 DOI: https://doi.org/10.12349/iser.v6i12.8472

Design and Implementation of Real-time Control Speech Recognition System Based on Whisper

Peng Zhao Bin Long Weihua Gao

Chongqing Civil Aviation Air Traffic Control Bureau, Chongqing, 401120, China

Abstract

This paper presents a real-time speech recognition system based on the OpenAI Whisper model. The system innovatively adopts a multi-threaded producer-consumer architecture, receiving audio streams from voice recorders via the UDP protocol, converting them into WAV format files, and performing real-time recognition. For practical applications, the system achieves improved recognition accuracy through GPU-accelerated optimization and the use of a fine-tuned Whisper-small model, while delivering end-to-end processing latency below 1 second. This provides a lightweight, scalable engineering solution for aviation air traffic control's speech-to-text services.

Keywords

real-time speech recognition; Whisper model; UDP audio stream; multithreading

基于 Whisper 的实时管制语音识别系统设计与实现

赵鹏 龙滨 高卫华

民航重庆空管分局，中国 · 重庆 401120

摘 要

本文提出一种基于OpenAI Whisper模型的实时语音识别系统。该系统创新性地采用多线程生产者-消费者架构，通过UDP协议接收语音记录仪的监听语音流，转存为WAV格式语音文件并实时识别。针对实际应用场景，通过GPU加速优化及引入微调后的Whisper-small模型有效提高了识别准确率，同时实现了端到端延迟小于1秒的实时处理性能，为民航空管管制语音转文字服务提供了一种轻量级、可扩展的工程解决方案。

关键词

实时语音识别；Whisper模型；UDP音频流；多线程

1 引言

随着全球化航空运输业的快速发展，空中交通管制（Air Traffic Control, ATC）面临着日益增长的航班流量压力与安全监管挑战。国际民航组织（ICAO）数据显示，全球日均航班量已超过 12 万架次，管制员每小时需处理 200-300 条语音指令。传统的人工监听与纸质记录方式不仅效率低下，且难以实现指令回溯与风险预警，已成为智慧空管建设的瓶颈环节。

自动语音识别（Automatic Speech Recognition, ASR）技术的成熟为这一难题提供了技术路径。近年来，深度学习驱动的端到端 ASR 系统在准确率和鲁棒性方面取得了突破性进展。特别是 OpenAI 于 2022 年发布的 Whisper 模型通过 68 万小时多语言、多场景数据的弱监督学习，展现出卓越的跨领域泛化能力与噪声鲁棒性 [1]。其多任务架构同时支持语音识别、翻译、语言识别和语音活动检测，为复杂环境下的语音处理提供了统一框架。

传统语音识别系统多依赖于云端处理，存在延迟高、泄密风险、网络依赖等问题，近年来，随着边缘计算和轻量化模型的发展，离线、本地化语音识别解决方案逐渐成为研究热点。

本文针对民航空管单位对管制语音实时转写需求，设计并实现了一种基于本地微调 Whisper-small 模型的实时管制语音识别系统。主要贡献包括：“（1）提出了适配工业音频流的‘缓冲 - 断句 - 批处理’三级实时化架构；（2）设计并实现了基于生产者 - 消费者模式与 UDP 超时断句的流式处理管线；（3）通过领域微调与 GPU 推理优化，显著提升了专业场景识别准确率与效率；（4）构建了完整的原型系统，验证了方案可行性。”

2 研究现状与相关工作

2.1Whisper 模型的架构特性与挑战：Whisper 模型采用

【作者简介】赵鹏（1976–），男，中国重庆人，硕士，高级工程师，从事空中交通管理通信、导航、监视研究。

编码器 - 解码器 Transformer 结构，支持 99 种语言的语音识别与翻译任务，其大规模弱监督训练策略使其在嘈杂环境下仍能保持稳定性能 [2]。但现有研究主要集中于领域自适应的 fine-tuning 方法，在实时性处理方面，尽管已有 whisperlive 等开源项目实现了流式推理，但此类方案在特定领域（如管制语音识别）的实测中表现仍不理想，普遍存在话语边界检测精度不足与上下文语义连贯性缺失等问题。

2.2 实时语音识别的技术演进

传统实时 ASR 系统长期依赖 Kaldi、WeNet 等框架，采用 CTC/Attention 混合架构实现流式解码 [3]。近期的一些研究虽探索了流式 Transformer 与 RNN-T 模型的低延迟优化，但其准确率与 Whisper 的大规模预训练模型仍存在显著差距 [4][5]。本次工作旨在提供一种 Whisper 模型在实时管制语音数据流处理方面的工程实践方法。

3 系统硬件架构

系统采用分层分布式架构（图 1），由语音记录仪、记录仪控制终端及语音识别服务器构成。其中，话音记录仪用于将管制席位内话终端录音端口输出模拟话音转换为 UDP 音频流；记录仪控制终端用于控制记录仪，将各路监听话音的 UDP 音频流分配到指定服务器 IP 及 port；语音识别服务器用于接收 UDP 音频流并缓存 WAV 文件、实时识别并存储识别记录。系统可以通过调整记录仪和语音识别服务器数量，灵活适配系统规模。

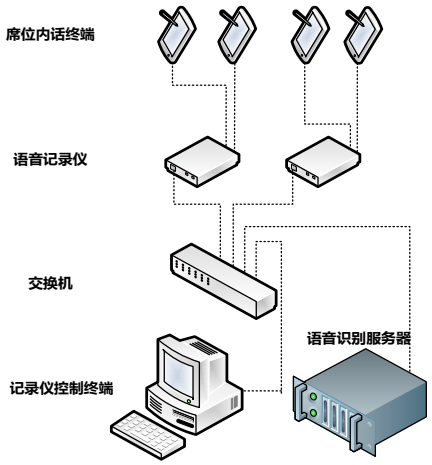

图 1 系统硬件架构

3.1 语音记录仪

系统采用现有货架产品铭道 MDL3002 电话录音仪实现话音采集。此电话录音仪支持常用录音功能，配置灵活，可通过参数调整与现场应用场景进行适配；同时提供了二次开发接口，通过录音通道实时监听输出与我方平台实现对接。

3.2 语音识别服务器

硬件：Intel i7-12700/16GB, NVIDIA RTX 3090/24GB；软　件：PyTorch 2.0, CUDA 11.8, Librosa 0.10；　模　型：whisper-small (fine-tuned on 1290 steps)

4 系统软件架构实现

Whisper 模型的设计哲学基于”大规模离线批处理”，其技术栈（PyTorch+Transformers）与实时流处理存在本质冲突：

Whisper 原生特性	实时处理需求	工程化挑战
30 秒固定窗口	连续语音流	需动态分段与重叠处理
文件 / 内存全量加载	边接收边处理	需环形缓冲与流式解码
同步推理模式	毫秒级响应	需异步化与批处理优化
通用词汇表	航空专业术语	需领域适配与后处理校正

为解决上述矛盾，本文提出 " 缓冲 - 断句 - 批处理 " 三级架构，在保持 Whisper 模型精度的前提下实现实时性。Whisper 模型原生不支持 UDP 语音流直接输入，其设计架构基于文件或内存中的完整音频片段处理，要求输入数据满足以下前提：①完整音频数据：模型需一次性接收完整音频片段（最长 30 秒）才能生成梅尔频谱图；②预处理要求：音频必须经过 whisper/audio.py 模块的预处理，包括重采样、归一化、FFT 变换等；③无网络协议感知：Whisper 库未内置任何网络套接字或流式协议处理能力。

基于 OpenAI Whisper 官方实现和社区实践，Whisper 模型对输入音频的核心要求是 16kHz 采样率、单声道、16-bit PCM、最大时长 30 秒。铭道 MDL3002 录音仪通道参数设置为话音激活时，当录音信号电平超过设定值，监听语音流将会以 UDP 模式输出 alaw 压缩数据。UDP 音频流作为无连接、数据包无序、实时传输的协议，与 Whisper 的批处理架构存在根本性差异。系统设计为缓存 UDP 数据流，并以 UDP 流中断超时为触发条件，将缓存 UDP 音频流数据经过格式转换、重采用处理并封装为 WAV 文件；由 whisper 模型对 WAV 文件进行识别。

4.1 整体框架

系统软件核心采用如图 2 所示的生产者 - 消费者多线程模型，以解耦音频接收与识别推理两个关键环节，并通过共享队列实现高效、线程安全的异步通信。包含两个核心线程：①音频接收线程：负责 UDP 数据包解析、格式转换与缓存；②语音识别线程：负责模型加载、音频特征提取与文本生成推理。通过 queue.Queue 实现线程间通信，确保数据处理的无锁化与异步性。

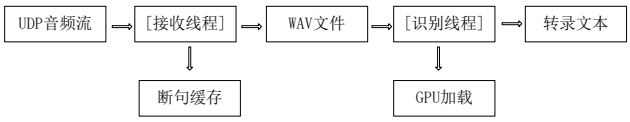

图 2 系统模型

4.2 音频接收线程

4.2.1 UDP 协议处理

线程监听特定端口（如 8001），采用无连接传输降低

延迟。每包最大 1024 字节，包含 12 字节头部与 alaw 压缩语音数据。通过 socket.settimeout(0.5s) 实现断句检测：当接收超时时，将缓存数据判定为完整语句并触发存储。

4.2.2 音频格式转换

输入音频为 8kHz 采样率的 A-law 编码，需转换为 16kHz PCM 格式以满足 Whisper 输入要求。使用 audioop.alaw2lin() 实现线性转换，audioop.ratecv() 实现重采样。相关转换在接收线程中实时完成，计算开销可忽略。

4.2.3 缓存策略

采用 bytearray 动态缓存，当满足以下条件时触发文件写入：①超时条件：130ms 无新数据包，实测 130ms 对于管制员和飞行员分别的对话能较好地分离出来，但仍存在少量输出混合语音的情况；②数据量阈值：最小语音段不小于 0.5 秒

4.2.4 信息通报

接收模块生成 wav 文件的命名格式为：{ 端口号 }_{ 日期 }_{ 时间 }.wav；文件保存路径以端口号、日期进行区分，分类保存到”./{ 端口号 }/{ 日期 }/”目录下。wav 文件生成后将文件名放入语音接收队列。系统可根据语音服务器识别梳理能力，调整音频接收线程数量，通过多路语音流同步接收，异步识别架构，充分利用服务器算力资源。

4.3 语音识别线程

4.3.1 模型加载优化

使用微调后的 Whisper-small 模型有效提高识别准确率。

model = WhisperForConditionalGeneration.from_pretrained(checkpoint_path).to(device)

关键优化：模型常驻 GPU 内存，避免重复加载。首次初始化耗时约 3 秒，后续推理复用模型实例。

4.3.2 特征提取与推理

音频通过 Librosa 加载并重采样至 16kHz，Mel 频谱特征提取由 Processor 自动完成：

inputs = processor(audio, return_tensors=”pt”, sampling_rate=16000)

inputs = {k: v.to(device) for k, v in inputs.items()}

with torch.no_grad():

predicted_ids = model.generate(inputs[“input_features”])

使用 torch.no_grad() 禁用梯度计算，显存占用降低 40%。

4.3.3 异步处理机制

通过队列通信实现多线程协作。线程循环读取接收队列，根据获得 wav 文件名，以事件驱动，实现 whisper 模型对 wav 文件的语音识别。识别结果以如下 JSON 格式进行处理，并通过 websocket 将识别结果同步给后端服务器，格式如下：{ 类型：“识别”，”音频通道号”：UDP_PORT,”数据包编号”：_data_number,”数据包文件”：_wave_path,”识别文字”：text,”角色”：“管制员 | 飞行员 | 混合”}

4.4 用户角色分类和文本切割

管制语音音频源来自内话席位地空录音接口，其输出信号叠加了管制员与飞行员的地空通话内容。实测表明，经数据流转换生成的 WAV 文件中，大部分时段可通过能量检测实现话音分离，仅少数情况下因通话间隔小，导致通话无法切割、音频混杂。由于管制员和飞行员通话有一定规则，为方便管制员区分角色，我们依据《空中交通无线电通话用语》（MH/T4014-2003）设计了一套通话模板。通过模板进行匹配进行角色识别。　　管制员通话示例：< 航班号 > {< 管制区 >} { 你好 }{ 继续 }[上 | 上升 | 下 | 下降] 到 { 反向 }{ 高度 | 标准气压 | 标压 }< 数字组合 >{ 保持 }。飞行员通话示例：[增 | 增加 | 减 | 减小] 速 { 度 }{ 到 }< 数字组合 >{ 保持 }< 航班号 >。对于混合通话，我们采用顺序比对模板方式，将通话文本角色标记为管制员或飞行员。对于无法匹配角色的，标记为混合。

4.5 语音后端处理处理与前端显示

项目后端采用 node.js 设计开发，在收到 websocket 信息后，将 websocket 数据保存到数据库，同时通过 websocket 将 json 数据实时发送到前端页面。前端采用 node.js+vue3 进行设计，根据用户登录的角色（如区调、进近、塔台等），实施分角色显示管制 / 飞行语音文本，并可回放相应音频。

图 3 前端界面

5 结论与展望

5.1 结论

本研究针对民航空管领域对管制语音实时识别的需求，设计并实现了一套基于 Whisper-small 模型的本地化实时管制语音识别系统。通过深入分析 Whisper 模型批量离线处理的设计特性与实时流处理之间的矛盾，创新性地提出了“生产者 - 消费者多线程架构”与“UDP 流缓冲 - 超时断句 - 异步识别 - 角色分类 - 数据推送”的核心工程解决方案。所实现的系统能够稳定接收多路语音流，完成实时转写，并以结构化 JSON 格式输出结果，推送到前端以类似微信气泡方式进行分角色显示。

5.2 不足与展望

尽管本研究取得了预期成果，但受限于时间与条件，仍存在若干可改进之处，未来工作可从以下几个方向展开：更换语音识别模型：因硬件条件限制，本次采用 whisper-small 模型，后续将采用 whisper-large-V3-turbo 或 kimi audio 等模型进行对比测试，提升通用对话识别能力。

领域自适应深化：语料微调数据规模有限。目前我们搜集现场 850 条管制语音识别数据，通过训练，将准确率从 0.3 提升到 0.7。未来将收集更大规模、涵盖更多口音、噪声环境及特情通话的管制语音语料进行持续训练，逐步提升专业词汇识别准确率。说话人分离与角色标注：当前通过配置通话规则模板，能够分离和匹配多数角色文本，但通过规则方式方式无法匹配通话不规范和各种复杂语境的情况，未来需研究通过话音内容和前后上下文等手段，通过训练自动区分管制员与飞行员角色。大规模部署与运维：需设计完善的系统监控、负载均衡与故障转移机制，以支持在大型空管中心多席位、上百路语音流的并发处理。

参考文献

[1] Radford A, et al. Robust Speech Recognition via Large-Scale Weak Supervision[C]. ICML, 2023.

[2] Chen, Y., et al. (2025). Generative AI for Character Animation: A Comprehensive Survey. arXiv:2504.19056.

[3] Watanabe, S., et al. (2017). Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. IEEE Journal of Selected Topics in Signal Processing.

[4] Li, B., et al. (2020). Transformer Transducer: One Model Unifying Streaming and Non-Streaming Speech Recognition. arXiv:2010.03192.

[5] OpenReview. (2025). Dual-Mode ASR: Unify and Improve Streaming ASR with Full-Context Modeling. OpenReview Preprint.