基于消费类视听终端操作系统的AI代理终端实现技术研究调研报告

目录

[一、目标 2](#_Toc188629260)

[二、架构设计 2](#_Toc188629261)

[三、TVOS 2](#_Toc188629262)

[3.1 文献调研 2](#_Toc188629263)

[3.2 参考文章 3](#_Toc188629264)

[四、Agent 3](#_Toc188629265)

[4.1 分析 3](#_Toc188629266)

[4.2 Agent架构 4](#_Toc188629267)

[4.2.1 文献调研 4](#_Toc188629268)

[4.2.2 参考文章 4](#_Toc188629269)

[4.3 规则引擎 4](#_Toc188629270)

[4.3.1 开源代码 4](#_Toc188629271)

[4.4 搜索引擎 4](#_Toc188629272)

[4.4.1 开源代码 4](#_Toc188629273)

[4.5 本地模型 4](#_Toc188629274)

[4.5.1 文献调研 4](#_Toc188629275)

[4.6 特殊模型 5](#_Toc188629276)

[4.6.1 文献调研 5](#_Toc188629277)

[4.7 任务执行 5](#_Toc188629278)

[4.7.1 Plan-and-Execute：规划执行分离策略 5](#_Toc188629279)

[4.7.2 ReAct：思考-行动循环机制 5](#_Toc188629280)

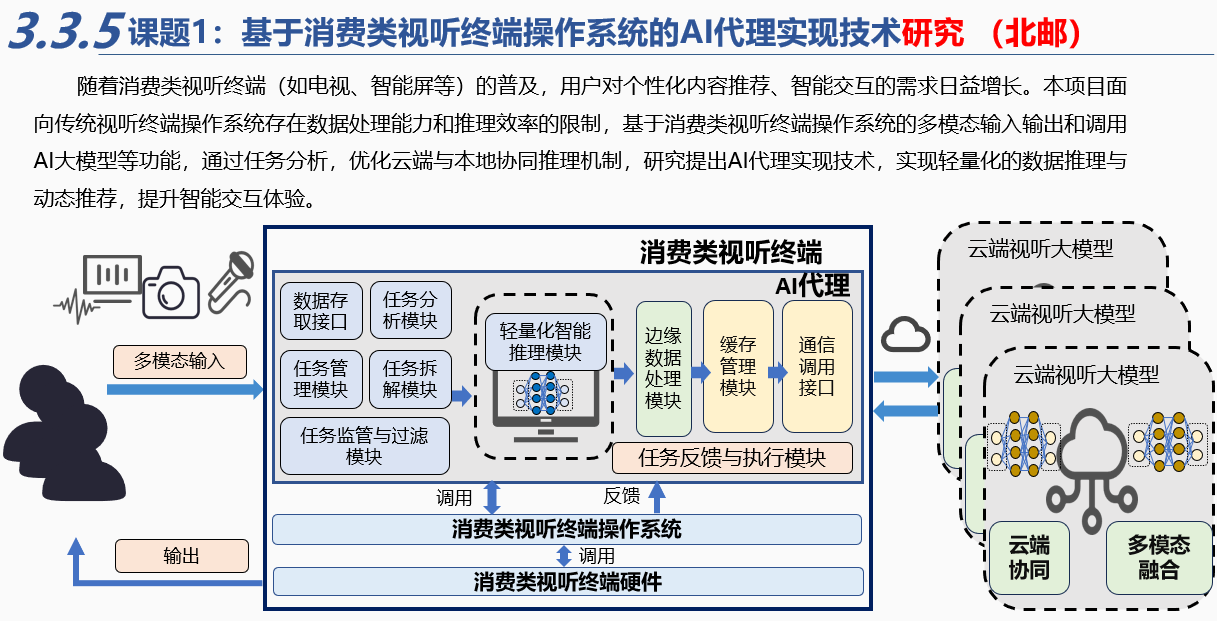
[4.8 参考文章 5](#_Toc188629281)

# 一、目标

基于消费类视听终端操作系统的AI代理终端实现技术研究

面向传统视听终端操作系统数据处理能力限制，基于消费类视听终端操作系统多模态输入输出和调用AI大模型等功能，研究AI代理实现技术，开发面向操作系统的应用层AI代理应用，通过任务监管与过滤模块、任务分析模块、轻量化智能推理模块、任务拆解模块、云端视听大模型交互模块、反馈和执行模块，结合云端与本地协同推理提升智能交互体验。

# 二、架构设计



# 三、TVOS

DVB+OTT形态的综合宽带广播（IBB）融合智能终端是智能终端的必然发展趋势。TVOS组件统一采用C/S架构，一个组件可同时支持多个应用（组件化思想）；还支持组件的裁剪与组合，方便跨应用平台重用。

鸿蒙生态和TVOS都支持Java、Python和Web，暂定选择开发语言为Python和Java（可根据组件特性调整）。

## 3.1 文献调研

1. TVOS开发实例：（可参考架构）（已读）

[TVOS电视在适老康养领域的技术应用和实践探索 - 中国知网 (cnki.net)](https://kns.cnki.net/kcms2/article/abstract?v=uQDmaVEYwcwT2G3e2Qf4yJ0_l3_VveBl02IMC8U1vZbTz717MnXA11Io-ncm1gQF2WL8snfw_vQtVidqpsrabaLRJgeM1R5QO5NOY7w_hmRdUFKv-Kk_rxIJ4NKgBjUxC9OIUUtYx8uMV6OnNxx1AeMp_YVZqRDVEo9BzXcof42LXi2isIsLD2XLjwTtpAYo&uniplatform=NZKPT&language=CHS)

简述：本文产品采用公共的IaaS 层+PaaS 层的通用服务，且只开发与业务相关的SaaS 层应用。主体应用、数据调用、数据库运行和存储的应用软件采用 B/S 架构**（TVOS架构是CS，但是外置挂载数据库为BS）**。主要分为系统网络架构、网络安全架构（防火墙进行网络分离）和数据业务架构，其设计思路具有指导意义。

1. TVOS软件设计规范（浅读）

[面向TVOS系统的能力解耦开发方法及系统模型 - 中国知网 (cnki.net)](https://kns.cnki.net/kcms2/article/abstract?v=fAsipVvIRrgoy4R68Svxye1q-kWSPMmxvvX3bFNfwS-AcBBiYgiKtCzAlC0oW9S-h9os5jZPPRZuJyhKjoua0V3VQq80I8e9T6FpMDNaTBdvLs8pvGCHta54u_6LIgcSSSJq5HEbiLCO4gEOWuY-tLS2Rigsq4jviBSxsFvOWRfDtdLE7qvM5FvwwhsWSAPl&uniplatform=NZKPT&language=CHS)

简述：将应用APK 的开发分为应用APK 前端（用户端）和应用APK 后端（服务端）2 个部分，从而将依赖终端的功能需求和用户交互需求按功能组件模块实现分离，给出面向TVOS 系统完整的应用APK 开发方法及其系统模型，系统架构设计具有参考意义。

1. China DRM技术，自主可信的数字媒体保护技术的全流程技术体系：（不用读）

[广西广电基于TVOS智能机顶盒的ChinaDRM技术研究与实现 - 中国知网 (cnki.net)](https://kns.cnki.net/kcms2/article/abstract?v=uQDmaVEYwcxGyC8J_0CCGd84tXwcoa19FZ1Ep-E658Qq4Eoz9hVHJzJYwd7VmOSuulif0eIYtakMmdD9qLEJXOS32gcfKBhES4Y9JHqMHxXINyUxxOiAgemL3qbYU4IyFpAn9lIorq_vFr1PbAIDCghjgbyLlP6dOP4TAMkwhTxDQgUfa9IZLhTCsSuaYocx&uniplatform=NZKPT&language=CHS)

简述：ChinaDRM是TVOS核心功能，为应用层提供公共服务支撑能力，包括媒体引擎、DRM、多屏互动、数字电视、安全支付、网络协议、存储管理、AV 设置、应用管理、智能家居、数据采集、通信等共用功能组件模块。Agent系统开发时需要大量调用该服务提供的接口。

1. TVOS与组网相结合：（浅读）

[基于TVOS机顶盒应用的智慧医养服务平台刍议 - 中国知网 (cnki.net)](https://kns.cnki.net/kcms2/article/abstract?v=ZkvsKdWJ3SQ8MzxvJGUphSAYrOLwsOnuLrWqxOLfRZSstQnAL2lFuVMzaVAlKLE_4-jWbk91mmiQ8xvGtnl3iHxD5DAew68zYof1ZsEa1jefLWcOxXrMSoVk2CRA_mowizIGYr_eRy497yy45zc7diTQL2a_vTHPgQIjKzBeK6r6A3Isif7zhWnMWG2EKTAH&uniplatform=NZKPT&language=CHS)

简述：与摄像头外设进行搭配的实际产品，可能具有参考意义。

## 3.2 参考文章

移动端侧边小模型

[端侧Agent系列 | 端侧AI Agent任务拆解大师如何助力AI手机？(详解版)-CSDN博客](https://blog.csdn.net/ljp1919/article/details/142701122)

简述：因为TVOS是类android，用octo-planner拆解任务，所以可以参考。文章用lora进行了实验。

# 四、Agent

## 4.1 分析

Agent = LLM + Planning + Feedback + Tool use

本架构主要分为：任务监管与过滤模块、任务分析模块、轻量化智能推理模块、任务拆解模块、云端视听大模型交互模块、反馈和执行模块。

1. **音频处理（语音识别、合成、音频分析）**
2. **视觉处理（视频理解、图像处理、内容生成）**
3. **多模态交互（视听结合）**
4. 任务分析与任务拆解就是一些针对场景的预设**prompt**
5. 任务监管与过滤：接收并处理来自用户的任务请求；对任务请求进行过滤，去除无效或不合法的请求；对合法任务进行监管，确保任务按照预期执行。

云试听相关模型：

* 任务处理阶段： Flamingo（多模态输入->文本输出），BERT（文本输入输出）
* 执行阶段：sora（text->video），DALL.E（text->picture）

国内的云试听相关：**阿里云通义千问，百度文心一言，字节跳动豆包**

对于模型本身的论文就不调研了，不系统注重应用。针对任务推理拆解，还是偏向于**Plan-and-Execute**思路**（但是更消耗token）**。

## 4.2 Agent架构

### 4.2.1 文献调研

1. AI agent综述（浅读）

[[2401.03568] Agent AI: Surveying the Horizons of Multimodal Interaction (arxiv.org)](https://arxiv.org/abs/2401.03568)

简述：主要用来了解大模型再在Agent中的应用方式和一些设计思路。

### 4.2.2 参考文章

[[译] AI Workflow & AI Agent：架构、模式与工程建议（Anthropic，2024） (arthurchiao.art)](https://arthurchiao.art/blog/build-effective-ai-agent-zh/)

简述：超有参考价值的文章，描述了Agent，Workflow的场景，细节和设计思路。

## 4.3 规则引擎

### 4.3.1 开源代码

easy-rules

[GitHub - j-easy/easy-rules: The simple, stupid rules engine for Java](https://github.com/j-easy/easy-rules)

简述：Java的规则引擎，用于决策本地推理还是云端交互

## 4.4 搜索引擎

### 4.4.1 开源代码

search4all

[GitHub - fatwang2/search4all: Personal AI search copilot, open-source Perplexity](https://github.com/fatwang2/search4all)

简述：开源AI搜索引擎，具有整合多个大模型结果的能力，在任务监管模块处理过后针对场景一和场景二可以进行搜索和推荐（可根据任务细化）。

## 4.5 本地模型

### 4.5.1 文献调研

1. DistilBERT

[[1910.01108] DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (arxiv.org)](https://arxiv.org/abs/1910.01108)

简述：对原始BERT模型进行 distillation(蒸馏)后的一个**更小、更快、更便宜**的版本，获得了体积仅原始BERT模型40%、但语言理解能力保持97%的**DistilBERT**模型。

模型地址：[DistilBERT (huggingface.co)](https://huggingface.co/docs/transformers/model_doc/distilbert)

中文化：[GitHub - CLUEbenchmark/DistilBert: DistilBERT for Chinese 海量中文预训练蒸馏bert模型](https://github.com/CLUEbenchmark/DistilBert)

## 4.6 特殊模型

### 4.6.1 文献调研

1. 特定场景下的模型（未读）

[Playing football game using AI agents | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9921374)

参考：主要用来应对场景三，预测AI的运动员下一步举动。

## 任务执行

### 4.7.1 Plan-and-Execute：规划执行分离策略

[[2305.14564] PEARL: Prompting Large Language Models to Plan and Execute Actions Over Long Documents (arxiv.org)](https://arxiv.org/abs/2305.14564)

Plan-and-Execute 模式采用"先规划后执行"的策略，将任务分为两个明确的阶段：

1. **规划阶段（Planning）**：
   * 分析任务目标
   * 拆分子任务
   * 制定执行计划
2. **执行阶段（Execution）**：
   * 按计划顺序执行子任务
   * 处理执行结果
   * 调整执行计划（如需要）

### 4.7.2 ReAct：思考-行动循环机制

[[2210.03629] ReAct: Synergizing Reasoning and Acting in Language Models (arxiv.org)](https://arxiv.org/abs/2210.03629)

ReAct（Reasoning and Acting）模式是一种"思考-行动"交替进行的推理模式。其核心工作流程为：

1. **思考（Reasoning）**：分析当前状态和目标
2. **行动（Acting）**：执行具体操作
3. **观察（Observation）**：获取行动结果
4. **循环迭代**：基于观察结果继续思考和行动

## 4.8 参考文章

[ReAct vs Plan-and-Execute：LLM Agent 模式实战对比 - muzinan110 - 博客园 (cnblogs.com)](https://www.cnblogs.com/muzinan110/p/18552824)

内容有代码示例，如prompt，很具有参考价值。

[LLM Powered Autonomous Agents | Lil'Log (lilianweng.github.io)](https://lilianweng.github.io/posts/2023-06-23-agent/)