一种智能 采用多智能体架构、用于实时语音处理与任务执行的 AI 眼镜系统

Sheng-Kai Chen[1], Jyh-Horng Wu[2], Ching-Yao Lin[3] 和 Yen-Ting Lin[4]虚实融合与数字孪生部门 国家高速网络与计算中心 台湾新竹
s1134807@mail.yzu.edu.tw[1],{jhwu[2], chingyao[3]}@niar.org.tw,
2303131@niar.org.tw[4]

摘要

本文介绍一种 AI 眼镜系统，该系统整合了实时语音处理、人工智能（AI）智能体和跨网络流媒体传输功能。该系统采用双智能体架构，其中智能体 01 负责自动语音识别（ASR），智能体 02 则通过本地大语言模型（LLM）、模型上下文协议（MCP）工具和检索增强生成（RAG）执行人工智能处理。系统支持通过 RTSP 流媒体传输语音和视频数据、采集眼动追踪数据，并通过 RabbitMQ 消息传递实现远程任务执行。实现结果表明，该系统能够成功处理语音命令，支持多语言，并具备跨平台任务执行能力。

关键词：*AI* 眼镜、语音识别、人工智能、*RTSP* 流媒体传输、眼动追踪、分布式系统

1. 引言

人工智能（AI）眼镜为工业自动化、医疗保健和消费电子领域的免手操作应用提供了变革性的解决方案。然而，目前的 AI 眼镜在实时语音处理、智能任务解释和可靠的跨网络通信方面仍存在明显局限。这些限制阻碍了其在企业环境中的实际部署，而在此类环境中，流畅的语音交互和远程任务执行是关键需求。

将人工智能与增强现实硬件相结合面临着独特的挑战。处理语音命令需要低延迟的语音识别，同时还要在多样化的声学环境中保持准确性。智能任务执行需要先进的自然语言理解，以及对外部数据源和计算资源的访问。网络通信必须支持实时流媒体传输，同时适应不断变化的连接状况和安全要求。

本研究通过一种新颖的多智能体架构应对这些挑战，该架构将语音识别与人工智能驱动的处理分离，并满足实时性能要求。系统结合硬件层面的眼动追踪数据采集和基于软件的语音处理，构建了一个全面的人工智能交互平台。该架构支持在异构网络环境中执行分布式任务，并满足实际应用所需的低延迟通信要求。

2. 相关工作

语音控制增强现实系统已在各种场景中得到探索，但主要聚焦于单组件解决方案。Aouam 等 [1] 展示了用于汽车装配的语音激活增强现实界面，其中语音命令会被识别并转换为文本命令，供增强现实系统解读；但这类方案在嘈杂环境中的准确率有限，且依赖云端处理，因此会产生延迟，难以满足实时应用的要求。

分布式系统中的多智能体架构在复杂任务处理方面展现出良好的应用前景。Fortino 等 [2] 提出了用于物联网系统设计与实现的面向智能体的协作式智能对象框架，但其方法未涉及实时约束，也未满足增强现实应用特有的语音处理需求。

面向增强现实应用的 RTSP 流媒体传输研究主要集中于视频传输。Hosseini 和 Georganas [3] 实现了基于 RTP/RTSP、具备恢复能力的增强型分布式流媒体系统，重点关注视频质量，而非语音与眼动追踪数据的集成传输。增强现实系统中的眼动追踪集成主要聚焦于基于注视的交互机制。Park 等 [4] 在受控环境中开发了采用注视交互的可穿戴增强现实系统，但未涉及移动使用场景或网络韧性需求。

目前的文献缺乏一种全面的解决方案，能够在统一的增强现实系统中整合语音处理、智能任务执行和跨网络通信。本研究采用多智能体方法，同时满足这些需求，填补了这一空白。

3. 方法

1) 系统架构设计

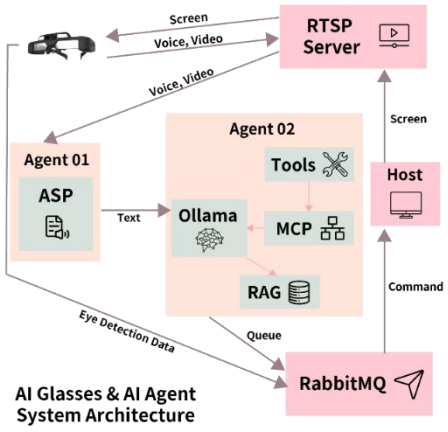

图1 系统架构

系统架构采用分布式设计，由 AI 眼镜硬件、处理代理、网络通信基础设施和远程执行主机组成。该设计将计算职责分配给不同的专用组件，同时通过定义明确的接口确保系统协调运行。

图1展示了整体系统架构。AI 眼镜采集语音、视频和眼动追踪数据，并通过 RTSP协议将其传输至处理代理。智能体 01负责语音识别任务，智能体 02负责智能处理和任务协调。我们采用 RabbitMQ 消息队列，以支持远程主机间的分布式任务执行。

2) 智能体 01：语音识别模块

智能体 01基于 Whisper.cpp 框架实现实时语音识别 [5] ，并针对流式音频处理进行了优化。该模块通过滑动窗口分析处理连续音频流，且缓冲区时长可配置。语音活动检测算法通过过滤静音音频片段来优化处理资源。

音频处理采用滑动窗口方法，每个缓冲区包含的音频采样如式（1）所示。根据式（2）计算连续窗口之间的重叠率，以确保处理过程平稳衔接。重叠可确保处理窗口之间平滑过渡，并防止分段过程中出现音频伪影。语音活动检测根据式（3）应用能量和过零率阈值，以过滤非语音片段并提高处理效率。

$$W_i = \{A(t) : t_{i-1} \leq t \leq t_i\} \tag{1}$$

where $A(t)$ is the continuous audio signal at time t,
and $t_i = t_0 + i \cdot \Delta t$ with Δt being the window duration.

$$\alpha = \frac{t_{i-1} + \Delta t - t_i}{\Delta t} \tag{2}$$

where α represents the overlap ratio between consecutive windows,
t_i is the start time of the current window,
t_{i-1} is the start time of the previous window,
and Δt is the window duration.

$$VAD(W_i) = \begin{cases} 1 & \text{if } E(W_i) > \theta_{energy} \text{ and } ZCR(W_i) < \theta_{zcr} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $E(W_i) = \sum_{t \in W_i} A(t)^2$ is the energy and $ZCR(W_i)$ is the zero-crossing rate.

语音识别管线首先使用经过配置以实现最小缓冲的 FFmpeg 库提取 RTSP 音频流[6] 。音频预处理包括降噪滤波和自动增益控制。多语言支持可通过特定语言的声学模型处理英语和中文语音输入。

3) 智能体 02：智能处理框架

智能体 02 通过整合多个人工智能子系统来协调复杂的推理任务。该框架结合了使用 Ollama [7] 进行的本地大语言模型处理，以及通过模型上下文协议（MCP） [8] 访问外部工具，并通过检索增强生成（RAG）实现内存管理。

意图识别采用多阶段分析流程，将基于规则的模式匹配与神经语言模型处理相结合。意图置信度评分通过加权平均结合模式匹配结果与大语言模型输出，如（4）所示。模式匹配组件使用（5）中定义的子串匹配指标，而上下文相关性则通过向量相似度度量纳入对话历史。

$$C_{intent} = \omega_p \cdot P_s + \omega_l \cdot L_s + \omega_c \cdot C_{context} \tag{4}$$

where $\omega_p + \omega_l + \omega_c = 1$ and $C_{context}$ is the contextual relevance score.

$$P_s(c) = \max_{p \in \mathcal{P}} \frac{|match(c, p)|}{|p|} \tag{5}$$

where $match(c, p)$ returns the matching substring length for command c against pattern set $\mathcal{P}$.

RAG 记忆系统采用 ChromaDB [9] 和 sentence-transformers [10] 进行向量相似度搜索。文档检索通过计算查询与文档嵌入之间的余弦相似度来实现，如式（6）所述。式（7）中的 Top-k检索函数会选出语义相关性最高的文档，以增强上下文。

$$sim(\mathbf{q}, \mathbf{d}_i) = \frac{\mathbf{q} \cdot \mathbf{d}_i}{|\mathbf{q}||\mathbf{d}_i|} \tag{6}$$

where $\mathbf{q}$ is the query embedding vector,
$\mathbf{d}_i$ is the i-th document embedding vector,
$\mathbf{q} \cdot \mathbf{d}_i$ is the dot product,
and $|\mathbf{q}|, |\mathbf{d}_i|$ are the respective vector magnitudes.

$$R_k(\mathbf{q}) = \arg \max_{S \subset \mathbf{D}, |S| = k} \sum_{\mathbf{d}_i \in S} sim(\mathbf{q}, \mathbf{d}_i) \tag{7}$$

where $\mathcal{D}$ is the document corpus,
S is a subset of documents with cardinality k,
and $sim(\mathbf{q}, \mathbf{d}_i)$ is the cosine similarity function.

4) 网络通信协议

该网络架构支持多种连接方式，以适应不同的部署场景。直接RTSP连接适用于局域网环境，而端口转发配置则可通过 NAT穿透机制实现广域网访问。

网络方式的选择采用多准则优化，综合考虑延迟、带宽和可靠性指标，具体如公式（8）所示。网络RTSP管理器通过网络拓扑分析实现自动连接检测。自适应流传输速率会根据可用带宽，使用公式（9）中的指数衰减函数进行动态调整。

$$Score_{method} = \frac{\omega_1}{L + \epsilon} + \omega_2 \cdot B + \omega_3 \cdot R \tag{8}$$

where L is latency, B is bandwidth, R is reliability,
ϵ prevents division by zero, and $\omega_1 + \omega_2 + \omega_3 = 1$.

$$r(t) = r_{max} \cdot \min\left(1, \frac{B_{available}(t)}{B_{required}}\right) \cdot \exp(-\lambda \cdot L(t)) \tag{9}$$

where λ is the latency penalty factor.

对于需要安全通信的企业部署，系统可配置为通过虚拟专用网络连接运行。此功能



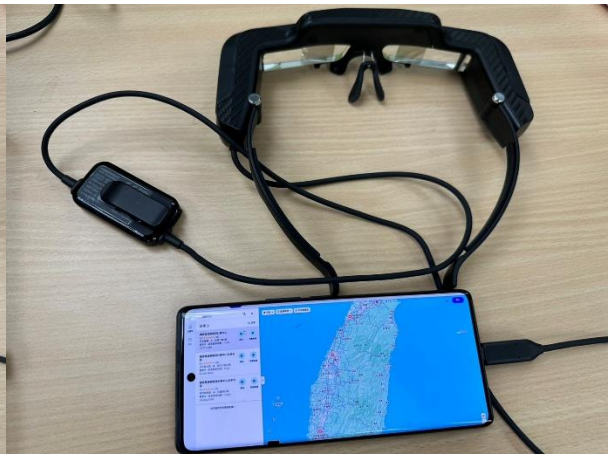

图 2 AI 眼镜中的 UR10 结果 图 3 AI 眼镜中的地图结果

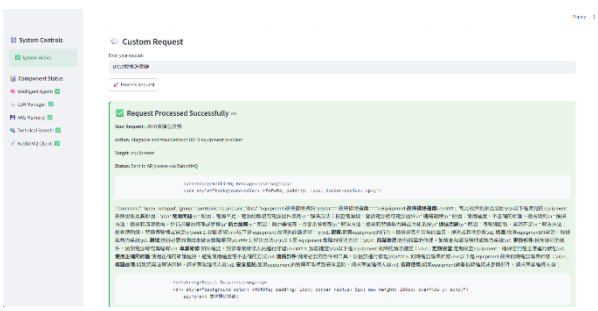

图 4 聊天机器人中的 UR10 请求

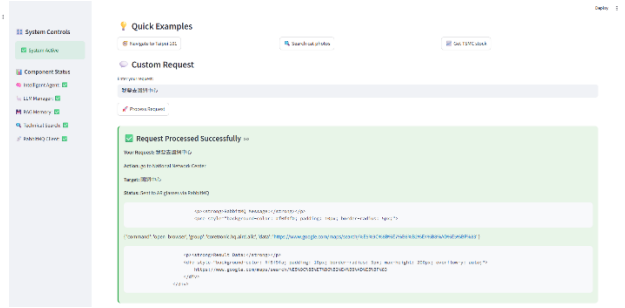

图 5 聊天机器人中的地图请求

满足企业环境中的安全要求，确保语音和视频数据传输符合组织的安全策略。网络管理器会自动检测虚拟专用网络是否可用，并调整流媒体传输参数，以在加密隧道中保持性能，确保 RTSP 流媒体传输和 RabbitMQ 消息传递正常运行，同时符合企业安全标准。

眼动数据采集独立于语音处理运行，以满足时间精度要求。眼动追踪模块通过原生 Android API 连接 Ganzin 硬件传感器，以 30 Hz 的采样频率采集双眼注视向量。

双眼注视融合采用置信度加权平均，如公式（10）所述，其中每只眼睛的权重根据追踪置信度和噪声特性确定，如公式（11）所示。世界坐标变换使用经过校准的旋转矩阵和平移向量，如公式（12）所示。

$$\mathbf{g}_{combined} = \frac{w_L \mathbf{g}_L + w_R \mathbf{g}_R}{w_L + w_R} \tag{10}$$

where $\mathbf{g}_L$ and $\mathbf{g}_R$ are left and right eye gaze vectors respectively, w_L and w_R are confidence-based weights, and $\mathbf{g}_{combined}$ is the fused gaze vector.

$$w_i = \frac{c_i^2}{c_i^2 + \sigma_i^2} \tag{11}$$

where c_i is the confidence score and σ_i is the noise variance for eye i.

$$\mathbf{p}_{world} = \mathbf{R} \cdot \mathbf{g}_{eye} + \mathbf{t} \tag{12}$$

where $\mathbf{R}$ is the calibrated per-user transformation matrix and $\mathbf{t}$ is the translation vector.

实时数据传输采用 RabbitMQ 消息传递 [11]，并使用基于主题的路由。数据序列化将浮点坐标和时间戳转换为 JSON格式，以便通过网络传输；消息持久化则确保数据在网络中断期间的完整性。

5) 分布式任务执行

任务执行利用 RabbitMQ 消息队列，将命令处理与任务执行解耦。系统采用基于主题的路由，根据任务类型和资源需求将任务分发至相应的执行主机。

任务调度采用基于优先级的排队机制，其中任务优先级随等待时间呈指数级递减，具体定义见公式（13）。算法 1 展示了任务执行调度流程，该流程在确保遵守资源约束的同时，最大限度地提高系统吞吐量。

Algorithm 1 Task Execution Scheduling

0: Initialize priority queue Q **while** *system active* **do**
0: $T \leftarrow$ highest priority task from Q **if** $T.resources$ $\leq$ available resources **then**
0: Execute task T
0: Update resource allocation **else**
0: Wait for resource availability
0:
0: =0

$$Priority(T_i) = \frac{U_i \cdot e^{-\alpha \cdot (t_{current} - t_{arrival})}}{D_i} \tag{13}$$

where U_i is the initial task utility,
$t_{current}$ and $t_{arrival}$ are current and task arrival times respectively,
D_i is the decay factor for task i,
and the exponential term prevents task starvation.

Task Executor 框架兼容 Windows、macOS 和 Linux 环境。任务类别包括网页浏览器控制、应用程序启动、文件系统操作和信息显示功能。

4. 结果

1) 系统实现

系统已成功实现，所有主要组件现均已投入运行。双智能体架构有效地将语音识别与智能处理分离，同时通过定义明确的消息接口保持系统协调一致。跨网络通信能力支持系统在各种网络基础设施中运行，并能自动适应不同的连接状况。

2) 功能验证

图 2 展示了 AI 眼镜屏幕上显示的 UR10 机械臂故障查询结果。系统成功在 AR 界面上呈现了全面的故障排查信息，包括中文的设备故障诊断和维修指导。该视觉呈现包含从技术文档系统中检索的结构化诊断流程和维护建议。

图 3 展示了 AI 眼镜屏幕上针对导航查询呈现的谷歌地图位置结果。系统成功呈现地图界面，显示国家电网中心的位置以及相应的地理标记和路线信息。AR 界面与基于网络的地图服务无缝集成，提供与上下文相关的位置数据。

图 4 展示了聊天机器人界面，显示为 UR10 技术支持请求成功传输 RabbitMQ 消息。界面呈现了系统处理设备故障查询，并生成适当的 RabbitMQ 命令消息以传输至 AI 眼镜。消息结构包括命令类型、组路由和完整的技术数据负载。

图 5 展示了聊天机器人界面，其中显示了针对位置请求生成 RabbitMQ 消息的过程。系统成功处理了“我想去 NCHC”这一导航命令，并创建了相应的 RabbitMQ 消息，其中包含用于在 AI 眼镜上显示的谷歌地图 URL 和路线信息。

3) 多语言处理

该系统展现出出色的多语言处理能力，能够准确理解英语和中文语音命令。无论使用哪种语言，系统都能准确处理导航请求、技术支持查询和一般信息请求。意图识别系统无论输入语言为何，都能保持稳定的性能，因此适合在多语言环境中实际部署。

4) 跨平台任务执行

任务执行验证表明，该系统能够在不同平台和任务类型中成功运行。浏览器控制任务能够成功打开相应的网络应用程序和导航界面。通过 RAG 记忆系统和外部工具集成，系统能够有效检索技术文档。信息显示任务能够生成格式规范的 HTML 输出，以便在 AR 显示屏上呈现。

5) 网络架构验证

网络通信基础设施能够在不同网络配置下成功维持连接。RTSP 流媒体传输可高效传输语音和视频数据，而 RabbitMQ 消息传递则通过适当的路由和交付确认，为任务协调提供可靠支持。系统还通过自动重连机制和消息持久化，展现出应对网络中断的韧性。

5. 讨论

双智能体架构成功克服了将语音识别与智能处理相结合所面临的计算挑战，同时保持了系统模块化和可扩展性。通过划分职责，各处理阶段均可独立优化，而不会影响系统整体性能。

智能体 01 专门负责语音识别处理，从而消除了资源竞争，并确保音频处理性能稳定一致。Whisper.cpp 集成展现了出色的实时转录能力，并支持多种语言。然而，顺序处理架构会造成累积延迟，可能无法满足实时交互需求。

智能体 02 集成多个人工智能子系统，证明了将本地大语言模型处理与外部数据访问和内存管理相结合的可行性。RAG 记忆系统在为命令解释提供上下文信息方面表现出良好效果。MCP 工具集成成功扩展了系统能力，突破了本地处理限制。

跨网络通信能力解决了企业环境中的实际部署挑战。自动连接方式检测在维持适当安全协议的同时，减少了管理开销。自适应流媒体方案则在不同网络状况下兼顾了质量与可靠性。

集成眼动追踪可提供有价值的上下文信息，从而提升语音命令解释能力。双目融合算法能够有效稳定注视向量。通过 RabbitMQ 实时传输数据，可满足交互式应用对时间精度的要求。

分布式任务执行框架通过稳健的消息传递基础设施，成功将增强现实系统扩展至 AI 眼镜。跨平台兼容性支持与现有企业计算环境集成。模块化设计便于系统扩展和

无需进行架构修改即可更换组件。

该系统存在一些局限性。首先，分布式任务执行依赖网络连接，且计算需求可能会影响长时间使用时的电池续航。其次，语音数据传输以及敏感命令信息可能暴露给网络基础设施，由此引发隐私问题。

6. 局限性与未来工作

当前系统实现存在若干局限性，也为未来研究指明了方向。网络依赖仍是一项关键制约因素，因为分布式任务执行需要稳定连接才能达到最佳性能。长时间使用期间的电池消耗会影响实际部署，尤其是需要全天运行的移动增强现实应用。

语音数据经网络基础设施传输会引发隐私考量，因此需要增强加密并提升端侧处理能力。顺序处理架构虽具备模块化优势，却会带来累积延迟，可能限制对响应时间要求低于 200 毫秒的实时交互场景。

未来工作应着重于边缘计算集成，以减少对网络的依赖并加强隐私保护。采用联邦学习方法可在保持数据本地性的同时实现模型改进。先进的功耗优化技术，包括动态模型切换和预测性资源管理，有望延长运行时间。

结合语音、注视和手势输入的多模态交互，是实现更直观的增强现实界面的一个有前景的研究方向。与新兴的 5G 和边缘计算基础设施集成，可显著降低延迟，同时支持更先进的人工智能处理能力。

7. 结论

本研究通过多智能体架构和分布式处理能力，展示了实现智能 AI 眼镜系统的一种实用方法。该系统在各种运行场景中，实现了语音识别、人工智能驱动的处理和任务执行的功能集成。

双智能体设计成功克服了计算方面的挑战，同时保持了系统模块化和可扩展性。跨网络通信能力支持在企业环境中进行实际部署，并可根据不同的连接状况自动调整。此外，多语言处理支持使系统能够在多种语言环境中运行。

分布式任务执行框架借助可靠的消息传递基础设施，使增强现实系统的能力突破本地处理的限制。跨平台兼容性使其能够与现有计算环境集成，而无需部署专用硬件。

未来的研究方向包括探索边缘计算方法以降低对网络的依赖，实施隐私保护技术以保障语音数据安全，以及研究结合语音、注视和手势输入的多模态交互。对于需要长时间运行的移动增强现实应用而言，功耗优化仍是一项关键问题。

该系统架构为先进的 AI 眼镜应用奠定了基础，这类应用需要强大的语音交互能力以及对分布式计算资源的访问。其模块化设计有助于持续开发，以适应增强现实硬件能力和人工智能处理技术的不断发展。

参考文献

[1] Aouam, T., Sahnoun, S., & Hamid, M. (2018). “汽车零部件组装的基于语音的增强现实交互系统。” IEEE计算机与通信工程技术国际会议。[2] Fortino, G., Russo, W., Savaglio, C., Shen, W., & Zhou, M. (2017). “面向智能体的协作式智能对象：从物联网系统设计到实现。” IEEE《系统、人类与控制论：系统》汇刊，48(11)，1939-1956。[3] Hosseini, M., & Georganas, N. D. (2007). “基于 RTP/RTSP 且具备恢复能力的增强型分布式流媒体系统。” IEEE高级信息网络与应用国际会议。[4] Park, H. M., Lee, S. H., & Choi, J. S. (2008). “使用注视交互的可穿戴增强现实系统。” 第七届 IEEE/ACM混合与增强现实国际研讨会论文集（ISMAR '08）。[5] OpenAI. (2022). Whisper：通过大规模弱监督实现稳健的语音识别。arXiv 预印本 arXiv:2212.04356。获取地址：https://github.com/openai/whisper[6] FFmpeg开发者。(2023). FFmpeg：用于录制、转换和流式传输音频与视频的完整跨平台解决方案。软件文档。获取地址：https://ffmpeg.org/[7] Ollama团队。(2023). Ollama：在本地快速启动并运行大语言模型。GitHub代码库。获取地址：https://github.com/ollama/ollama[8] Anthropic。(2024). 模型上下文协议：用于连接人工智能模型与数据源及工具的通用协议。技术规范。获取地址：https://github.com/modelcontextprotocol/[9] ChromaDB。(2023). Chroma：面向人工智能的开源嵌入数据库。GitHub代码库。获取地址：https://github.com/chroma-core/chroma [10] Reimers, N., & Gurevych, I. (2019). Sentence-BERT：使用孪生BERT网络的句子嵌入。2019年自然语言处理经验方法会议论文集，3982-3992。获取地址：https://github.com/UKPLab/sentence-transformers[11] VMware Tanzu。(2023). RabbitMQ：最初实现高级消息队列协议的开源消息代理。软件文档。获取地址：https://github.com/rabbitmq/rabbitmq-server