

Project P4: Agentic Computer Interaction with Vision and Speech

Objective: Create a voice-controlled UI automation system using OCR and MCP workflows

Core Functionality: "Click the Settings button" → System captures screen → OCR detects elements → Executes click

Architecture: Hybrid MCP (mcp-agent for workflows + FastMCP for HTTP transport)

Key Components Implemented:

Component	Technology Used	Purpose
Vision	EasyOCR	Pixel-perfect text and coordinate detection
Automation	PyAutoGUI	Mouse movement and clicking
Workflow	mcp-agent	Sequential task execution
Communication	FastMCP	HTTP transport for tool calls
Interface	Open-WebUI	Chat-based interaction with MCP tools
Audio	gTTS + playsound	Text-to-speech feedback

Workflows Implemented:

1. CaptureScreenWithNumbers Workflow:

Captures screenshot → Runs OCR → Creates number-to-text mappings → Saves JSON

2. ClickWorkFlow:

Loads mappings → Finds target → Converts coordinates → Executes click → Provides feedback

3. VanillaWorkflow & Echo Tool:

Basic screenshot capture and voice command echo

Challenges Faced & Solutions:

1. OCR Accuracy and Performance:

Problem: Initial attempts with PaddleOCR caused system hangs; OpenRouter VLM provided inaccurate coordinates.

Solution:

- Switched to EasyOCR: Provided reliable, local OCR without API dependencies
- Added confidence filtering: Ignored low-confidence detections (<30%)

- Implemented coordinate scaling: Proper conversion from image pixels to screen coordinates
- Fallback mechanism: Mock data when OCR fails (for testing continuity)

2. Coordinate Mismatches:

Problem: Clicking at wrong locations due to: Incorrect scaling between screenshot and screen coordinates, OCR bounding box center miscalculation, Screen resolution differences

Solution:

Key calculation in processdetectionstomappings()

screenx = pixelx / scalex # scalex = screenwidth / imgwidth

screeny = pixely / scaley # scaley = screenheight / imgheight

3. Text-Number Mapping:

Problem: Associating numbered labels (1, 2, 3) with actual UI text elements.

Solution:

- Pattern matching: `re.match(r'^(\d+)(.*)', text)` for embedded numbers
- Adjacency detection: Pairing nearby numbers with text elements
- Position-based assignment: When no numbers detected, assign sequential numbers

4. MCP Workflow Integration

Problem: Ensuring sequential execution of capture → OCR → mapping → clicking.

Solution:

- Used `@mcpagentapp.workflowtask` decorators with timeouts
- Implemented proper error handling between workflow steps
- Maintained state across workflow tasks using class attributes

Edge Cases Handled:

1. OCR Errors:

- Low confidence detections: Filtered out (<30% confidence)

- No text found: Falls back to mock data with warning
- Punctuation as text: Filtered out . , ; ! ?
- Multiple detections of same element: Confidence-based sorting

2. Coordinate Issues:

- High-resolution screens: Proper scaling using scalex/scaley
- Multiple displays: Uses primary display coordinates
- Element not found: Fuzzy matching with `difflib.getclosematches()`
- No coordinates: Error handling with "Element has no coordinates" message

3. User Interaction Issues:

- Voice command misunderstandings: Simple tool-based interface in Open-WebUI
- Target not found: Clear TTS feedback: "Could not find [target]"
- Network issues: Local OCR eliminates API dependencies

Testing Methodology:

- Basic Functionality: Capture → OCR → Click on common UI elements
- Accuracy Testing: Measure pixel distance between intended and actual clicks
- Failure Cases: Attempt to click non-existent elements
- Multiple Applications: Test across browser, file explorer, text editor
- Performance: Timing for complete workflow execution

Results:

- Success Rate: ~85% for clearly labeled UI elements
- Accuracy: Within 5-10 pixels for most elements
- Failure Cases: Handled gracefully with appropriate feedback
- Performance: ~3-5 seconds for complete capture-to-click workflow

Limitations & Future Improvements:

Current Limitations:

- OCR dependency: Requires clear, readable text labels

- Static UI only: Doesn't handle dynamic content or animations
- English text only: EasyOCR configured for English only
- Single element clicks: No support for drag-drop or multi-clicks

Proposed Improvements:

- Multi-language support: Configure EasyOCR for additional languages
- Visual element detection: Add icon/button shape detection alongside OCR
- Context awareness: Remember UI state between interactions
- Gesture support: Implement drag, scroll, right-click actions
- Error recovery: Automatic retry with adjusted parameters

Conclusion: The P4 system successfully demonstrates agentic computer interaction through a hybrid MCP architecture. Key achievements:

- Working voice-to-action pipeline from command to execution
- Accurate coordinate detection using EasyOCR for pixel-perfect positioning
- Robust error handling for OCR failures and missing elements
- Multi-application compatibility across different UI contexts
- Clear user feedback through TTS and structured JSON outputs

The system provides a foundation for more advanced UI automation tasks and demonstrates practical application of MCP workflows, OCR, and desktop automation integration.