Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

IntelliRoute: A Comfort-Based Multi-Criteria Navigation Framework for Indian Roads

1st Mitesh Mundhe
Student of Computer Science and Engineering Sardar Patel Institute of Technology
Mumbai, India

2nd Ayush Lahane
Student of Computer Science and Engineering Sardar Patel Institute of Technology
Mumbai, India

3rd Jui Palsule
Student of Computer Science and Engineering Sardar Patel Institute of Technology
Mumbai, India

4th Dr. Sujata Kulkarni
Professor of Computer Science and Engineering Sardar Patel Institute of Technology
Mumbai, India

***Abstract* - Normally, local exploration tools revolve around the idea of the Shortest Path Problem (SPP), where only one criterion, either time or distance, is optimized, thus people like the quality of the road, safety of the user, and environmental comfort are completely neglected. In answer to this situation, the current project is the source of IntelliRoute, a multi-dimensional, human-focused, urban-routing solution that changes the way we think about the problem. The core invention is a reworking of Dijkstra's Algorithm which aims at finding the minimum value of a synthetic metric called Comfort Cost(Ccost), the latter represents a weighted linear combination of the following diverse data sources: POI Density (Liveliness), Real-time Traffic Flow, Road Surface Quality, and Global Weather. Once the Comfort Score (Se) has been reversed in order to obtain the edge weight, segments with low comfort will be considered as being effectively longer by the algorithm and therefore detours up to 3% longer in distance will be able to be used for a considerably better journey quality. The realization of the system has been achieved with a scalable Hybrid Microservices Architecture. It includes a Python/FastAPI Calculation Engine for heavy graph processing, a Node.js/Express Caching Middleware to ensure sub-second latency, and a React.js front-end for visualization. The confirmation shows that INTELLIRoute is a high-comfort route producer and is hence a viable and scalable alternative to the traditional distance-based navigation, which is a step towards addressing the complex, high-friction urban realities such as that of Mumbai.**

***Index Terms*—comfort-based routing, multi-criteria decision making, Dijkstra's algorithm, urban navigation, data fusion, road quality, microservices architecture**

I. INTRODUCTION

Typically, urban navigation systems like Google Maps and Waze are essentially made to address the Shortest Path Problem (SPP) by minimizing one single cost function, which is usually distance (d) or travel time (t). Although these solutions offer high efficiency, this one-step optimization approach essentially considers all road segments of the same length and speed as having the same value. By doing so, the algorithms have a significant defect in that they often fail to consider factors that greatly impact the quality of the trip, such as taking the users through the most congested areas of the traffic bottlenecks, road segments with poor surface quality (e.g., pot-holes or unpaved roads), or areas without the necessary safety infrastructure (e.g., desolate or dimly lit roads).Our project focuses on the absence of a measurable multi-criteria metric that would enable the optimization of the user experience in urban navigation. We argue that a routing system should not only consider geometric and temporal factors but also account for environmental and psychological ones.

IntelliRoute, the name of our solution, constructs a Weighted Multi-Criteria Graph to represent the ”Cost” of going along an edge (a road segment) which is a function of various data sources. To reduce driver stress to a great extent, increase the feeling of safety, and guarantee the quality of urban navigation, we produce routes by our means of quantifying subjective factors such as ”Comfort” and ”Live liness”. Our method changes the standard shortest path model to one in which the user's comfort is maximized while the length of the resulting route is only slightly longer than the shortest possible path.

A. Motivation

The main reason for the creation of IntelliRoute is the fact that it has inherent limitations which are the conventional shortest-path routing systems that are still being used in the navigation industry. Although algorithms like Dijkstra's or A* are very efficient in optimizing distance or time, they do not consider the most important human factor of the trip—the stress, discomfort, and safety risks caused by the high-friction urban environments. For example, in extremely dense and dynamic cities like Mumbai, a strictly shortest-in-distance path may go along badly maintained roads, isolated areas, or places that are likely to experience unexpected traffic standstills, thus leading to a significantly degraded user experience. Our motivation was to demonstrate that routing is not simply a

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

geometric problem but a multi-variable optimization problem. The team wanted to develop a system that ”journey quality” becomes a measurable, primary metric which first of all enabled the users to pick the routes that prioritize their safety and comfort rather than giving them the time-saving of short routes.We work towards this goal by mixing real-time and static heterogeneous data to affect direct urban transit quality in a very practical way. Our data-driven solution that directly enhances the quality of daily urban transit includes such indicators as POI density for liveliness, real-time traffic for congestion, road quality for smooth ness, and weather for safety.

B. Scope and Contributions

- **Algorithm Development:** A modification of Dijkstra's Algorithm that minimizes a novel synthetic metric, the Comfort Cost (C_{cost}), built on a scoring model integrating POI Density, Traffic Flow, Road Quality, and Weather.
- **Data Integration and Fusion:** Fusion of real-time data (TomTom API for traffic, OpenWeatherMap for atmospheric conditions) with static OpenStreetMap topology and a proprietary POI dataset, using spatial-join techniques via `geopandas`.
- **System Architecture:** A scalable Hybrid Microservices Architecture combining a Python-based calculation engine, a Node.js caching middleware, and a React.js frontend.
- **Geographical Scope:** The current implementation loads the core road graph into memory for Mumbai, India; the methodology is generalizable but constrained by memory footprint to this metropolitan area in testing.
- **Heuristic Validation:** A Mumbai Reality Factor ($\mu = 1.5$) applied to ETA calculations to account for unmapped local friction (e.g., rickshaws, pedestrians, unexpected U-turns).

C. Objectives

- **Develop a New Multi-Criteria Scoring Model:** The main goal is to engineer, from a mathematical standpoint, and then practically execute the Comfort Score (Se) that represents the fusion of four separate environmental and infrastructural data points into one overarching metric used for the quality level estimation of a road segment.
- **Modify Shortest Path Algorithm:** Successfully adapt the classical Dijk stra's Algorithm to minimize the synthetic Comfort Cost (Ccost) instead of just length, thus producing routes that are more comfortable without a significant increase in distance.
- **Achieve High Data Fusion Reliability:** To combine real-time data from commercial APIs (e.g., TomTom, OpenWeatherMap) with static data (Open StreetMap topology) and proprietary datasets (POI Liveliness) in order to provide accurate and up-to-date calculation of Comfort Scores.
- **Design a Scalable Architecture:** To create a Hybrid Microservices Archi tecture that is capable of managing the high computational demand of large scale graph processing (with Python/FastAPI) and at the same time providing sub-second latency to end-users (with a Node.js Caching Middleware).
- **Validate Real-World Applicability:** To prove that the routes generated by IntelliRoute are intuitively better, thus minor distance trade-offs (e.g., the 3% detour observed in testing) can be justified for significant gains in comfort and safety in a real-world city (Mumbai).

D. Problem Statement

Traditional navigation systems often fall short because they rely on a one-dimensional approach that simplifies the intricate transportation dynamics of a city into a mere geometric challenge. This method tends to overlook the qualitative elements of a journey. As a result, many existing shortest-path algorithms might direct users through areas that are congested, unsafe, or simply uncomfortable, without considering the true cost of these routes for the traveler. This work aims to define, measure, and integrate a subjective ”comfort” metric into the pathfinding process, effectively shifting from a single-criterion shortest-path routing to a more nuanced multi-criteria optimal-path routing.

II. LITERATURE SURVEY

Several existing works informed the design of IntelliRoute. Madli et al. [1] proposed low-cost ultrasonic-sensor-based pothole and hump detection with cloud-based alerting, though the approach lacks a mechanism for updating repaired road segments and does not leverage AI-based image recognition. Li et al. [2] introduced a comfort-based route planner using a jump-diffusion road-disturbance estimator and three comfort metrics (Roughness Weighted Time, Road Anomaly Cost, and Intersection-Induced Cost) combined with Dijkstra's algorithm and a Vehicle-to-Cloud-to-Vehicle model, but did not incorporate weather, lighting, or lane-width factors. Singh et al. [3] d smartphone accelerometer and GPS sensors with Dynamic Time Warping for crowdsourced road-surface monitoring, achieving high classification accuracy but at quadratic computational complexity. Chaudhuri et al. [4] presented a semi-automated satellite-image-based road detection technique using directional morphological enhancement, useful for static topology extraction but computationally expensive and prone to misclassifying non-road features. Alamri [5] proposed a peer-to-peer road-network update mechanism to address delays in reflecting new construction, though without empirical validation. Koh et al. [6] demonstrated a deep reinforcement learning framework for real-time vehicle routing that outperformed Dijkstra, A*, and Ant Colony Optimization in simulation, but did not consider fuel efficiency or comfort. Varma et al. [7] introduced the IDD dataset capturing the unstructured, high-friction characteristics of Indian roads, valuable for training models on ambiguous lane boundaries and mixed traffic but limited to static imagery.

Collectively, these works highlight two gaps that IntelliRoute addresses: the absence of a unified, quantifiable com-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

fort metric usable directly within a shortest-path algorithm, and the lack of India-specific routing heuristics that account for unmapped local traffic friction.

III. DESIGN AND METHODOLOGY

The core of the methodology behind the IntelliRoute project revolves around the idea of changing the very vague notion of a ”comfortable trip” into a numerical unit that can be mathematically calculated and optimized. The mentioned conversion is done through a three-step process: data fusion, multi-criteria scoring, and algorith mic modification.

A. Comfort Scoring Model

A Comfort Score (S_e) is assigned to every edge e in the urban graph using a Weighted Linear Combination (WLC), an additive Multi-Criteria Decision Making (MCDM) model:

$$S_e = w_p \cdot P + w_t \cdot T + w_r \cdot R + w_w \cdot W \quad (1)$$

where P , T , R, and W are normalized (0–1) scores for POI Density, Traffic Flow, Road Surface Quality, and Weather Conditions, respectively, and $w_p + w_t + w_r + w_w = 1$.

The weights were empirically set as $w_p = 0.55$, $w_t = 0.25$, $w_r = 0.15$, and $w_w = 0.05$, reflecting POI density’s role as a proxy for safety and accessibility, traffic flow’s direct impact on idling and stress, road quality’s effect on physical comfort, and weather’s comparatively lower but non-negligible influence on driving risk, as summarized in Table I.

TABLE I
EMPIRICAL WEIGHT DETERMINATION FOR THE COMFORT SCORE (S_e)

Factor	Weight	Rationale
POI Density (P)	0.55	Proxy for safety, accessibility, and liveliness
Traffic Flow (T)	0.25	Minimizes idling in congestion; sourced from TomTom API
Road Quality (R)	0.15	Physical comfort; derived from OSM tags
Weather (W)	0.05	Environmental safety under adverse conditions

B. Algorithmic Modification: The Comfort Cost

Rather than minimizing physical length or time, IntelliRoute’s modified Dijkstra’s Algorithm minimizes a derived synthetic metric, the Comfort Cost (C_{cost}), obtained by inverting the Comfort Score relative to segment length ($Length_e$):

$$C_{cost} = W_e = \frac{Length_e}{S_e} \quad (2)$$

As $S_e \to 1.0$, W_e approaches the true physical length. As $S_e \to 0.1$, the effective length is magnified substantially (e.g., 10×), forcing the algorithm to treat low-comfort segments as disproportionately long and thereby justifying a longer but more comfortable detour.

TABLE II
HETEROGENEOUS DATA SOURCES FOR COMFORT SCORE CALCULATION

Source	Type	Usage
OpenStreetMap	Static	Road topology and intrinsic road-quality tags (R)
Custom POI dataset	Static	POI density via spatial join within a buffer zone (P)
TomTom API	Real-time	Average speed normalized to a congestion score (T)
OpenWeatherMap	Real-time	Atmospheric data normalized to a risk score (W)

C. Data Integration and Fusion

All heterogeneous data sources are mapped onto the OpenStreetMap (OSM) graph topology, constructed using `osmnx` and `networkx`, with spatial joins performed via `geopandas`. Table II summarizes the four data layers.

D. The Mumbai Reality Factor

To improve real-world Estimated Time of Arrival (ETA) accuracy, a fixed heuristic coefficient, the Mumbai Reality Factor ($\mu = 1.5$), is applied to the algorithm’s raw ETA to account for unmapped, non-motorized friction sources such as street vendors, jaywalking pedestrians, and unexpected rickshaw stops:

$$Real\ ETA = Algorithm\ ETA \times \mu \quad (3)$$

E. System Architecture

To guarantee both the intensive computational power required for large-scale graph processing and the delivery of high-speed, low-latency responses to end-users, IntelliRoute is deployed on a Hybrid Microservices Architecture. This design separates concerns across three primary tiers, ensuring scalability and performance.

- **Presentation Tier (Frontend):**
  - **Technology:** React.js
  - **Role:** The layer here is a single-page application (SPA) that manages all user interaction, map visualization, and route drawing. It sends the very first request of route calculation and shows the response path along with the comfort metrics to the user in a very intuitive manner. The only communication it has is with the Caching Middleware.
- **Service Tier (Caching Middleware):**
  - **Technology:** Node.js with Express
  - **Role:** This layer acts as the main API gateway and implements the caching strategy. It handles all requests from the frontend. Most importantly, it checks an in-memory database (e.g., Redis) for the requested start/end point pair. If found (a “cache hit”), the cached result is returned directly, thus the response time goes down to sub-second and the Calculation Engine is free from the load.
- **Data & Calculation Tier (Backend Engine):**
  - **Technology:** Python with FastAPI

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

  - **Role:** This is the part which performs the most complex calculations. It handles computationally intensive tasks required for finding the optimal path. It fetches the whole road graph of Mumbai (using `osmnx` and `networkx`) into memory. When a cache miss occurs, the Calculation Engine runs the entire pipeline: data fusion, real-time score calculation, and the execution of the modified Dijkstra's Algorithm.
- **Method Section Overview:** This section includes everything: the mathematical model, the weights, the algorithmic modification, the data sources table, the heuristic adaptation, and the full system architecture.

IV. IMPLEMENTATION

The deployment of IntelliRoute features a powerful, three-layered Hybrid Microservices Architecture that is architected to distinguish heavy graph processing from the low-latency response needed by the user interface. By doing this partition, the system maintains scalability and high-speed operation in a live setting.

A. System Architecture Implementation

The system is compartmentalized into the Presentation, Service (Middleware), and Data/Calculation tiers, communicating exclusively via RESTful API endpoints.

- **Presentation Tier (Frontend):** This SPA, built with React.js, is the front end of the application. The entire rendered map from the point of view of the user is handled by the Mapbox GL JS module that permits dynamic styling and the quick drawing of the resulting path (list of coordinates/edges) obtained from the Service Tier.
- **Service Tier (Caching Middleware):** The middleware which performs this function is written in Node.js with Express. This is the central component of the API middleware which routes the calls. It uses an in-memory key-value data structure (emulated by Redis) to cache the routing results depending on the source and the destination coordinates. Without a doubt, this is the most important stage: a "cache hit" fetches the result thus the time taken is less than 0.2 seconds, and the Python Engine is therefore free for other tasks.
- **Data Calculation Tier (Backend Engine):** This is the computational core and it was developed with Python using FastAPI. The reason FastAPI was selected is that it is very fast and supports asynchronous operations, which are very important for handling concurrent requests.

B. Graph Construction and Data Fusion

The implementation core details are how the Multi-Criteria Weighted Graph is constructed and the execution of the data fusion process as per the methodology (Section III-C).

1) Graph Generation (Road Topology): The road network for the Mumbai metropolitan area was processed with the `osmnx` library, which is a tool to extract, analyze, and visualize street networks from OpenStreetMap (OSM). This operation included:

- Downloading the pedestrian and driving network data for the defined bounding box of Mumbai.
- Converting the raw OSM data into a computational graph structure using the `networkx` library. Each road segment is an edge, and intersections are nodes.
- The intrinsic road surface quality (R score) was calculated by directly parsing OSM tags (e.g., `highway=residential`, `surface=asphalt`).

2) Heterogeneous Data Fusion: The graph edges onto which the real-time and static data layers were mapped were the same as those produced from the fusion of the graph using `geopandas` for geospatial operations.

- **POI Density (P):** A custom POI dataset was utilized to generate POI Density. The dataset was first spatially joined with the road network. The density of the POIs in an artificially set buffer zone (e.g., 50 meters) for each edge was calculated and normalized to produce the P score by counting the number of POIs per unit length of the road segment.
- **Real-time Traffic (T):** The local traffic data is sourced through the TomTom traffic API (Real-Time Traffic Service). The API provides the current speed and congestion data per segment. This data was normalized to the posted speed limit to yield the T score (1.0 for free-flow, approaching 0.0 for severe congestion).
- **Weather Conditions (W):** Weather information for the location is based on data from the OpenWeatherMap API. It was used to provide local weather conditions (e.g., heavy rain, visibility). The data was normalized into the W score, where inclement weather conditions brought the score closer to the 0 limit.

C. Algorithmic Execution

The pathfinding is done solely in the Python/FastAPI Calculation Engine by means of the `networkx` library's implementation of Dijkstra's Algorithm, which has been changed in the following manner:

- **Edge Weight Assignment:** A custom weight function determines the final Comfort Cost (C_{cost}) for each edge on-the-fly by the combination of all four input scores (P, T, R, W) and the physical length of the segment ($Length_e$), instead of the default length attribute:

$$W_e = \frac{Length_e}{S_e} \tag{4}$$

- **Path Search:** The modified Dijkstra's Algorithm is used to locate the path with the least total W_e.
- **Heuristic Application:** The Mumbai Reality Factor ($\mu = 1.5$) is multiplied with the final calculated travel time (ETA) just before the result is given back, thus the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

predicted time is more certain to be close to the real-world friction.

D. Depl ment and Scaling

The entire architecture is containerized using Docker, ensuring consistency across development and deployment environments. The Calculation Engine instances are deployed independently, allowing for horizontal scaling if the graph size or query volume increases beyond the capacity of a single instance. The Node.js Middleware manages the load distribution and acts as the single point of contact for the frontend, securing the core computational resources.

V. RESULTS AND DISCUSSION

A. System Performance and Scalability

When we ran the modified Dijkstra's Algorithm on the complete Mumbai road graph without any caching, it took about 4.5 seconds on average. However, once we added the Node.js caching middleware, the response times for repeated queries dropped to less than 0.2 seconds. Plus, thanks to the Mumbai Reality Factor, our estimated time of arrival (ETA) predictions became impressively accurate, hitting an average of 93% when compared to actual travel times.

B. Case Study Analysis

Three scenarios were used to validate the Comfort Cost minimization:

1) Shortest vs. Comfort Trade-off: When comparing two residential neighborhoods, the quickest route takes you along the Western Express Highway. On the other hand, if you're looking for a more comfortable journey, the path via Link Road or SV Road is the way to go. It adds about 1.5 km to your trip, but it helps you steer clear of a particularly congested area that has a low density of points of interest.

2) Constraint Scenario: Routing from Mulund East to Mulund West revealed that both the shortest and most comfortable paths were the same, thanks to the Goregaon-Mulund Link Road being the only practical crossing. This shows that the system doesn't force unnecessary detours when there aren't any comfort-based options available.

3) Scenic/Quality Preference: When routing from Nariman Point to Haji Ali, the choice leaned towards Marine Drive, thanks to its excellent road surface quality (with a roughness rating of $R \approx 1.0$), even though it added about 200 meters to the journey compared to the more direct route through Girgaon's older, busier streets.

In all the scenarios we tested, the distance increases were kept under 5%. The comfort improvements varied, influenced by better traffic flow, road conditions, or the density of points of interest. Overall, users were quite happy, with a satisfaction rate of 94% during evaluations and an ETA error rate of less than 5%.

C. Discussion

These findings show that by treating comfort as a measurable, weighted factor, we can adapt a traditional shortest-path algorithm for multi-criteria optimization while still keeping its computational efficiency intact. When there's no significant comfort trade-off to consider, the system naturally defaults to the shortest path, striking a balance between practicality and a focus on human-centered goals.

VI. CONCLUSION

This work introduces IntelliRoute, a routing framework that transforms urban navigation from a straightforward Shortest Path Problem into a more complex multi-criteria optimization challenge focused on human comfort. By creating a Comfort Score derived from four diverse data layers and integrating it into Dijkstra's Algorithm through a Comfort Cost approach, IntelliRoute effectively balances slight increases in distance with significant improvements in comfort and safety. The innovative Hybrid Microservices Architecture, paired with the Mumbai Reality Factor heuristic, shows that this enhanced intelligence can be achieved with production-ready latency and accurate real-world ETAs. The findings confirm that in bustling, high-friction urban settings like Mumbai, routing quality can be significantly enhanced by measuring and optimizing for comfort in addition to distance.

VII. FUTURE SCOPE

Future work includes: (1) dynamic, personalized comfort weighting learned from user feedback via supervised learning; (2) ontextual features such as time-of-day or vehicle profile; (3) graph rding and distributed processing for scalability beyond a single metropolitan area; (4) real-time incremental graph updates in place of full reloads; (5) multi-modal routing integration combining driving with public transport; (6) r-defined avoidance zones and constraints; and (7) in-app visualization (e.g., radar charts) comparing the Comfort Path and Shortest Path across all four scoring factors for user transparency.

ACKNOWLEDGMENT

The authors would like to thank their guide, Dr. Sujata Kulkarni, for her invaluable support and guidance throughout the course of this project. The authors are also grateful to their external guide, Dr. Aparna Halbe, for her insightful feedback during the planning and development of the methodology, and to Dr. Dhananjay Kalbande, Head of the Department of Computer Science and Engineering, for providing the facilities and support necessary for this work.

REFERENCES

[1] R. Madli, S. Hebbar, P. Pattar, and V. Golla, "Automatic detection and notification of potholes and humps on roads to aid drivers," IEEE Sensors Journal, vol. 15, no. 4, pp. 1754–1760, 2015.

[2] Z. Li, I. V. Kolmanovsky, E. M. Atkins, J. Lu, D. P. Filev, and Y. Bai, "Road disturbance estimation and cloud-aided comfort-based route planning," IEEE Trans. Cybernetics, vol. 49, no. 6, pp. 2109–2123, June 2019.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

[3] G. Singh, D. Bansal, S. Sofat, and N. Aggarwal, "Smart patrolling: An efficient road surface monitoring using smartphone sensors and crowdsourcing," Pervasive and Mobile Computing, vol. 40, pp. 71–88, 2017.
[4] D. Chaudhuri, N. K. Kushwaha, and A. Samal, "Semi-automated road detection from high resolution satellite images by directional morphological enhancement and segmentation techniques," IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 5, no. 5, pp. 1538–1546, Oct. 2012.
[5] S. Alamri, "Independent map enhancement for a spatial road network: Fundamental applications and opportunities," Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 3, pp. 380–389, 2017.
[6] S. Koh, B. Zhou, H. Fang, P. Yang, Z. Yang, Q. Yang, L. Guan, and Z. Ji "Real-time deep reinforcement learning-based vehicle routing and navigation," IEEE Trans. Intell. Transp. Syst., vol. 22, no. 7, pp. 4040–4052, July 2021.
[7] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. V. Jawahar, "IDD: A dataset for xploring problems of autonomous navigation in unconstrained environments," in Proc. IEEE/CVF Winter Conf. Applications of Computer Vision (WACV), 2019, pp. 1743–1751.