南開大學
Nankai University

南　　开　　大　　学

计　算　机　学　院

软件工程模块详细说明

车载多模态智能交互系统
模块详细说明

年级：2022 级

专业：计算机科学与技术

指导教师：李起成

林逸典　唐显达　姜宇　郭笑语
刘芳宜　王禹衡　何畅　董珺

2025 年 6 月 17 日

目录

一、 引言

我们团队精心研发了一款车载多模态智能交互系统软件。该系统巧妙地融合了语音模态和视觉模态，突破传统车载系统单一交互模式的局限。该系统的语音模态由语音识别子系统和语音合成子系统构成，视觉模态则涵盖头部姿态子系统、手势动作子系统以及视觉方向子系统。在本模块详细说明文档中，我们将深入剖析上述各子系统的实现逻辑，展现其背后的技术支撑与实现逻辑。

二、 语音模态

(一) 语音识别子系统

1. 功能简介

语音识别子系统通过实时接收并识别驾驶员语音命令，实现车载功能的自然语言控制。系统集成了语音命令识别、声纹识别和唤醒词检测等功能，支持多种交互场景，如播放音乐、车辆导航等。

系统采用实时流式语音处理机制，通过 WebRTC 的 VAD 技术动态判断语音段落，有效过滤静音片段和环境噪声。同时，引入唤醒词检测逻辑，确保系统仅在用户明确表达意图后，才进入识别与控制流程，有效避免误唤醒情况的发生。

本系统基于 FunASR 框架调用达摩院的预训练模型 iic/SenseVoiceSmall 完成中文语音识别，并结合 damo/speech_campplus_sv_zh-cn_16k-common 声纹识别模型，支持多用户声纹注册与识别。系统最终输出结构化的识别结果，包含文本内容、关键词类别、唤醒状态与用户信息，供上层系统调用。例如，应用功能控制系统会依据语音识别子系统的语音命令和识别结果结合个性化系统的配置文件调用相应的应用功能，权限控制系统则会根据声纹识别功能的结果进行权限管理。

综上所述，本系统具备以下功能:

- 实时语音采集与处理: 使用 PyAudio 持续录音，配合 WebRTC VAD 实现高效语音活动检测。
- 唤醒词识别与管理: 支持多个自定义唤醒词，内置拼音转换与匹配机制。
- 语音内容识别: 基于 FunASR 框架调用中文语音识别模型，适应车载指令场景。
- 关键词分类识别: 自动提取语音中的关键词，并映射至具体控制类别（如音乐、导航）。
- 说话人身份识别: 调用 ModelScope 声纹识别模型，支持注册用户与临时用户识别。
- 声纹注册与管理: 可添加、删除用户声纹档案，临时用户可一键转为正式用户。
- 结构化结果输出: 统一封装语音识别状态，便于系统其他模块调用与解析。

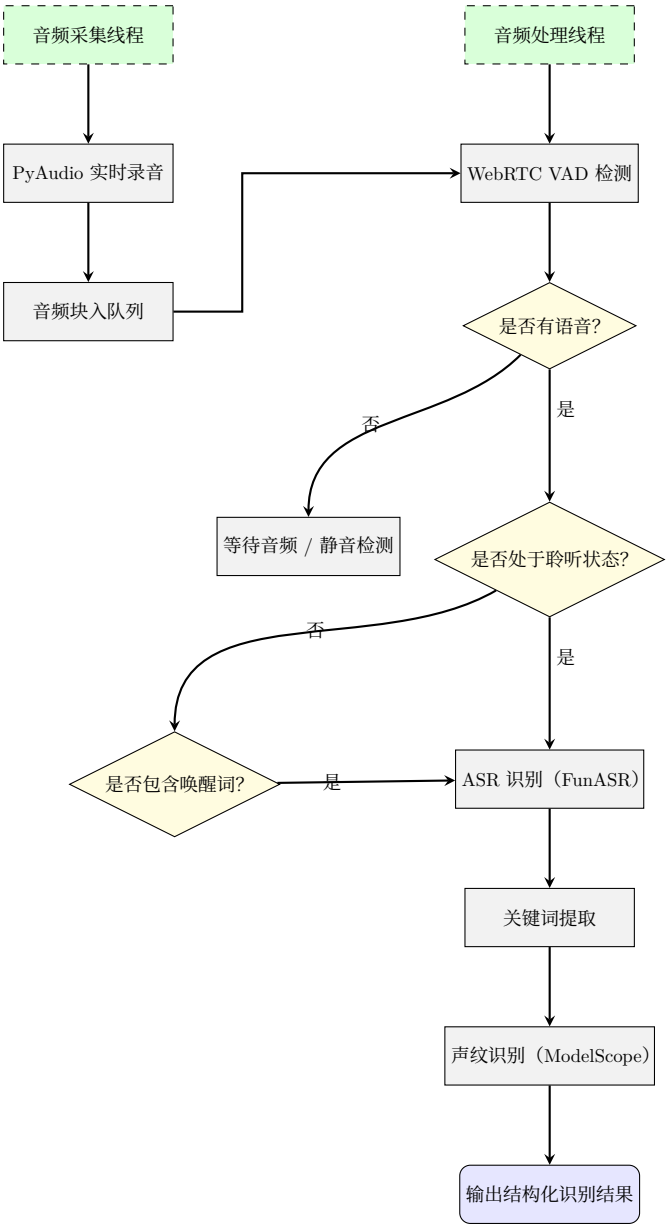

图 1: 语音识别子系统流程图

2. 模型来源

本系统采用的语音识别与声纹识别功能均基于达摩院发布的预训练模型，模型托管于 ModelScope 平台，模型来源主要包括以下两部分：

- 语音识别模型（ASR）：该模型适用于中英文语音命令识别，具备良好的响应速度与识别准确性。
- 声纹识别模型（SV）：该模型用于支持 16kHz 语音条件下的用户身份验证与识别。

阿里巴巴达摩院作为全球顶尖的科研机构，在人工智能领域具备深厚的技术积累与强大的研发实力。其汇聚了众多行业内顶尖的科研人才，拥有先进的科研设备和丰富的数据资源，为模型的训练与优化提供了坚实保障。这种强大的科研背景和技术实力，确保了达摩院所研发的模型在性能和效果上具有显著优势。

3. 模型与训练方法

语音识别子系统采用“调包使用 + 本地缓存”策略，依托阿里巴巴达摩院托管在开源平台 ModelScope 的 FunASR 模型，实现中文语音识别与声纹识别功能，构建端到端的语音理解流程。

主要依赖的 Python 环境包括：

- **FunASR**：达摩院开源的语音识别框架，提供统一的模型加载与推理接口，支持多种前端特征处理方式、流式识别机制和中文语音命令场景优化。本项目中通过 `AutoModel` 与 `AutoTokenizer` 接口加载 `iic/SenseVoiceSmall` 模型，完成中文语音转文本（ASR）任务。
- **ModelScope**：阿里云提供的多模态 AI 模型库，支持模型统一注册、部署与调用。本项目借助其 `pipeline` 接口加载声纹识别模型 `damo/speech_campplus_sv_zh-cn_16k-common`，实现用户身份的语音验证。
- **webrtcvad**：Google 提供的轻量级语音活动检测（VAD）模块，用于剔除静音和噪声段，提升语音处理效率。
- **pyaudio**：用于实时音频采集，支持 16kHz 单声道 PCM 数据流输入，构成系统前端入口。

模型首次调用时将自动从 ModelScope 下载并缓存至本地，后续调用优先使用缓存目录，避免重复下载与网络依赖，提升模块稳定性与启动效率。模型本身为阿里达摩院预训练所得，无需用户自行训练，适配性强、部署简单，便于集成至车载系统等资源受限环境。

4. 模块技术细节

模块	实现技术	说明
音频输入	‘pyaudio‘ 实时采集	单声道、16kHz、16-bit PCM，支持缓存与异常恢复
VAD 检测	‘webrtcvad‘	检测语音起止，过滤静音片段
唤醒词识别	拼音匹配	用户可动态添加/移除唤醒词，支持多唤醒词拼音列表
ASR 识别	‘AutoModel‘ from FunASR	加载 SenseVoiceSmall 模型，支持语种选择与缓存策略
关键词识别	自定义词典匹配	配置中可定义“打开空调”等关键词及其类别
声纹识别	ModelScope ‘pipeline‘	支持多说话人比对、注册与临时存储逻辑
模态状态更新	自定义 ‘SpeechState‘	返回文本、关键词、置信度、说话人信息等结构化结果

表 1: 模块实现技术与说明

(二) 语音合成子系统

1. 功能简介

语音合成子系统旨在将车载系统产生的文本信息实时转换为自然流畅的语音，提升车载交互的人性化与反馈效率。例如，在系统提示或与用户进行语音交互时，系统可将提示内容、交互需求等文本通过语音播报输出，确保驾驶员在不分散注意力的前提下接收有效信息。

系统默认支持 GPU 加速，合成语音输出为 16kHz、单声道 PCM 格式，可直接送入音频播放线程，适配车载扬声器系统。

该系统具备以下功能：

- 文本预处理与格式适配: 支持中文文本的编码转换与清洗，保障合成效果。
- 语音合成推理: 基于 ModelScope 推理框架加载声学模型（Sambert）与声码器（HiFi-GAN）完成语音生成。
- 音频缓存与回放接口：合成后音频可存为 wav 文件或直接供扬声器播放，支持异步播放与清理。
- 多场景适配能力: 适用于系统播报、对话反馈、驾驶辅助等多种语音交互场景。
- 高效调用机制: 首次加载后支持模型常驻内存，提升响应速度。

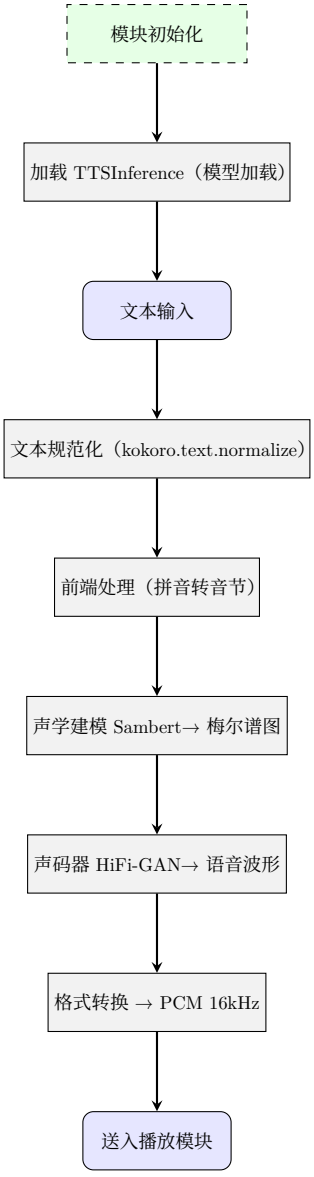

图 2: 语音合成子系统流程图

2. 模型来源

该系统采用开源库 kokoro 及其模型 Kokoro-82M-v1.1-zh 进行语音合成, 由阿里巴巴达摩院在 ModelScope 平台发布。Kokoro 是一系列存储空间较小但功能强大的 TTS 模型，可实现高质

量的中文语音合成，具备自然音色、低延迟以及高质量等特性，适配车载环境需求。

3. 模型与训练方法

语音合成模块采用“Kokoro 包调用 + 端到端推理”的方式构建，基于达摩院语音研究团队开发的 kokoro 包，封装语音合成流程的各个环节。该工具包集成了 ModelScope 平台的预训练模型 `damo/speech_sambert-hifigan_tts_zhist`，支持输入文本到最终语音波形的完整生成。

kokoro 提供了统一的 `TTSInference` 接口，自动完成模型加载、文本预处理（如分词、拼音转换）与后处理过程，简化了开发流程。接口支持 GPU 加速推理，生成 16kHz 单声道 PCM 格式音频，可直接用于播放模块。

4. 模块技术细节

模块	实现技术	说明
结构化输出	函数分装	对原始模型的接口按照项目内部约定进行再封装
多语种支持	pipline	支持语音输出中文、英语等多种语言

表 2: 语音合成模块实现技术与说明

三、 视觉模态

(一) 摄像头管理器

在车载多模态智能交互系统里，我们采用全局唯一的摄像头管理器来统一管理摄像头资源。该管理器借助 cv2.VideoCapture 获取摄像头资源，并支持头部姿态子系统、手势动作子系统、视线方向子系统以及 UI 界面的摄像头区域同步读取摄像头数据，以此规避多种视觉模态直接竞争摄像头资源所引发的性能降低乃至程序崩溃问题。

(二) 头部姿态子系统

1. 功能简介

头部姿态子系统用于检测用户在交互过程中的头部运动模式，精确识别“静止”,“点头”和“摇头”等行为状态，并将其作为视觉输入信号传递给上层系统，从而提升人机交互的自然程度、容错能力与多样性。

该功能在需要用户反馈“是”或“否”的场景中尤为适用，例如对语音识别结果进行再确认，或是其他需要用户确认的交互场景，用户可通过“点头”表示确认、“摇头”表示拒绝，从而给出准确答复。

系统通过普通摄像头采集实时视频流，依托 MediaPipe 模型检测人脸关键点，选取与姿态估计相关的特定索引点（如鼻尖、眼角、嘴角等）用于构建三维头部模型。利用 OpenCV 的 `solvePnP` 算法，根据这些二维图像点和对应的三维人脸模型点，计算头部的三维旋转矩阵，并进一步转化为欧拉角（pitch、yaw、roll），表示头部在三个轴向上的姿态变化。

为了避免单帧判断带来的抖动与误识别，系统引入角度滑动窗口机制，将连续帧角度序列缓存，并使用训练好的 GRU（门控循环单元）神经网络进行时序建模。模型能有效捕捉角度变化趋势，对“静止”,“点头”和“摇头”三类动作进行准确分类。

为增强识别鲁棒性，模块还引入投票机制与状态缓冲池，对近期输出结果进行统计整合，过滤偶发误差，确保最终输出的结构化状态具有较高的置信度与实时性。

本系统支持以下关键功能:

- 面部关键点提取: 调用 MediaPipe 检测人脸三维特征点，覆盖鼻尖、眼角、嘴角、耳部等区域，适应不同角度与光照环境。
- 姿态角估计: 基于三维人脸建模与 `solvePnP` 解算，实现高精度 pitch/yaw/roll 姿态获取。
- 历史轨迹缓存: 采用双端队列（deque）缓存角度序列，支持滑动窗口与动态更新。
- 时序行为识别: 使用轻量级 GRU 网络建模角度变化趋势，稳定识别点头、摇头、静止三种动作。
- 置信度与投票机制: 通过窗口内结果统计判断动作置信度，提高输出稳定性。
- 结构化状态输出: 统一封装结果字段（动作类型、姿态角、置信度等），支持上层模块按需解析。

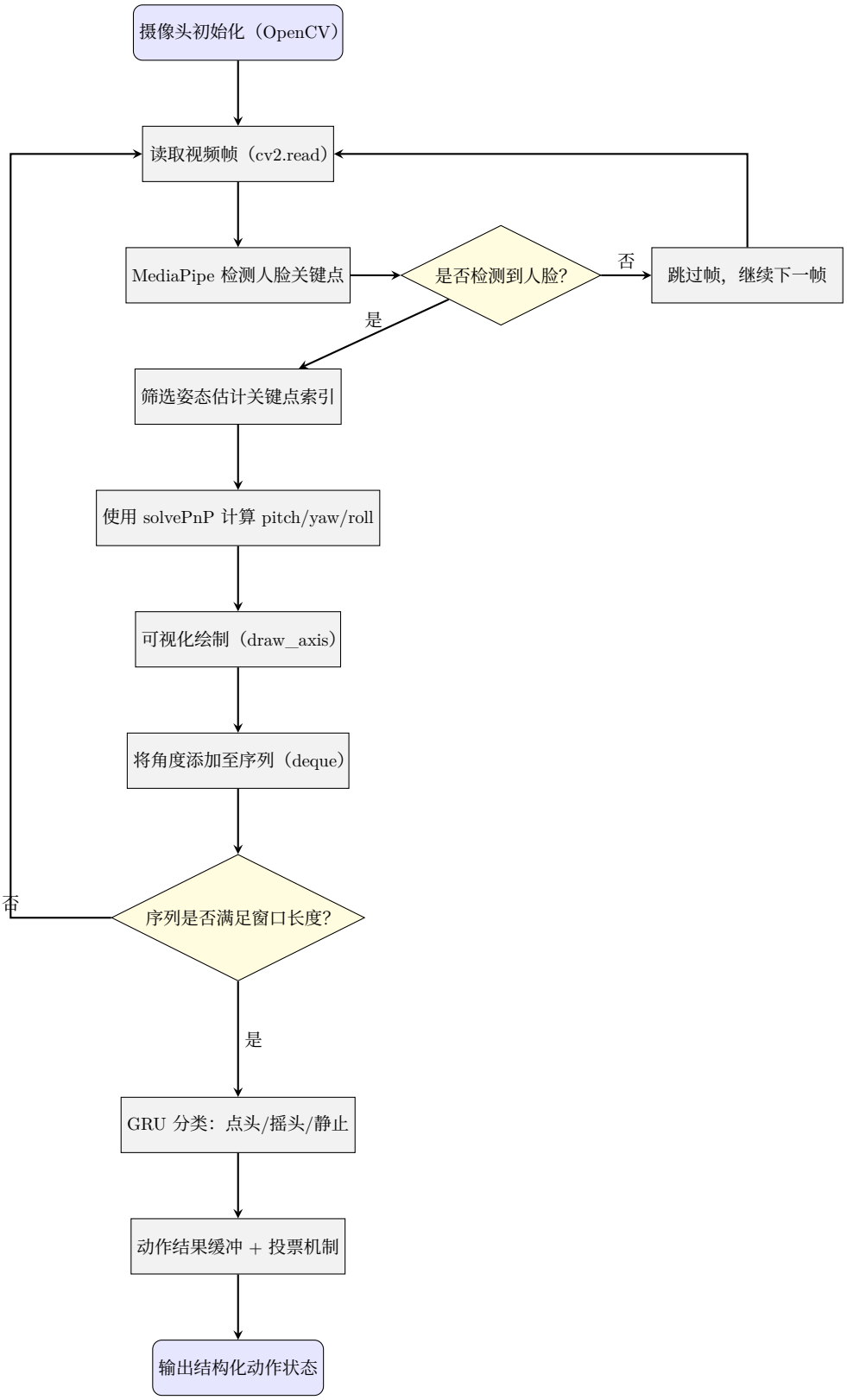

图 3: 头部姿态子系统流程图

2. 数据集来源

本系统采用自采集数据集进行训练与评估，以保障头部姿态识别模型在真实交互环境中的泛化能力与实用性。

具体而言，团队在模拟驾驶舱环境中采集了大量交互视频，包括“静止”,“点头”和“摇头”三类典型动作，共计 5000+ 序列，采用人工标注方式构建高质量训练样本 (**本数据集完全由开发团队采集与构建，具备良好的定制性与扩展性，适用于多种头部姿态识别场景的开发与验证**)。

此外，系统还将通过实时摄像头输入不断采集运行数据，在用户授权前提下进行离线标注与模型微调，支持未来在车载环境中的持续优化与自适应部署。

3. 模型与训练方法

头部姿态子系统采用“传统姿态解算 + GRU 时序分类器”的二阶段建模架构，结合计算几何与深度学习方法，精准识别用户在自然交互场景中的“静止”,“点头”和“摇头”动作趋势，确保识别结果既具实时性又具鲁棒性。

1. 姿态角提取模块（传统解算阶段）

- 系统首先使用 MediaPipe 提取人脸 468 个关键点，其中选取鼻尖、眼角、下巴、嘴角等与姿态相关的固定索引点作为特征输入。
- 结合 OpenCV 的 `solvePnP` 方法，使用一组标准 3D 模型坐标与图像中对应的 2D 点对拟合摄像头位姿，从而反推出欧拉角表示的三轴旋转角度：俯仰角（pitch）、偏航角（yaw）、翻滚角（roll）。
- 为提升稳定性，在每帧角度数据处理后加入低通滤波平滑算法，过滤因关键点抖动造成的角度突变。

此阶段以极低延迟实现帧级角度估计，为后续的时序建模提供稳定输入。

2. GRU 时序分类模块（深度分类阶段）

- 系统使用两层 GRU（Gated Recurrent Unit）网络对历史帧角度序列进行学习与建模，输入为长度为 15 的角度窗口序列（每帧包含 pitch/yaw/roll 三维特征）。
- GRU 相较于传统 RNN 或 LSTM，在结构上更简洁，训练速度快、性能鲁棒，特别适合在边缘设备或嵌入式平台部署。
- 其内部结构包含两大门控机制：更新门控制记忆保留比例，重置门决定新旧信息的融合方式，使模型能精准捕捉“静止”和“连续动作”的差异。
- 输出为三分类 softmax 概率向量，表示当前窗口序列对应“静止”,“点头”或“摇头”动作。同时结合滑动窗口，进一步增强识别鲁棒性，降低误报率。

3. 模型训练策略

- 训练数据来源于本地采集的车载视频样本，涵盖不同光照、距离与角度条件下的头部动作片段，由人工标注帧级动作标签。
- 使用交叉熵损失函数进行监督训练，优化器为 Adam，初始学习率设为 1×10^{-3}，训练总轮数为 50 轮，期间采用验证集监控防止过拟合。

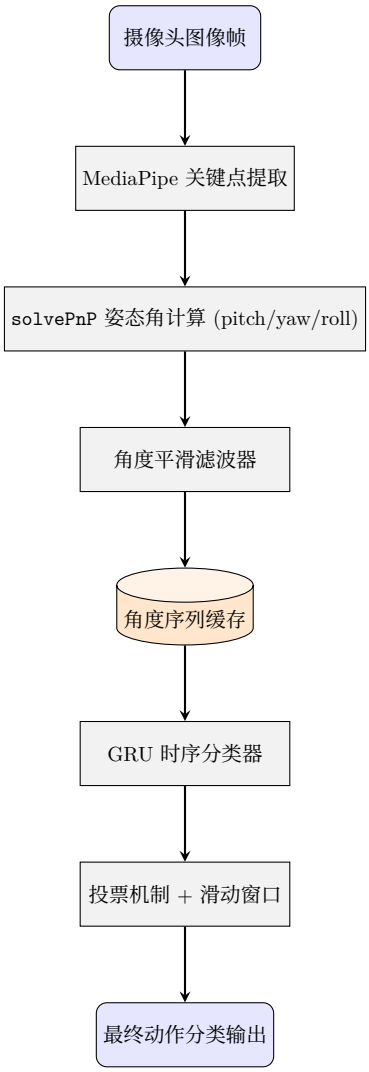

图 4: 头部姿态子系统建模流程图：从传统姿态解算到 GRU 分类

4. 模块技术细节

头部姿态子系统融合了传统图像处理与时序深度学习方法，采用模块化设计以确保高效性与可扩展性。系统整体分为三大核心阶段：关键点检测、姿态解算与行为识别，各环节细节如下表所示：

模块	技术细节
图像采集	使用 OpenCV 接入摄像头设备，设置固定分辨率与帧率，持续采集图像帧。
面部关键点检测	基于 MediaPipe FaceMesh 模型，提取 468 个高精度面部关键点，延迟低，稳定性好。
姿态角计算	选取姿态相关的特征点，通过 OpenCV 的 `solvePnP` 求解头部三维旋转向量，转化为 pitch/yaw/roll 欧拉角。
角度平滑处理	引入一阶滤波器方法，缓解关键点定位波动带来的角度抖动。
时序序列缓存	使用 Python `deque` 缓存历史 15 帧姿态角，组成三维时序输入序列。
GRU 动作识别	构建两层 GRU 网络，输入为 15×3 的角度矩阵，输出点头、摇头、静止和其它四分类标签，配合 softmax 输出置信度。
结果优化机制	引入窗口平滑算法，减少标签抖动，提高识别鲁棒性。
结构化输出	封装输出内容包括：动作类别、三轴姿态角值、动作置信度，供上层交互逻辑使用。

表 3: 头部姿态子系统的模块技术细节

(三) 手势动作子系统

1. 功能简介

手势动作子系统实现对用户静态手势与动态手势的精准识别。该系统包含两个独立模块，即静态手势识别模块和动态手势识别模块。这两个模块可并行处理手势动作信息，并将结果反馈至上层系统。上层系统可依据实际需求，选择使用单一模块信息，或综合两个模块信息进行处理。

具体而言，静态手势识别模块专注于对单帧图像中手势动作进行实时分类，可识别如“握拳”，“竖起大拇指”和“摇手”等简单手势动作；动态手势识别模块则侧重于捕捉连续帧之间的时序变化，能够识别如“滑动”，“推出”和“旋转”等复杂动态手势。

在我们的车载多模态智能交互系统中，手势动作子系统在典型场景交互中承担着“同意”与“拒绝”的功能。此外，它还具备在通用场景下启动应用功能的能力，并且支持个性化配置。例如，默认情况下，用户可通过“竖起大拇指”这一手势播放音乐；而在个性化配置后，可以通过“握拳”手势播放音乐或者“竖起大拇指”手势暂停音乐播放。

该子系统面向自然人机交互的应用需求，结合传统图像处理与深度学习模型，提供实时性高、交互性强的手势检测能力。系统通过摄像头连续采集图像流，利用 MediaPipe 等图像关键点检测工具或手工设计的预处理操作提取手部区域特征，再配合预训练的卷积神经网络（CNN）和时序建模结构进行手势分类与动作识别，最终生成结构化的交互指令供上层模块调用。

静态识别模块主要处理单帧图像中的空间特征。通过提取 21 个手部关键点的三维坐标，结合归一化与数据增强，构建尺度不变的特征输入，再输入至多层感知机（MLP）进行分类识别。动态识别模块则重点关注手势随时间的变化轨迹，采用 Conv3D 结构提取手势的时空动态信息，有效应对连续动作如“滑动”“推出”“旋转”等复杂动态动作的变化特征。

系统还引入质量检测机制，对关键点置信度不足或图像遮挡等低质量帧进行自动丢弃或路径切换，并通过集成多帧投票与置信度加权策略，提升输出结果的稳定性与准确性。识别结果包括标准格式结构化输出的动作类型、置信度及时间戳，便于上层系统使用，满足多种实际交互需求。

本系统支持常见的手势动作，具体包括:

- **静态手势**:“握拳”“竖起大拇指”“摇手”等简单手势动作。

编号	手势名称	具体描述
0	握拳	五指并拢呈拳头，仅拇指和食指圈成“0”形
1	竖起一根手指	竖起食指，其余四指收拢
2	竖起两根手指	竖起食指和中指，其余三指收拢
3	竖起三根手指	竖起拇指、食指和中指，无名指和小指收拢
4	竖起四根手指	竖起四指（除拇指外），拇指收拢
5	挥手	五指自然张开
6	竖起大拇指	拇指伸出，其余四指收拢
7	三指合拢	拇指、食指和中指伸出，其余两指收拢
8	手枪手	拇指与食指伸出，呈手枪形，其余三指收拢
9	弯钩	仅伸出食指弯曲勾起，其余收拢（或变体）
10	忽略	无明确手势或无效帧，跳过识别

- **动态手势**: 各类“滑动”,“推出”和“旋转”等共 27 个复杂动态动作手势。

编号	手势名称	描述说明
1	Swiping Left	单手向左快速滑动
2	Swiping Right	单手向右快速滑动
3	Swiping Down	单手向下快速滑动
4	Swiping Up	单手向上快速滑动
5	Pushing Hand Away	单手向前推出
6	Pulling Hand In	单手向内收回
7	Sliding Two Fingers Left	两指向左滑动
8	Sliding Two Fingers Right	两指向右滑动
9	Sliding Two Fingers Down	两指向下滑动
10	Sliding Two Fingers Up	两指向上滑动
11	Pushing Two Fingers Away	两指向前推出
12	Pulling Two Fingers In	两指向内拉回
13	Rolling Hand Forward	手部向前旋转
14	Rolling Hand Backward	手部向后旋转
15	Turning Hand Clockwise	顺时针旋转手部
16	Turning Hand Counterclockwise	逆时针旋转手部
17	Zooming In With Full Hand	整手向内靠拢
18	Zooming Out With Full Hand	整手向外张开
19	Zooming In With Two Fingers	两指捏合动作
20	Zooming Out With Two Fingers	两指张开动作
21	Thumb Up	竖大拇指
22	Thumb Down	竖倒大拇指
23	Shaking Hand	摇动手掌
24	Stop Sign	举起手掌
25	Drumming Fingers	手指轻敲桌面
26	No gesture	没有检测到手势
27	Doing other things	与手势无关动作

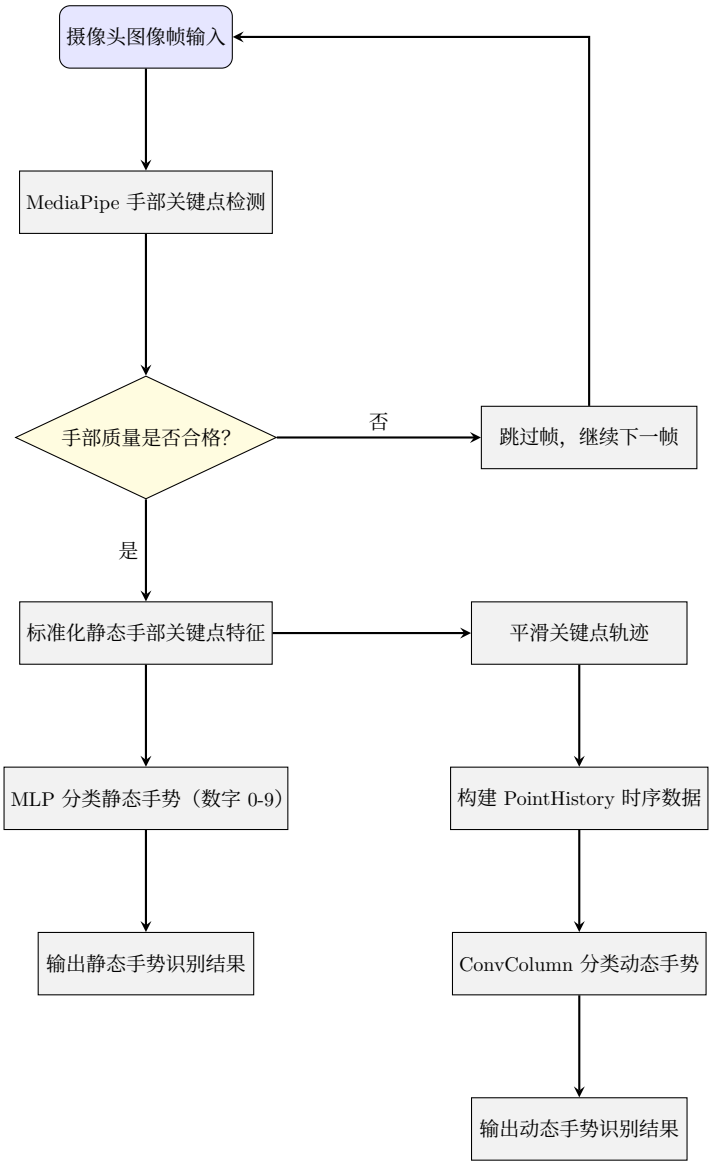

图 5: 手势动作子系统流程图（静态 + 动态识别）

2. 数据集来源

手势动作子系统分别针对静态手势与动态手势设计训练数据来源策略，以确保在轻量部署条件下实现高鲁棒性与覆盖性。

- 静态手势数据集: 静态手势识别部分基于团队自建数据集进行训练。采集过程中，多个用户在不同背景、角度与光照条件下进行手势展示，涵盖十类手势动作。每张图像经 MediaPipe 关键点提取处理为标准化坐标特征，用于训练轻量化 CNN 分类器。采集数据统一存储为 CSV 格式，包含帧编号、类别标签、21 个手部关键点的二维坐标信息。
- 动态手势数据集: 动态手势识别模块使用公开数据集 **Jester Dataset**，由 20BN 公司发布，专用于动态手势识别。该数据集包含超过 148,000 个视频样本，涵盖“Swiping Left-/Right/Up/Down”、“Zooming”、“Pushing”、“Pulling”、“Rolling”、“Turning”等多种日常交互动作。视频帧序列经时序关键点提取后转化为坐标序列，输入至时序模型 ConvColumn 进行学习。Jester 数据集提供良好标签与标准格式，适合训练通用动态手势模型。

- 数据增强策略: 训练阶段对原始坐标序列或图像进行旋转扰动、缩放、遮挡模拟等增强，提升模型在真实场景中的适应性。对于动态手势，额外进行时间轴上的采样与归一化处理，以适配模型输入要求。
- 调用方式: 通过代码中 `load_gesture_classifier()` 与 `load_dynamic_model()` 等函数接口加载静态 CNN 模型与动态 ConvColumn 模型的权重，模型输入数据来源于运行时采集或预处理后的历史轨迹。

3. 模型与训练方法

静态手势识别模型

静态手势识别模型采用手部关键点坐标为输入特征，使用轻量级全连接神经网络进行多分类训练，最终输出对应的手势编号。

1. 模型架构设计：模型采用三层全连接神经网络结构，输入为手部关键点坐标构成的一维向量（21 个关键点 $\times$ 2 维坐标 $=$ 42 维特征）。具体结构如下:

层级	单元数	激活函数	正则化组件
FC1	256	ReLU	BatchNorm + Dropout(0.3)
FC2	256	ReLU	BatchNorm + Dropout(0.3)
FC3	128	ReLU	BatchNorm + Dropout(0.3)
输出层	10	Softmax	–

2. 数据预处理与特征工程输入图像首先通过 MediaPipe Hands 模型检测出 21 个关键点（每个手部关键点包含 x,y,z 三个坐标，共 63 维）。在此基础上进一步构造了增强特征:

- 弯曲角特征：根据指节之间的角度计算手指的弯曲程度。
- 夹角特征：计算拇指与其他手指之间的夹角信息。
- 归一化坐标：将关键点坐标标准化，增强模型在不同手势尺度上的鲁棒性。
- 融合特征向量：最终输入向量长度为 113 维（63 原始坐标 + 50 维几何特征）。

3. 训练策略与参数配置

- 损失函数：交叉熵损失（Cross Entropy Loss）。
- 优化器：Adam（学习率为 0.001）。
- 训练轮次：共训练 200 个 epoch。
- 早停策略：验证准确率连续 10 轮不提升即停止训练。
- 模型保存：保存验证精度最高的模型。
- 学习率衰减：patience=5，factor=0.5。
- 正则化：采用 Dropout 防止过拟合和 L2 weight decay 提升泛化性。

- 训练环境：本地 GPU，批量大小为 64。

4. 模型评估结果

类别	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	866
1	1.00	1.00	1.00	866
2	1.00	1.00	1.00	666
3	1.00	1.00	1.00	666
4	1.00	1.00	1.00	666
5	1.00	1.00	1.00	666
6	1.00	1.00	1.00	666
7	1.00	1.00	1.00	666
8	1.00	1.00	1.00	666
9	1.00	1.00	1.00	666
Accuracy		1.00		7060
Macro avg	1.00	1.00	1.00	7060
Weighted avg	1.00	1.00	1.00	7060

在训练集与验证集分布一致的条件下，模型在静态手势识别任务上表现优异：所有手势类别的精确率、召回率和 F1 分数均为 1.0，总体准确率达到 100%，测试样本数为 7060 个，包含所有手势类别。

动态手势识别模型

动态手势识别模块采用 ConvColumn 架构作为核心模型，融合了三维卷积神经网络（3D CNN）与时序建模思想，专用于提取手势在空间与时间维度上的连续变化特征。该模型具备轻量化、高效性与可部署性，适配于实际人机交互场景中对动作识别的实时性与鲁棒性需求。

1. 模型架构设计：模型输入为大小为 84×84 的单通道图像帧序列，时长为 18 帧，构成输入张量 $(1, 18, 84, 84)$。网络结构共包含 4 个卷积块，每个块由 3D 卷积（Conv3D）、批归一化（BatchNorm）、ELU 激活函数以及最大池化（MaxPooling3D）组成。逐层提取后，特征展平为长度为 12800 的向量，依次通过全连接层（FC1：512 节点，FC2：256 节点）并最终输出 27 维 softmax 概率，覆盖全部手势类别。

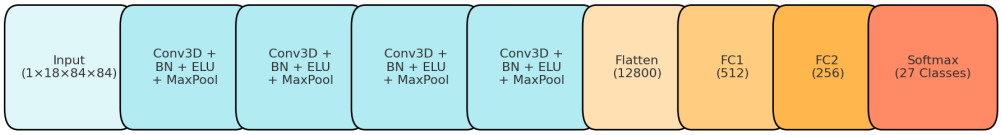

图 6: 动态手势识别模型架构图

2. 输入与数据预处理：模型将缓存的连续图像帧序列作为输入，帧序列通过滑动窗口机制动态更新。图像在送入模型前进行标准化处理，包含缩放、中心裁剪、归一化等步骤，并在训练阶段引入随机翻转与裁剪等数据增强手段，以提升模型在不同光照与背景条件下的泛化能力。

3. 训练方法：训练使用 Jester 动态手势数据集，包含 27 类常见交互动作，如滑动、推动、缩放等。模型以交叉熵作为损失函数，优化器采用 SGD 并结合动量（momentum=0.9）与学习

率衰减机制，初始学习率设为 0.01，权重衰减 5e-4。训练过程中通过 top-1 与 top-5 准确率指标进行监控，并保存验证集性能最优模型权重。

4. 推理与后处理策略：系统实时缓存手部图像帧并判断是否形成稳定片段，随后触发模型预测。输出置信度超过 0.7 且预测类别与上一阶段预测一致时，才认为识别动作有效。此外系统引入冷却时间机制与动作稳定阈值，避免同一手势在短时间内被多次触发，提升输出的稳定性与交互体验。

5. 系统特性与性能总结：模型整体推理速度快、适配多平台部署。在 Jester 数据集测试集上达成 top-1 准确率 92.5%、top-5 准确率 99.3%。可稳定识别如 Swiping、Zooming、Turning 等一系列动态交互动作，广泛适用于智能终端、车载交互、AR/VR 等多模态应用场景。

4. 模块技术细节

模块	技术细节
图像采集	使用 OpenCV 调用摄像头实时读取视频流，统一图像尺寸与帧率，保障输入稳定性。
手部检测与关键点提取	利用 MediaPipe Hands 模型提取 21 个手部关键点（包括手指关节与掌心），用于后续姿态建模与特征提取。
静态手势识别：特征工程	以关键点的相对坐标与角度特征作为输入，构建 1×63 的关键点向量（21 点 $\times$ 3D 坐标），去除尺度与位置影响。
静态手势识别：分类模型	构建三层全连接神经网络（FC1：256，FC2：256，FC3：128），输出层为 softmax（10 类），配合 BatchNorm 与 Dropout 提高鲁棒性。
动态手势识别：预处理	对 18 帧手部图像进行灰度处理与归一化，统一为 84×84 的单通道图像帧，构成 $(1, 18, 84, 84)$ 的时序输入张量。
动态手势识别：建模结构	采用 3D 卷积神经网络（Conv3D + BatchNorm + ELU + MaxPooling）提取时空特征，特征展平后送入两层全连接层（512, 256）分类为 27 个手势动作。
手势质量检测	基于关键点置信度与结构完整性判断手势质量，低质量手势将跳过或降级为静态处理路径。
输出策略与融合	两种模型分别输出手势标签与置信度，统一封装结构化交互指令，包括手势类型、识别概率与时间戳，供交互逻辑调用。

表 4: 手势动作子系统的模块技术细节

（四） 视线方向子系统

1. 功能简介

视线方向子系统主要用于实时检测用户的注视方向（上、下、左、右、中央），基于 MediaPipe 的人脸网格模型（Face Mesh）对眼部关键点进行高精度定位，从而计算虹膜在眼睛轮廓中的相对位置，最终推断用户的注视方向。

在车载多模态智能交互系统中，视线方向子系统负责启动典型场景交互。例如，在异常状态反馈场景下，当视线偏离道路超过 3 秒时，该子系统会启动相应交互；在车载多模态智能交互系统启动时，若视线正视前方超过 3 秒，则会触发自动开启应用功能的交互。

视线方向子系统的具体功能包括：

1. **人脸与眼部关键点检测**

   - 利用 MediaPipe 的 FaceMesh 模型。
   - 在视频帧中检测人脸区域。
   - 精确提取左右眼及虹膜的关键点坐标（包含眼角、上下眼睑与虹膜中心等关键点索引）。
   - 提供关键点置信度检测与可用性判断，以跳过低质量帧。

2. **虹膜相对位置计算**

   - 根据眼部轮廓与虹膜中心点的几何关系。
   - 计算虹膜在水平与垂直方向上的位置比例。
   - 比例值范围为 $[-1, 1]$，反映虹膜相对于眼球中心的偏移程度。
   - 支持左右眼独立计算及融合策略，提升方向判断鲁棒性。

3. **注视方向判断与分类逻辑**

   - 设定水平与垂直阈值（0.45），中央区域判定阈值（0.2）。
   - 根据比例值与阈值比较，判定注视方向为左、右、上、下或中央。
   - 提供简洁的判定函数接口，兼容规则更新与参数微调。
   - 支持左右眼不一致情况的特判处理逻辑。

4. **结果稳定机制**

   - 设置固定长度缓冲区（如 `collections.deque`），缓存历史方向判断结果。
   - 每帧注视结果将进入缓冲区，若缓冲长度达到判定阈值（如 5 帧），则触发稳定性判断。
   - 使用投票机制对缓冲区内方向结果进行统计，取众数作为最终输出。
   - 有效降低偶发跳变与关键点抖动对输出结果的影响。

5. **结构化输出与系统接口**:

   - 将判断结果封装为 `GazeDirectionState` 对象。
   - 包含注视方向类别（枚举类型）、置信度、虹膜比例值、眼部关键点位置等。
   - 支持与 UI 界面联动、日志记录模块对接与后续行为控制模块调用。
   - 具备良好的可扩展性，便于集成入多模态感知系统中。

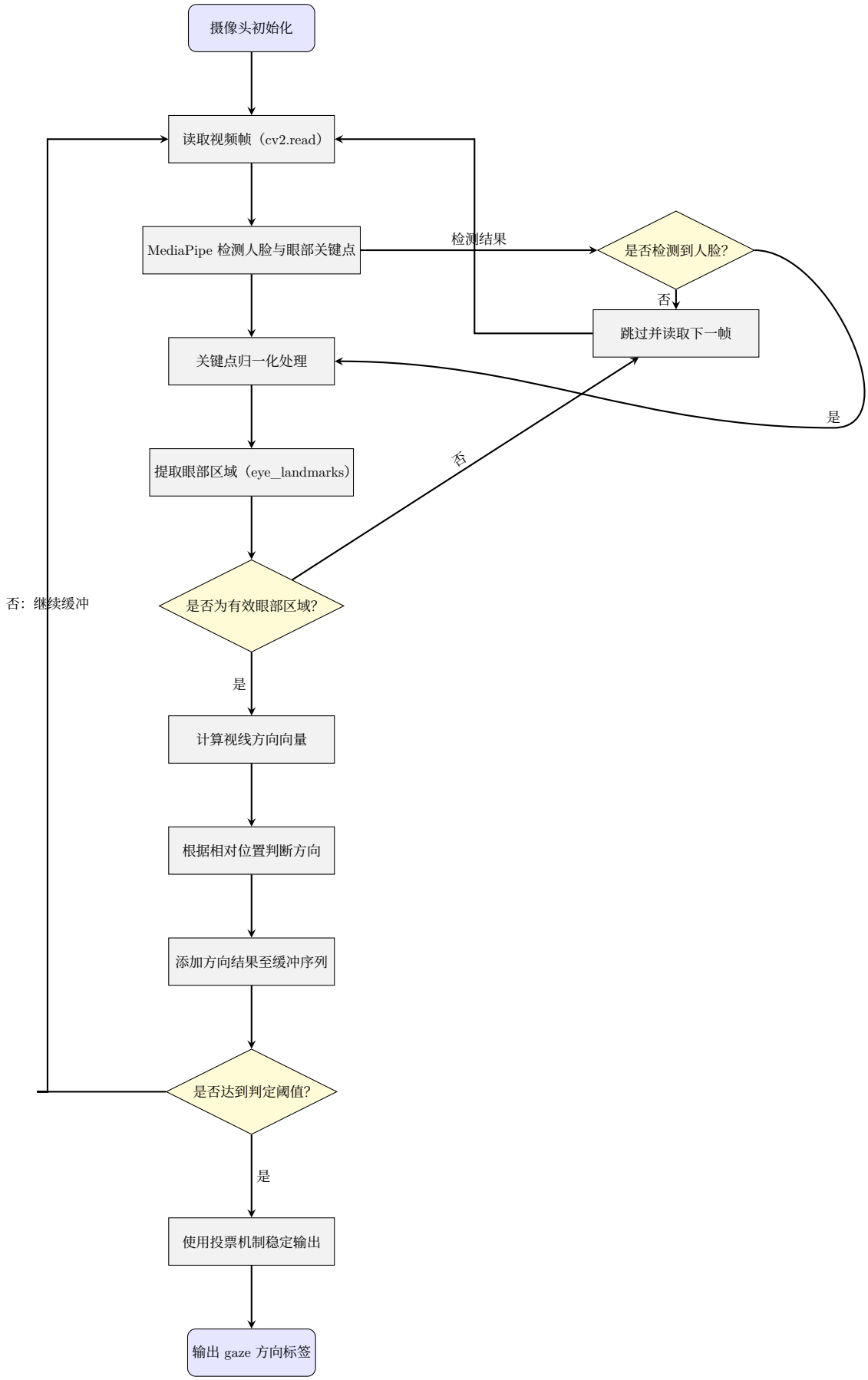

图 7: 视线方向子系统流程图

2. 数据集来源

视线方向子系统主要面向实时视频流进行推理，因此在系统运行阶段依赖摄像头实时采集的数据，而非传统静态数据集进行训练。在模块开发与调试过程中，使用了如下数据源:

1. **实时摄像头数据**:
   - 通过摄像头管理器获取设备摄像头资源;
   - 持续获取视频帧图像作为输入;
   - 用于实时人脸与眼部关键点检测，以及视线方向分析。
2. **MediaPipe 提供的关键点检测模型（Face Mesh）**:
   - 来自 Google MediaPipe 框架;
   - 内置预训练模型可直接输出 468 个面部关键点及虹膜区域;
   - 免去构建本地训练集的复杂流程，显著提升开发效率。
3. **调试用视频片段（可选）**:
   - 在某些测试场景下，为验证模块稳定性，可能使用本地录制的视频流作为输入;
   - 输入路径通过 `cv2.VideoCapture("video.mp4")` 设置;
   - 便于在非实时环境中复现实验场景。

系统主要以摄像头实时视频为数据输入源，结合 MediaPipe 提供的高精度关键点检测模型完成注视方向识别任务，无需额外构建手动标注的数据集。

3. 模型与训练方法

本子系统并未依赖自定义训练模型，而是基于 Google MediaPipe 提供的预训练人脸网格模型（Face Mesh）进行开发。因此，核心视线识别逻辑不涉及传统意义上的模型训练过程，而是依托关键点几何关系进行规则判断。具体实现方法如下:

1. **关键点检测模型**:
   - 使用 MediaPipe 的 `FaceMesh` 模型，该模型已通过大规模面部图像数据训练完成;
   - 可精准提取 468 个面部关键点及虹膜区域的附加关键点，满足视线方向分析的精度要求;
   - 模型为轻量级，适配 CPU 实时运行，便于边缘设备部署。
2. **方向判断算法设计**:
   - 提取左眼和右眼各自的眼角、上下边缘及虹膜中心关键点;
   - 通过几何比例计算虹膜在眼眶内部的相对位置，归一化为 $[-1, 1]$ 的水平（x）与垂直（y）值;
   - 设定方向判定阈值：当 $x < -0.45$ 判定为左看，$x > 0.45$ 判定为右看；$y < -0.45$ 为上看，$y > 0.45$ 为下看；$|x|, |y| < 0.2$ 为注视中央;
   - 利用缓冲序列和投票机制平滑方向输出结果，降低短时波动干扰。
3. **无需再训练特征模型**:

- 系统通过高精度关键点 + 规则判断实现注视方向识别，无需额外构建监督学习数据集；
- 可直接迁移应用于各类环境，无需训练时间与算力资源投入。

该模块以预训练关键点检测为基础，结合轻量级的规则引擎算法，实现了高效、可靠的注视方向识别，具备良好的实时性与可部署性。

4. 模块技术细节

模块	技术细节
模块组成	视线方向估计模块 + 人脸关键点检测模块（MediaPipe FaceMesh）
输入数据	实时视频帧（BGR 图像），转换为 RGB 格式后送入 MediaPipe 处理
输出结构	`GazeDirectionState` 对象，包含注视方向、置信度、虹膜相对位置、眼部关键点
关键依赖	• MediaPipe FaceMesh：检测眼部与虹膜关键点； • OpenCV：图像读取与处理； • `collections.deque`：构造历史缓冲用于投票机制；
处理流程	• 初始化摄像头与 FaceMesh； • 提取眼部关键点与虹膜中心坐标； • 计算虹膜在眼眶中的相对位置（归一化坐标）； • 结合设定阈值判断注视方向； • 利用缓冲区实现结果稳定输出；
调用方式	可通过 `get_gaze_direction()` 独立调用，支持系统嵌入式联动

表 5: 视线方向子系统的模块技术细节