):
        self.signal_plan.emit()

    def signal_trainer_emit(self):
        self.signal_trainer_infor.emit()

    def print_position(self):  # 刷新页面，动作库打印到对应 label
        self.clear_mid()
        positions = os.listdir('./pose_module/pose_library')
        buttons = [self.mid_label_1, self.mid_label_2, self.mid_label_3, self.mid_label_4,
self.mid_label_5,
                   self.mid_label_6, self.mid_label_7, self.mid_label_8, self.mid_label_9,
self.mid_label_10,
                   self.mid_label_11, self.mid_label_12, self.mid_label_13]
        self.lineedit_11.setStyleSheet('''QLineEdit{
                                                font-size:23px;font-weight:700;
                                                font-family: "Helvetica Neue";
                                                border-radius:12px;
                                                height:25px;
                                                color:white;
                                                border-radius:15px;
                                                background:#444444;
                                                }''')
        i = 0
        len_positions = len(positions)
        if len_positions == 0:
            self.clear_right()
            pass
        else:
            buttons = buttons[:len_positions]
            while i < len_positions:
                buttons[i].setText(positions[i].split('.')[0])
                i = i + 1
            for button in buttons:
                button.clicked.connect(self.read_positions)  # 读取动作详情

    def show_new_action(self):  # 在右边显示新动作创建情况
        linetext = [self.lineedit_1, self.lineedit_2, self.lineedit_3, self.lineedit_4,
self.lineedit_5,
```

```
                self.lineedit_6,    self.lineedit_7,    self.lineedit_8,    self.lineedit_l1,
self.lineedit_l2]
        for lineedit in linetext:
            lineedit.setText('')
            lineedit.setStyleSheet('"QLineEdit{
                                    border:2px solid gray;
                                    font-size:18px;font-weight:700;
                                    font-family: "Helvetica Neue";
                                    border-radius:12px;
                                    height:25px;
                                    color:black;
                                    border:2px solid white;
                                    border-radius:15px;
                                    background:white;
                                     }"')
        self.right_label_1.setText('录制新动作')
        self.lineedit_l1.clear()
        self.name.setText('动作名称')
        self.lineedit_l1.setPlaceholderText("请输入动作名称：")
        self.lineedit_l1.setAlignment(Qt.AlignCenter)  # 输入的字放到中间
        self.right_label_2.setText('开始录制')
        self.right_label_4.setText('播放')
        self.right_label_5.setText("设置动作")
        self.right_label_6.setText("帧数")
        self.right_label_7.setText("部位")
        self.right_label_8.setText("允许偏差")
        self.right_label_9.setText("左上臂")
        self.right_label_10.setText("右上臂")
        self.right_label_11.setText("左下臂")
        self.right_label_12.setText("右下臂")
        self.right_label_13.setText("左大腿")
        self.right_label_14.setText("右大腿")
        self.right_label_15.setText("左小腿")
        self.right_label_16.setText("右小腿")
        self.right_label_17.setText("确定")
        self.action_name = ''
        self.right_label_3.setText('')

    def start_recording(self):
        if self.action_name == '':  # 是空的，说明点击了创建新动作，不是空的说明点击
了按键已有动作
            self.action_name = self.lineedit_l1.text()
        if self.action_name == '':
            QMessageBox.about(self, 'warning!', '请输入动作名称')
```

```
            else:
                write_positions_file(self.action_name, self.interval_frames_num,
                                     self.time_step,
                                     self.time_width, use_show_2d=self.use_show_2d,
                                     use_show_3d=self.use_show_3d)
                self.print_position()

    def read_positions(self):  #
        self.action_name = self.sender().text()  # 记录哪个按键被点击
        self.right_label_1.setText('当前选中:' + self.action_name)
        self.lineedit_l1.clear()
        self.lineedit_l1.setPlaceholderText("")
        self.lineedit_l1.setStyleSheet('''QLineEdit{
                                          font-size:18px;font-weight:700;
                                          font-family: "Helvetica Neue";
                                          border-radius:12px;
                                          height:25px;
                                          color:white;
                                          border-radius:15px;
                                          background:#444444;
                                                  }''')
        self.name.setText('')
        self.right_label_2.setText('重新录制')
        self.right_label_3.setText('删除动作')
        self.right_label_4.setText('查看详情')
        self.right_label_5.setText("重新设置动作")
        self.right_label_6.setText("帧数")
        self.right_label_7.setText("部位")
        self.right_label_8.setText("允许偏差")
        self.right_label_9.setText("左上臂")
        self.right_label_10.setText("右上臂")
        self.right_label_11.setText("左下臂")
        self.right_label_12.setText("右下臂")
        self.right_label_13.setText("左大腿")
        self.right_label_14.setText("右大腿")
        self.right_label_15.setText("左小腿")
        self.right_label_16.setText("右小腿")
        self.right_label_17.setText("确定")
        linetext = [self.lineedit_1, self.lineedit_2, self.lineedit_3, self.lineedit_4,
self.lineedit_5,
                    self.lineedit_6, self.lineedit_7, self.lineedit_8, self.lineedit_l2]
        for lineedit in linetext:
            lineedit.clear()
            lineedit.setText('')
```

```
                lineedit.setStyleSheet('"QLineEdit{
                                                    border:2px solid gray;
                                                    font-size:23px;font-weight:700;
                                                    font-family: "Helvetica Neue";
                                                    border-radius:12px;
                                                    height:25px;
                                                    color:black;
                                                    border:2px solid white;
                                                    border-radius:15px;
                                                    background:white;
                                                     }"')

    def delete_pose(self):
        if self.action_name == '':
            QMessageBox.about(self, 'warning', '请先点击动作库中您想删除的动作')
        else:
            path = "./pose_module/pose_library/" + self.action_name + '.txt'
            os.remove(path)
            self.print_position()

    def read_pose(self):
        if self.action_name == '':
            QMessageBox.about(self, 'warning', '请先选择动作库中您想查看的动作或录制新动作')
        else:
            QApplication.processEvents()
            read_positions_file(self.action_name)

    def return_to_login(self):
        self.signal_exit.emit()  # 登录界面
        self.user_name = ''
        self.left_label_2.setText('')
        self.close()  # 加清屏

    def commit_action(self):
        frames_num = self.lineedit_l2.text()
        errors = [self.lineedit_1.text(), self.lineedit_2.text(), self.lineedit_3.text(),
self.lineedit_4.text(),
                  self.lineedit_5.text(), self.lineedit_6.text(), self.lineedit_7.text(),
self.lineedit_8.text()]
        body_parts = []
        angle_errors = []
        body_all = ['ab', 'cd', 'e', 'f', 'gh', 'ij', 'k', 'l']
        flag = True
```

```
        if self.action_name == '':
            QMessageBox.about(self, "warning", "请先创建新动作或在动作库中选择动作")
        else:
            if frames_num == '':
                QMessageBox.about(self, "warning", "请先播放动作选择帧数")
            else:
                error_count = 0
                i = 0
                while i < 8:
                    if errors[i] != '':
                        if int(errors[i]) > 90:
                            flag = False
                            QMessageBox.about(self, 'warning', '偏差太大')
                            break
                        error_count += 1
                        body_parts.append(body_all[i])
                        angle_errors.append(int(errors[i]))
                        if i == 0 or i == 1 or i == 4 or i == 5:
                            angle_errors.append(int(errors[i]))
                    i = i + 1
                if error_count == 8:
                    QMessageBox.about(self, "warning", "请填写识别误差")
                else:
                    if flag:
                        sql = 'select * from actions where action_name="%s"' % self.action_name
                        res = self.cur.execute(sql)
                        if res:
                            sql = 'DELETE FROM actions where action_name="%s"' % self.action_name
                            self.cur.execute(sql)
                        sql = 'insert into actions(action_name,frames_num,body_parts,angle_errors) ' \
                              'values("%s","%s","%s","%s")' % (
                                  self.action_name, frames_num,
                                  str(body_parts)[1:-1].replace(', ', '').replace('\'', ''),
                                  str(angle_errors)[1:-1].replace(',', ''))
                        self.cur.execute(sql)
                        self.con.commit()


class MineWindow_plan(QMainWindow, Ui_MainWindow_plan):
    signal_exit = pyqtSignal()
```

```
    def __init__(self, parent=None):
        # super(MineWindow, self).__init__(None, Qt.FramelessWindowHint)  # 这句和普通的不一样 因为可以实现无边框
        super(MineWindow_plan, self).__init__(parent)
        # 初始化 ui
        self.setupUi(self)

        # 设定数据库参数
        self.con = connect_mysql
        self.cur = self.con.cursor()

        # 设定其它参数
        self.print_plan_library()
        action_buttons = self.print_pose_library()  # 进入页面首先打印动作库
        for button in action_buttons:
            button.clicked.connect(self.add_plan)
        self.plan_name = ''
        self.plan_content = []

        # 设定按钮传递函数
        self.left_close.clicked.connect(self.close_plan)
        self.left_mini.clicked.connect(self.showMinimized)
        self.left_visit.clicked.connect(self.showMaximized)
        self.pushButton_l3.clicked.connect(self.close_plan)
        self.right_label.clicked.connect(self.commit_plan)
        self.mid_label_15.clicked.connect(self.delete_plan)

    def close_plan(self):
        self.close()

    def clear_mid(self):
        self.mid_label_content.setText('')
        buttons = [self.mid_label_1, self.mid_label_2, self.mid_label_3, self.mid_label_4, self.mid_label_5,
                   self.mid_label_6,
                   self.mid_label_7, self.mid_label_8, self.mid_label_9, self.mid_label_10, self.mid_label_11,
                   self.mid_label_12, self.mid_label_13]
        for button in buttons:
            button.clicked.connect(self.clear_button)
            button.disconnect()
            button.setText('')
```

```
    def print_plan_library(self):
        self.clear_mid()
        plans = os.listdir('./pose_module/plans')
        buttons = [self.mid_label_1, self.mid_label_2, self.mid_label_3, self.mid_label_4,
self.mid_label_5,
                   self.mid_label_6, self.mid_label_7, self.mid_label_8, self.mid_label_9,
                   self.mid_label_10, self.mid_label_11, self.mid_label_12,
self.mid_label_13]
        i = 0
        len_plans = len(plans)
        if len_plans == 0:
            pass
        else:
            buttons = buttons[:len_plans]
            while i < len_plans:
                buttons[i].setText(plans[i].split('.')[0])
                i = i + 1
            for button in buttons:
                button.clicked.connect(self.read_plan)

    def delete_plan(self):
        if self.plan_name == '':
            QMessageBox.about(self, 'warning', '请先点击方案')
        else:
            path = "./pose_module/plans/" + self.plan_name + ".txt"
            os.remove(path)
            self.print_plan_library()

    def read_plan(self):
        path = "./pose_module/plans/" + self.sender().text() + ".txt"
        file = open(path, 'r')
        content = file.read()
        self.mid_label_content.setText(content)
        self.plan_name = self.sender().text()
        file.close()

    def print_pose_library(self):  # 刷新页面，动作库打印到对应 label
        poses = os.listdir('./pose_module/pose_library')
        buttons = [self.right_label_12, self.right_label_13, self.right_label_14,
self.right_label_15,
                   self.right_label_16, self.right_label_17, self.right_label_18,
self.right_label_19,
                   self.right_label_20, self.right_label_21, self.right_label_22]
        i = 0
```

```
            len_positions = len(poses)
            buttons = buttons[:len_positions]
            while i < len_positions:
                buttons[i].setText(poses[i].split('.')[0])
                i = i + 1
            return buttons

        def add_plan(self):
            if self.sender().text() not in self.plan_content:
                self.plan_content.append(self.sender().text())
                buttons = [self.right_label_1, self.right_label_2, self.right_label_3,
self.right_label_4,
                           self.right_label_5,
                           self.right_label_6, self.right_label_7, self.right_label_8,
self.right_label_9,
                           self.right_label_10,
                           self.right_label_11]
                i = 0
                buttons = buttons[:len(self.plan_content)]
                while i < len(self.plan_content):
                    buttons[i].setText(self.plan_content[i])
                    buttons[i].clicked.connect(self.clear_button)
                    buttons[i].disconnect()
                    buttons[i].clicked.connect(self.del_plan)
                    i = i + 1
                self.clear_count()
            else:
                pass

        def del_plan(self):
            self.plan_content.remove(self.sender().text())
            buttons = [self.right_label_1, self.right_label_2, self.right_label_3, self.right_label_4,
self.right_label_5,
                       self.right_label_6, self.right_label_7, self.right_label_8,
self.right_label_9, self.right_label_10,
                       self.right_label_11]
            for button in buttons:
                button.setText('')
                button.clicked.connect(self.clear_button)
                button.disconnect()
            i = 0
            buttons = buttons[:len(self.plan_content)]
            while i < len(self.plan_content):
                buttons[i].setText(self.plan_content[i])
```

```
                buttons[i].clicked.connect(self.del_plan)
                i = i + 1
            self.clear_count()

        def clear_button(self):
            pass

        def commit_plan(self):   # 将方案保存到文件夹中
            self.plan_name = self.lineedit_l1.text()
            if self.plan_name == '' or (len(self.plan_content) > 9 or len(self.plan_content) == 0):
                if self.plan_name == '':
                    QMessageBox.about(self, 'warning', '请输入方案名')
                if len(self.plan_content) > 9 or len(self.plan_content) == 0:
                    QMessageBox.about(self, 'warning', '一个方案内的动作数应在 1-9 之间')
            else:
                line_edits = [self.lineedit1, self.lineedit2, self.lineedit3, self.lineedit4,
self.lineedit5,
                              self.lineedit6, self.lineedit7, self.lineedit8, self.lineedit9,
self.lineedit10,
                              self.lineedit11]
                i = 0
                line_edits = line_edits[:len(self.plan_content)]
                plan_str = ''
                flag = 0
                while i < len(self.plan_content):
                    count = line_edits[i].text()
                    if str.isdigit(count):
                        flag = 1
                        sql = 'select * from actions where action_name="%s"' %
self.plan_content[i]
                        res = self.cur.execute(sql)
                        results = self.cur.fetchall()
                        plan_str = plan_str + self.plan_content[i] + ' ' + count + ' ' + results[0][2]
+ ' ' + results[0][
                                   3] + ' ' + results[0][4] + '\n'
                        i = i + 1
                    else:
                        flag = 0
                        QMessageBox.about(self, 'warning', '请以正确形式输入个数')
                        break
                if flag == 1:
                    path = "./pose_module/plans/" + self.plan_name + ".txt"
                    file = open(path, 'w')
                    file.write(plan_str)
```

```
                file.close()
            self.print_plan_library()
            self.plan_name = ''

    def clear_count(self):
        line_edits = [self.lineedit1, self.lineedit2, self.lineedit3, self.lineedit4, self.lineedit5,
                      self.lineedit6, self.lineedit7, self.lineedit8, self.lineedit9,
self.lineedit10,
                      self.lineedit11]
        for line_edit in line_edits:
            line_edit.clear()


class MineWindow_information(QMainWindow, Ui_MainWindow_information):
    signal_exit = pyqtSignal()

    def __init__(self, parent=None):
        super(MineWindow_information, self).__init__(parent)
        # 初始化 ui
        self.setupUi(self)

        # 设定数据库参数
        # self.con = connect_mysql
        # self.cur = self.con.cursor()
        self.show_trainer()

        # 设定按钮传递函数
        self.left_close.clicked.connect(self.close)
        self.left_mini.clicked.connect(self.showMinimized)
        self.left_visit.clicked.connect(self.showMaximized)

    def show_trainer(self):
        self.con = connect_mysql
        self.cur = self.con.cursor()
        sql = 'select name from user where identity = "患者"'
        res = self.cur.execute(sql)
        results = self.cur.fetchall()
        counts_trainer = len(results)
        name_buttons = [self.pushButton_l2, self.pushButton_l3, self.pushButton_l4,
                        self.pushButton_l5, self.pushButton_l6, self.pushButton_l7,
self.pushButton_l8,
                        self.pushButton_l9, self.pushButton_l10, self.pushButton_l11,
self.pushButton_l12,
                        self.pushButton_l13, self.pushButton_l14]
```

```
            i = 0
            while i < counts_trainer:
                name_buttons[i].setText(results[i][0])
                name_buttons[i].clicked.connect(self.show_information)
                i = i + 1

        def clear_all(self):
            texts = [self.mid_label_11, self.mid_label_12, self.mid_label_13, self.mid_label_14,
                     self.mid_label_21,  self.mid_label_22,  self.mid_label_23,
self.mid_label_24,
                     self.mid_label_31,  self.mid_label_32,  self.mid_label_33,
self.mid_label_34,
                     self.mid_label_41,  self.mid_label_42,  self.mid_label_43,
self.mid_label_44,
                     self.mid_label_51,  self.mid_label_52,  self.mid_label_53,
self.mid_label_54,
                     self.mid_label_61,  self.mid_label_62,  self.mid_label_63,
self.mid_label_64,
                     self.mid_label_71,  self.mid_label_72,  self.mid_label_73,
self.mid_label_74,
                     self.mid_label_81,  self.mid_label_82,  self.mid_label_83,
self.mid_label_84,
                     self.mid_label_91,  self.mid_label_92,  self.mid_label_93,
self.mid_label_94,
                     self.right_label_1, self.right_label_2, self.right_label_3, self.right_label_4]
            for text in texts:
                text.setText('')

        def show_information(self):
            self.con = connect_mysql
            self.cur = self.con.cursor()
            self.clear_all()
            self.trainer_name = self.sender().text()
            if self.trainer_name == '':
                QMessageBox.about(self, 'warning', '请点击按钮')
            else:
                texts    =    [[self.mid_label_11,    self.mid_label_12,    self.mid_label_13,
self.mid_label_14],
                              [self.mid_label_21,    self.mid_label_22,    self.mid_label_23,
self.mid_label_24],
                              [self.mid_label_31,    self.mid_label_32,    self.mid_label_33,
self.mid_label_34],
                              [self.mid_label_41,    self.mid_label_42,    self.mid_label_43,
self.mid_label_44],
```

```
                        [self.mid_label_51,     self.mid_label_52,     self.mid_label_53,
self.mid_label_54],
                        [self.mid_label_61,     self.mid_label_62,     self.mid_label_63,
self.mid_label_64],
                        [self.mid_label_71,     self.mid_label_72,     self.mid_label_73,
self.mid_label_74],
                        [self.mid_label_81,     self.mid_label_82,     self.mid_label_83,
self.mid_label_84],
                        [self.mid_label_91,     self.mid_label_92,     self.mid_label_93,
self.mid_label_94]]
            sql = 'select * from %s ' % self.trainer_name
            self.cur.execute(sql)
            self.results = self.cur.fetchall()
            counts = len(self.results)
            i = 0
            while i < counts:
                texts[i][0].setText(str(self.results[i][0]))
                texts[i][1].setText(str(
                    self.results[i][1] + '/' + self.results[i][2] + '/' + self.results[i][3] +
self.results[i][
                        4]))
                texts[i][2].setText(str(self.results[i][5]))
                texts[i][3].setText('查看 ' + str(i))
                texts[i][3].clicked.connect(self.show_limitation)
                i = i + 1

    def show_limitation(self):
        self.right_label_1.setText('极限角度')
        self.right_label_3.setText('完成情况（有效/实作/应做）')
        i = int(self.sender().text().split(' ')[1])
        angles_name = ['左上臂前举', '右上臂前举', '左下臂弯曲', '右下臂弯曲',
                       '左大腿前踢',  '右大腿前踢', '左小腿弯曲', '右小腿弯曲']
        j = 0
        str_lim = ''
        while j < self.results[i][6].count('.'):
            str_lim = str_lim + angles_name[j] + ":" + self.results[i][6].replace(' ', '').split('.')[j]
+ '\n'
            j = j + 1
        self.right_label_2.setText(str_lim)

        j = 0
        str_out = ''
        file = open('./pose_module/plans/' + self.results[i][5] + '.txt', 'r')
        s = file.read()
```

```
            file.close()
            while j < self.results[i][7].count(' ') + 1:
                str_out = str_out + s.split('\n')[j].split(' ')[0] + ":" + '\n' + self.results[i][7].split(' ')[j] + '\n'
                j = j + 1
            self.right_label_4.setText(str_out)

# 用一个控制器来控制页面的跳转
class Controller:
    def __init__(self):
        pass

    # 跳转到 login 窗口
    def show_login(self):
        self.login = MyMainWindow()
        self.login.show()
        self.login.signal1.connect(self.show_register)
        self.login.signal2_1.connect(self.show_trainer)
        self.login.signal2_2.connect(self.show_doctor)

    def show_register(self):
        self.register = MineWindow_register()
        self.register.show()

    def show_trainer(self):
        self.trainer = MineWindow2_trainer()
        self.trainer.signal_exit.connect(self.login.show)
        self.trainer.show()

    def show_doctor(self):
        self.doctor = MineWindow2_doctor()
        self.doctor.signal_exit.connect(self.login.show)
        self.doctor.signal_plan.connect(self.show_plan)
        self.doctor.signal_trainer_infor.connect(self.show_information)
        self.doctor.show()

    def show_plan(self):
        self.plan = MineWindow_plan()
        self.plan.show()

    def show_information(self):
        self.information = MineWindow_information()
        self.information.show()
```

```
def run_ui():
    app = QApplication(sys.argv)
    controller = Controller()   # 控制器实例
    controller.show_login()   # 默认展示的是 login 页面
    sys.exit(app.exec_())
```