OmniMCP: A Framework for Self-Generating UI Understanding Through Spatial-Temporal Synthesis

Richard Abrich
OpenAdapt.AI

March 2025

Abstract

We present OmniMCP, a novel framework that enables large language models to develop comprehensive UI understanding through the synthesis of spatial and temporal features. The framework combines fine-grained UI segmentation with process graphs derived from human demonstrations to construct rich contextual representations of interface states and interaction patterns. Our approach introduces a self-generating semantic layer that bridges the gap between raw UI elements and task-specific interaction strategies. Initial results demonstrate the framework's potential for robust, context-aware UI automation across diverse interface patterns.

1 Introduction

User interface automation remains a significant challenge in artificial intelligence, particularly in developing systems that can generalize across diverse interfaces and adapt to varying contexts. While recent advances in computer vision and natural language processing have improved UI element detection, existing approaches often lack the ability to synthesize spatial understanding with temporal interaction patterns.

This paper introduces OmniMCP, a framework that addresses these limitations through two key mechanisms:

- Real-time UI structure analysis via OmniParser
- Temporal pattern learning through process graph representations

2 Related Work

The challenge of UI automation has been approached from multiple angles in recent years. Screen parsing approaches have focused on hierarchical element

detection, while demonstration-based methods have emphasized pattern recognition. However, few approaches have attempted to synthesize both spatial and temporal understanding in a unified framework.

3 Methodology

3.1 Framework Overview

OmniMCP's architecture enables language models to generate semantic understanding by analyzing:

- UI element hierarchies and spatial relationships
- Historical demonstration patterns encoded in process graphs
- Contextual mappings between current states and successful interaction sequences

3.2 Process Graph Representation

We formalize UI automation sequences as directed graphs G(V, E) where:

- V represents concrete UI interactions
- E represents UI state transitions

This representation enables:

- Efficient encoding of demonstrated interaction patterns
- Natural language descriptions of states and actions
- Systematic validation of execution sequences

3.3 Spatial-Temporal Feature Synthesis

The core innovation of our approach lies in the dynamic synthesis of:

```
Spatial Features (via OmniParser):
- Element hierarchy and relationships
- Layout structure analysis
- Interactive element affordances

Temporal Features (via Process Graphs):
- Demonstration-derived interaction patterns
- State transition sequences
- Context-specific action selection
```

4 Implementation

We implement the core synthesis mechanism as follows:

```
def generate_understanding(
    current_ui: UIState,           # From OmniParser
    task_description: str,         # Current goal
    process_graphs: List[Graph]    # Demonstrated patterns
) -> ActionPlan:
    """Generate contextual UI understanding"""

    # Get spatial features
    ui_elements = omniparser.analyze(current_ui)
    spatial_context = describe_elements(ui_elements)

    # Get temporal features
    similar_demos = retrieve_demonstrations(
        task_description,
        process_graphs
    )
    temporal_context = describe_patterns(similar_demos)

    # Synthesize understanding
    prompt = f"""
    Current UI:
    {spatial_context}

    Relevant patterns:
    {temporal_context}

    Task goal:
    {task_description}

    Generate action plan considering both current
    UI state and demonstrated patterns.
    """

    return model.generate_plan(prompt)
```

5 Testing

Our testing methodology focuses on generative validation across diverse UI scenarios, enabling rapid iteration and refinement of the understanding synthesis process. While quantitative metrics are still being developed, qualitative results show promising capabilities in cross-platform adaptation.

6 Limitations and Future Work

Current limitations include:

- Need for more extensive validation across UI patterns
- Optimization of pattern recognition in process graphs
- Refinement of spatial-temporal feature synthesis

Future work will focus on:

- Development of comprehensive evaluation metrics
- Enhanced pattern recognition capabilities
- Expanded cross-platform validation
- Integration with broader LLM architectures

7 Conclusion

We present OmniMCP as a promising approach to self-generating UI understanding. While still in development, initial results suggest potential for robust, adaptable UI automation through the synthesis of spatial and temporal features. Continued work will focus on validation and refinement of the core mechanisms.

8 References

TODO