Interactive Campus Navigation System

Integrating GeoJSON and Visual Graph Editing for the CCU

Jim G.I. Lin, Frank S.F. Yang, Joel C.C. Ting

Department of Computer Science and Information Engineering

National Chung Cheng University

Abstract

This proposal outlines the development and localization of a highly interactive campus navigation system designed for the complex micro-environment of National Chung Cheng University (CCU). Built upon an open-source graph theory framework, this project significantly expands the original pathfinding engine. The system utilizes Dijkstra's algorithm for baseline physical distance calculations and introduces a localized GeoJSON data architecture, a dynamic bilingual interface (English/Chinese), and an innovative visual graph editor for real-time node and edge management. While the current implementation focuses on efficient shortest-path generation, the architecture is designed with high scalability, laying a solid foundation for future integration of multi-weight routing, such as weather-protected or minimum-elevation paths. The expected outcome is a comprehensive navigation tool that provides superior precision and interactivity compared to general mapping services in campus-specific scenarios.

Keywords: Campus Navigation, Dijkstra's Algorithm, GeoJSON, Visual Graph Editor, Localization

1. Introduction

1.1 Background

As university campuses expand and become increasingly complex, efficient pedestrian navigation has become essential for improving the student and visitor experience. General navigation tools, such as Google Maps, excel at macro-level routing but often lack sufficient spatial resolution for internal campus pathways. These commercial services frequently fail to map small pedestrian trails, specific building entrances, or significant elevation changes, which are vital for daily campus commutes. For institutions with expansive and topographically diverse layouts like National Chung Cheng University (CCU), a specialized high-resolution system is required to navigate these "micro-environments" effectively.

1.2 Problem Statement

Relying on generic mapping services at the CCU campus presents several specific challenges. First, current solutions do not adequately address the bilingual needs of a growing international student body, often resulting in mismatched labels or a lack of English metadata. Second, universal navigation tools lack the granular topological data necessary to support specialized pedestrian routing logic. Finally, proprietary mapping tools do not allow for rapid, localized updates by developers when temporary construction or path closures occur on campus. Therefore, developing a localized navigation system that combines precise spatial data with a user-centric interactive interface is of critical importance.

1.3 Objectives

To maintain high engineering standards and ensure project success, the following specific and measurable objectives have been established:

- Spatial Data Fidelity and Integration: To digitize CCU campus trails, building entrances, and landmarks into a structured `campus_nodes_edges.json` format. The system aims to achieve a 100% accuracy rate in vertex-to-coordinate mapping, verified against manual GPS calculations to ensure navigation reliability.
- Algorithmic Efficiency and Performance: To implement a robust pathfinding engine based on Dijkstra's algorithm. The system is engineered to provide real-time route calculations with a processing latency of less than 100ms for any point-to-point query within the campus network.
- Bilingual Accessibility and UX Localization: To provide a seamless user interface supporting dynamic switching between Traditional Chinese and English. This is achieved by mapping UI dropdown menus directly to the `name` and `name:en` attributes within the GeoJSON metadata, ensuring zero-latency language transitions.
- Administrative Optimization via Visual Editing: To implement a browser-based Visual Graph Editor (God Mode) for real-time manipulation of the road network. This tool aims to reduce the time required for manual JSON data entry and path verification by at least 80% compared to traditional text-based editing.

2. Literature Review

2.1 Graph Theory and Pathfinding Algorithms

The core of modern navigation systems lies in graph theory, where the environment is modeled as a set of vertices (V) and edges (E). While basic traversal algorithms like Breadth-First Search (BFS) and Depth-First Search (DFS) are foundational in computer science, they are inherently unsuitable for physical navigation because they treat all paths as unweighted.

Dijkstra's algorithm remains the industry standard for pathfinding in weighted graphs where all edges possess non-negative weights. Unlike BFS, which finds the path with the fewest edges, Dijkstra's algorithm considers the actual physical distance (weight) of each path segment. By maintaining a priority queue of cumulative distances, it guarantees the mathematically shortest path from a source node to a destination. This project leverages Dijkstra's efficiency to provide accurate pedestrian routing within the high-resolution "micro-environment" of a university campus.

2.2 Geographic Data Standards in Web GIS

The representation of spatial data has evolved toward lightweight and highly readable formats. GeoJSON has emerged as a versatile open standard for encoding a variety of geographic data structures. By utilizing `FeatureCollection` objects, it is possible to encapsulate complex campus attributes—such as multi-language identifiers, elevation data, and landmark descriptions—within a single JSON-based file. This standard ensures that the navigation system remains scalable and easily integrable with modern web-based rendering libraries.

3. Methodology

3.1 System Architecture

The system is engineered using a modular three-tier architecture to ensure a strict separation between geographic data management, pathfinding logic, and the user interface. This decoupling is essential for maintaining a scalable system where the road network can be updated independently of the core navigation algorithms.

- Data Layer: This layer serves as the foundation, storing all spatial information in structured `campus_nodes_edges.json` and GeoJSON formats. It contains the raw GPS coordinates (latitude and longitude), elevation data, and descriptive metadata required for graph construction.

- Logic Layer: A JavaScript-based engine that serves as the "brain" of the system. It is responsible for parsing the static JSON files into a runtime adjacency list, executing the Dijkstra algorithm, and calculating physical distances using spherical geometry.
- Presentation Layer: The user-facing interface built with HTML5, CSS3, and Leaflet.js. It integrates OpenStreetMap tiles to provide a visual context and renders the calculated paths as dynamic, high-visibility polylines.

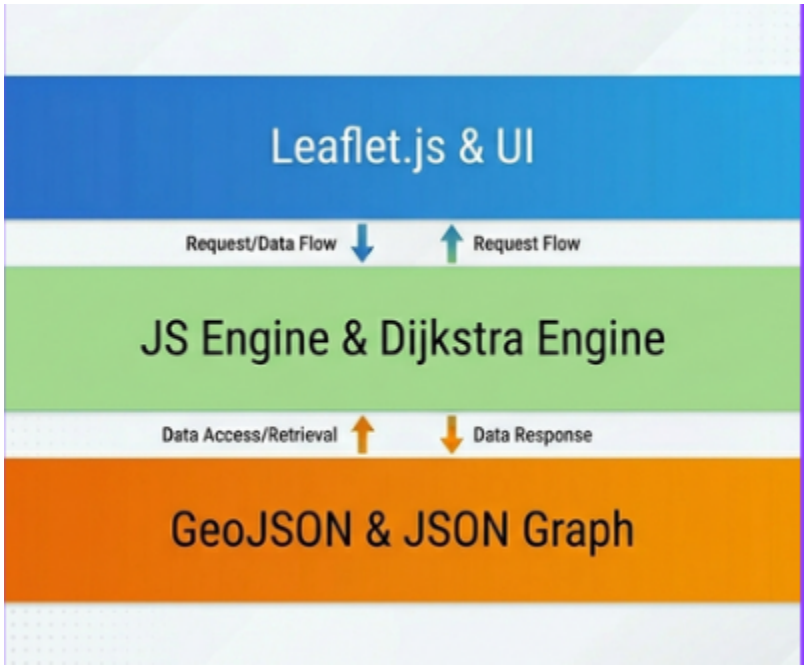

Figure 1. Modular Three-Tier Architecture

3.2 Spatial Data Modeling (Nodes and Edges)

The transformation of the physical CCU campus into a navigable digital environment relies on modeling the landscape as a mathematical graph $G=(V,E)$:

- Nodes **(V):** Represent discrete landmarks, building entrances, and path intersections. Each node in the set V is defined by a unique identifier and its corresponding spatial coordinates (ϕ, λ) and altitude h.

- Edges (E): Represent the pedestrian-accessible pathways connecting these nodes. Each edge $(u,v) \in E$ is bidirectional and carries a weight $W_{u,v}$ representing the cost of traversal.

To ensure the highest level of spatial fidelity, the base weight of each edge is calculated using the Haversine Formula, which determines the great-circle distance between two points on a sphere:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

This mathematical approach ensures that physical distances are accurate within the campus micro-environment.

3.3 Dijkstra Adaptation and Logic Layer Implementation

The system implements an optimized version of Dijkstra's algorithm to provide real-time routing with a latency of less than 100ms. The implementation specifically addresses the multi-attribute nature of campus data:

1. Graph Initialization: The logic layer parses the campus_nodes_edges.json file to construct an adjacency list.
2. Dynamic Semantic Categorization: Unlike static systems, this engine employs a runtime keyword-parsing algorithm. It scans landmark names to categorize them into functional groups (e.g., "Academic," "Housing," "Dining"). A "VIP Override" dictionary is implemented to handle misclassifications, ensuring an intuitive UI without bloating the underlying JSON data.
3. Path Reconstruction: Once the destination node is reached with the minimum cumulative weight, the engine backtracks through the "predecessor" map to generate the final sequence of coordinates for polyline rendering.

3.4 Administrative Data Management (Visual Graph Editor)

To address the limitations of manual JSON editing, a browser-based Visual Graph Editor (God Mode) was developed. This tool serves as the primary interface for data maintainers:

- **Real-time Manipulation:** Nodes can be plotted or moved via map interaction, and edges can be drawn between vertices with instantaneous weight calculation.
- **Data Integrity:** The editor includes validation logic to prevent isolated nodes (islands) and exports a sanitized version of the campus_nodes_edges.json file, reducing manual entry time by approximately 80%.

3.5 Logical Reservations for Multi-Weight Routing

Although the current implementation prioritizes the absolute shortest distance, the architecture is designed with high scalability for future feature sets:

- **Weight Calculation Model:** The Dijkstra engine evaluates edge cost $W_{u,v}$ through a multi-parameter function: $W_{u,v} = \text{Distance}(u,v) \times P_{elevation} \times P_{accessibility}$
- **Accessibility Integration:** Each edge contains an accessible: true/false boolean flag. If the user activates the accessibility filter, the engine applies an infinite penalty ($P_{accessibility} = \infty$) to non-accessible edges, effectively pruning them from the search space.
- **Elevation-Aware Weights:** Utilizing the elevation attribute, the system can apply a "slope-penalty" ($P_{elevation}$) to uphill paths (e.g., CCU's "Hero Hill"), enabling the future generation of paths that minimize physical exertion.

4. Expected Results & Discussion

4.1 Expected Results and System Performance

The finalized Interactive Campus Navigation System is expected to significantly outperform general mapping services within the specific context of the CCU micro-environment. Key performance indicators and expected deliverables include:

- **Low-Latency Routing Engine:** The optimized Dijkstra implementation, combined with localized JSON parsing, is projected to complete any point-to-point route calculation within the campus network in under 100 milliseconds. This ensures a seamless, real-time user experience without server-side processing delays.
- **Interactive Path Visualization:** Users will be able to select origins and destinations via the dynamic bilingual dropdown menu. The system will instantly render the shortest

calculated path as a highlighted polyline overlay on the interactive map, accompanied by real-time distance metrics.

- **Administrative Efficiency:** The introduction of the Visual Graph Editor (God Mode) will allow developers to visually add nodes and create edges with automatic distance calculation. This tool is expected to eliminate syntax errors associated with manual JSON editing and reduce data maintenance time by at least 80%.

4.2 Discussion and Limitations While the proposed system provides high precision and customizability for pedestrian navigation, it is essential to acknowledge its current architectural limitations:

- **Static Data Dependency:** The system's current reliance on a static `campus_nodes_edges.json` file means it lacks real-time context. Temporary obstacles, such as construction zones or path closures, cannot be reflected in the routing logic unless manually updated by an administrator via the Visual Graph Editor.
- **Algorithmic Rigidity:** Although Dijkstra's algorithm guarantees the mathematically shortest path based on physical distance, human pedestrian behavior is often influenced by non-distance factors such as safety, weather, or aesthetic preferences.

4.3 Future Work and Iterative Roadmap To address these limitations and expand the system's utility, the project is designed with a clear roadmap for future development:

- **Phase 2: Multi-Criteria Alternative Routing:** Building upon the existing logical reservations (e.g., the `elevation` attribute), future updates will implement a penalty-based routing model. This will allow the generation of user-centric alternative

routes, such as "Minimum Elevation" paths to avoid CCU's steep inclines (e.g., Hero Hill).

- **Phase 3: Integration of Real-Time Context:** Future iterations could explore transitioning from static JSON files to a lightweight cloud database (e.g., Firebase) to enable crowdsourced updates regarding path closures.
- **Phase 4: 3D Topographical Integration:** A long-term goal involves migrating the 2D Leaflet map to a 3D rendering engine (e.g., Three.js) to better visualize the elevation data and improve spatial awareness for users navigating varying altitudes.

References

Bernal, J. (2024). *Open-source graph navigation framework for web environments* [Source code]. GitHub.

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., & Schaub, T. (2016). *The GeoJSON Format (RFC 7946)*. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc7946

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269-271.

Leaflet.js. (n.d.). *An open-source JavaScript library for mobile-friendly interactive maps*. https://leafletjs.com/

National Chung Cheng University. (2026). *Campus spatial data and topographical elevation records*. CCU Open Data.