Arquitecturas de Agentes Multimodales y Sistemas de Control por Voz: Un Marco Integrado para la Accesibilidad Universal en Dispositivos Móviles

La evolución de las interfaces de usuario hacia ecosistemas puramente visuales ha generado una brecha de accesibilidad significativa para las personas de la tercera edad y aquellas con deficiencias visuales severas. La creación de una solución de control total mediante voz que sea capaz de operar cualquier aplicación de manera agnóstica requiere una convergencia de procesamiento de lenguaje natural (NLP), visión por computadora y, fundamentalmente, una **arquitectura de memoria persistente** que almacene la estructura de las aplicaciones para una ejecución eficiente.

1. Procesamiento de Voz y Reconocimiento de Patrones Acústicos

El primer componente crítico es la transformación de la señal acústica en texto. Para usuarios adultos mayores, los modelos deben manejar variaciones en la cadencia y claridad fonética. OpenAI Whisper es el estándar actual debido a su entrenamiento masivo, permitiendo robustez ante ruidos de fondo.[1]

Herramienta	Arquitectura	Latencia Estimada	Capacidad Local
Whisper Large V3	Transformer Encoder-Decoder	Alta (1-2s)	Sí [1]
Whisper Turbo	Optimizada (809M params)	Media (400-800ms)	Sí
Vosk	Kaldi-based	Muy Baja	Sí (Muy ligera)

Para una tesis, la implementación de **Whisper.cpp** permite ejecutar estos modelos directamente en el hardware móvil mediante Vulkan o Metal, garantizando la privacidad del usuario al procesar todo *on-device*.[4]

2. Percepción Visual y Detección de Componentes (Zero-Label)

Para que el sistema sea agnóstico, no debe depender de las etiquetas de los desarrolladores. El sistema debe "ver" la interfaz y generar semántica dinámica.

- **OmniParser:** Este modelo de Microsoft descompone capturas de pantalla en elementos

interactuables (iconos, botones) y les asigna etiquetas numéricas o semánticas sin acceder al código fuente.[7]
- **ScreenAI y Florence-2:** Estos modelos visuales (VLM) permiten identificar la función de un icono (ej. un sobre como "enviar") basándose exclusivamente en su apariencia visual.[7]

3. Arquitectura de Memoria: El Grafo de Conocimiento del Sistema

Para resolver el problema de las aplicaciones con múltiples pantallas y *scrolling*, el sistema no debe ser reactivo (analizar solo lo que ve), sino **proactivo**, basándose en una base de datos local de la estructura de las apps.

3.1 Fase de Exploración Offline (Crawling)

Antes del uso diario, el sistema puede realizar una exploración sistemática de las aplicaciones instaladas utilizando herramientas como **xTester** o algoritmos de búsqueda en anchura (BFS).

- **Construcción de UTGs (User Transition Graphs):** El agente navega automáticamente por la app, ejecuta scrolls y clics, y registra cada pantalla como un "nodo" y cada acción como una "arista" en un grafo.
- **Manejo de Scrolling:** Al explorar, el sistema realiza capturas de pantalla extendidas o múltiples clics de scroll para mapear elementos que no son visibles en la primera carga, guardando sus coordenadas relativas en la base de datos.[9]

3.2 Almacenamiento en Base de Datos Vectorial (KG-RAG)

Los datos extraídos (IDs de componentes, descripciones semánticas, rutas de navegación) se transforman en *embeddings* y se almacenan en una base de datos vectorial local (como **ToolNeuron** o bases integradas).

- **Búsqueda por ID de App:** Cuando el usuario abre una aplicación, el sistema detecta el ID del paquete (package name) y recupera instantáneamente el "mapa" de acciones posibles de la base de datos.
- **Match de Voz:** El mensaje de voz del usuario se compara mediante similitud de coseno con las descripciones de las acciones almacenadas. Esto permite que, aunque un botón de "Enviar" esté oculto tras un scroll, el sistema sepa que existe y ejecute el desplazamiento automáticamente para alcanzarlo.

4. Toma de Decisiones y Ejecución de Patrones

La toma de decisiones se basa en un modelo de **Máquina de Estados Finitos (FSM)**. Cada pantalla es un estado; si el usuario da una orden, el sistema calcula la ruta más corta en el grafo de conocimiento para llegar al estado objetivo.[11]

4.1 Estrategia de Verificación y Seguridad

Dado que algunas acciones son irreversibles (ej. realizar un pago), se recomienda integrar un

sistema de **VeriSafe** o **V-Droid**:

- **Pre-acción:** El sistema formaliza la intención del usuario en una especificación lógica antes de actuar.
- **Confirmación:** Si la confianza del patrón más probable es baja o la acción es crítica, el sistema solicita confirmación por voz antes de ejecutar el clic.

4.2 Ejecución a Nivel de Sistema

Para controlar cualquier aplicación con privilegios totales sin necesidad de *root*, la herramienta clave es **Shizuku**:

- **Elevación de APIs:** Shizuku permite a tu aplicación llamar a APIs de sistema ocultas mediante ADB, permitiendo inyectar eventos de toque, texto y gestos de forma mucho más rápida y estable que el AccessibilityService estándar.
- **Simulación de Gestos:** Permite ejecutar secuencias complejas (swipe + click) para navegar a través de los patrones guardados en la base de datos.

5. Diseño para Adultos Mayores y Déficit Visual

El éxito de la tesis depende de la usabilidad. La interfaz debe seguir los principios **POUR** (Perceivable, Operable, Understandable, Robust):

- **Retroalimentación háptica y auditiva:** Cada detección exitosa debe confirmarse con una vibración o un mensaje sonoro corto.
- **Navegación Circular:** En los menús de la app de control, se debe permitir que el usuario navegue cíclicamente para evitar que se pierda en listas largas.[1]
- **Simplicidad Cognitiva:** Evitar comandos anidados; el sistema debe ser capaz de inferir el contexto (ej. si el usuario dice "Dile a Juan que llego tarde", el sistema debe saber buscar a Juan en contactos y abrir la app de mensajería preferida).[12]

Conclusiones para la Tesis

Es totalmente posible lograr esta hazaña mediante un enfoque **híbrido**:

1. **Exploración Offline:** Crear el "mapa" (UTG) de las apps una sola vez.
2. **Memoria Local:** Guardar este mapa en una base de datos vectorial (KG-RAG) vinculada al ID de la aplicación.
3. **Razonamiento On-Device:** Usar un LLM pequeño (ej. MiniCPM-V o Qwen-VL) para emparejar la voz con la base de datos y decidir la ruta de navegación.
4. **Ejecución Privilegiada:** Utilizar Shizuku para el control total del hardware sin las limitaciones de la capa de usuario.

Obras citadas

1. Best open source speech-to-text (STT) model in 2026 (with benchmarks) | Blog - Northflank, fecha de acceso: febrero 13, 2026, https://northflank.com/blog/best-open-source-speech-to-text-stt-model-in-2026-benchmarks

2. Is Whisper the Best Speech-to-Text Software? - Slator, fecha de acceso: febrero 13, 2026, https://slator.com/resources/is-whisper-the-best-speech-to-text-software/
3. OpenAI Whisper vs Google Speech-to-Text vs Amazon Transcribe: The ASR Rundown, fecha de acceso: febrero 13, 2026, https://www.gladia.io/blog/openai-whisper-vs-google-speech-to-text-vs-amazon-transcribe
4. Android example app #283 - ggml-org whisper.cpp - GitHub, fecha de acceso: febrero 13, 2026, https://github.com/ggml-org/whisper.cpp/discussions/283
5. My experience with whisper.cpp, local no-dependency speech to text - Reddit, fecha de acceso: febrero 13, 2026, https://www.reddit.com/r/LocalLLaMA/comments/1fcfcx6/my_experience_with_whispercpp_local_nodependency/
6. A curated list of awesome TensorFlow Lite models, samples, tutorials, tools and learning resources. - GitHub, fecha de acceso: febrero 13, 2026, https://github.com/iglaweb/awesome-tflite
7. OmniParser: Vision Based GUI Agent - Learn OpenCV, fecha de acceso: febrero 13, 2026, https://learnopencv.com/omniparser-vision-based-gui-agent/
8. KG-RAG: Enhancing GUI Agent Decision-Making via Knowledge Graph-Driven Retrieval-Augmented Generation - arXiv, fecha de acceso: febrero 13, 2026, https://arxiv.org/html/2509.00366v1
9. ScreenAI: A visual language model for UI and visually-situated ..., fecha de acceso: febrero 13, 2026, https://research.google/blog/screenai-a-visual-language-model-for-ui-and-visually-situated-language-understanding/
10. Setup · d4rken-org/sdmaid-se Wiki - GitHub, fecha de acceso: febrero 13, 2026, https://github.com/d4rken-org/sdmaid-se/wiki/Setup
11. Create your own accessibility service | Android Developers, fecha de acceso: febrero 13, 2026, https://developer.android.com/guide/topics/ui/accessibility/service
12. App development | Android Open Source Project, fecha de acceso: febrero 13, 2026, https://source.android.com/docs/automotive/voice/voice_interaction_guide/app_development
13. TencentQQGYLab/AppAgent: AppAgent: Multimodal ... - GitHub, fecha de acceso: febrero 13, 2026, https://github.com/mnotgod96/AppAgent
14. Diseño de interfaces de voz en la Web - ITDO, fecha de acceso: febrero 13, 2026, https://www.itdo.com/blog/diseno-de-interfaces-de-voz-en-la-web/