A Dynamic Multi-Objective Cost Function for Personalized and Accessible Urban Pathfinding

Aayam Bajaj
Information Technology
Mumbai University
Mumbai, India
aayambajaj4@gmail.com

Atharva Dhokte
Information Technology
Mumbai University
Mumbai, India
atharva.a.d2004@gmail.com

Yuvrajsing Bahure
Information Technology
Mumbai University
Mumbai, India
bahureyuvi@gmail.com

***Abstract*:** Standard urban navigation systems often fail to meet the needs of individuals with mobility impairments. These systems focus on optimizing the shortest path, ignoring critical accessibility factors like surface quality, dynamic real-time obstacles, and sidewalk slopes, thereby creating an informational gap for users. In order to address these shortcomings, this paper introduces a pathfinding framework, by enhancing the A* search algorithm using a dynamic, multi-objective cost function. With calculation of the traversal cost for every path segment, our methodology provides personalised and context-aware routing. The derivation of this cost happens through the integration of multiple layers of data, including: (1) Personalized used profiles, (2) Community-sourced data on obstructions and temporary obstacles, (3) Machine learning model for predictive risk assessment, and (4) Live weather updates. This results in generation of routes that consider user safety and comfort. instead of just distance. Hence this system offers a reliable and accessible navigation experience.

Keywords: Pathfinding, A* Algorithm, Accessibility, Multi-Objective Optimization, Urban Navigation.

I. INTRODUCTION

With the growing population and increase of urban environment spaces, effective navigation is important for social connectivity and freedom of mobility. However, this basic navigation becomes a challenge for those with limited mobility. This includes the surface conditions, construction blockages, slope of sidewalks, and presence of curb ramps, etc. These barriers pose health risks and can also lead to detachment from the community life as well as reduced participation.

Current mapping services provide efficient routing for general pedestrians. Although, they fall short in meeting the mobility needs of users. This is because they only focus on a single objective, like finding the shortest path. This leaves out metrics like comfort, accessibility, and safety. There is data on accessibility, but it is variegated across parts of datasets, which creates a gap for the users who seek a dynamic nature of information.

In order to overcome these challenges, this paper introduces a framework which extends the A* algorithm with a dynamic, multi-objected cost function. We intend to provide personalized routing by integrating multiple layers of information into a cost metric. Our objective is to formulate a cost function that calculates path based on four key aspects

1.Personalized user-profile with mobility constraints and user preferences.
2.Real-time reporting and community sourced data on temporary obstacles.
3.Risk assessment based on observed patterns using machine learning.
4.Weather-condition data that may affect path metrics.

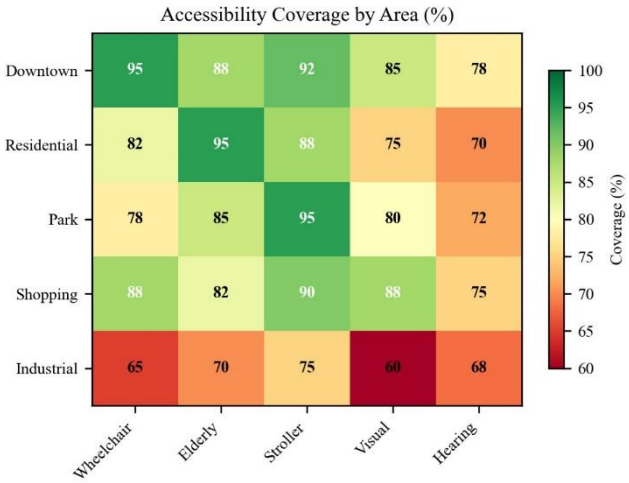

Fig 1. Matrix representing conditions

II. SYSTEM ARCHITECTURE

The system's backend operates as a computational pipeline designed to transform raw geographic data into a personalized, safe, and accessible routing service. The workflow consists of five stages, from initial data processing to the final path computation.

Step 1: Graph Construction from OpenStreetMap (OSM)
The foundation of our system is a routable graph built from OSM data. This fall under data pre-processing.
The first step is uploading the raw OSM data for a particular region in the procedure. Only "ways" and "nodes" that are relevant to pedestrians designated by tags such as highway=footway and sidewalk='yes', are chosen by the system after filtering this data. After that, a mathematical graph is produced, in which the nodes stand for intersections and the edges for the segments of paths that connect them. Each edge is initially stored with its static attributes from OSM, such as its physical length (L(e)) and surface type.

Step 2: Route Request Initialization
This stage begins when a user requests a route.

The backend receives the user's request, which contains three key pieces of information: a **start location**, an **end location**, and a **mobility profile** (containing the necessary metric values like maximum slope, disliked surfaces, etc.). The system then loads the relevant section of the pre-processed graph into the memory to begin dynamic analysis.

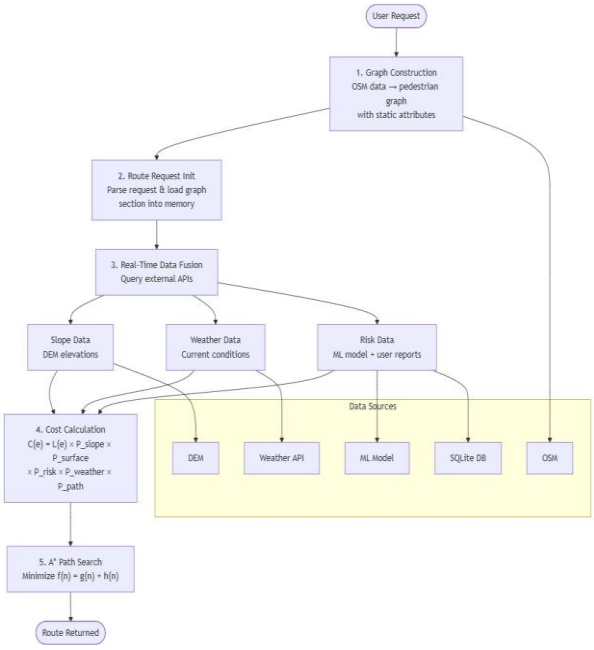

Fig2. System Architecture

Step 3: Real-Time Data Fusion
With the base graph loaded, the system enhances it with live, dynamic data.

- The system gathers real-time information from multiple sources, for each edge in the graph that is relevant to the potential route
- **Slope Data**: for an accurate slope calculation, It queries a **Digital Elevation Model (DEM)** service to get the elevation of the start and end nodes of the edge, allowing.
- **Weather Data**: An API call is made to a live **weather service** to determine current conditions (e.g., rain), which make certain surfaces more hazardous.
- **Risk Data**: It and checks our **SQLite database** for any user-reported temporary blockages and then queries our internal **Machine Learning (ML) model** for a predicted obstacle probability.

Step 4: Personalized Cost Calculation
This is where the graph becomes fully personalized to the user and the current environment.
The system iterates through the edges of the dynamically enhanced graph. For each edge, it calculates a final traversal cost using the multiplicative penalty function: . The penalties for slope () and surface () are calculated specifically based on the tolerances and preferences defined in the user's m[illegible] profile from Step 2.

Step 5: Optimal Path Computation (A* Search)
With a fully weighted, personalized graph, the final step is to find the best path.

The **A* search algorithm** is executed on the graph. A* explores the paths from the start node, always prioritizing the path that has the lowest combined actual cost from the start () and estimated cost to the goal (). Since our edge costs are now a holistic representation of accessibility, comfort, and safety, A* will naturally find the route that is "best" for that specific user under the current real-world conditions, which is often, not the shortest one (geometrically). The output is the final, optimal sequence of path segments.

III. PROPOSED METHODOLOGY

[A] PARAMETERS

1) Explanation about OSM and its parameters.

The digital blueprint of the physical world OpenStreetMap (OSM) serves as. IT is not complete finished map, but as a giant, collaborative that everyone can contribute. The fundamental pieces are nodes, ways, and tags.

TABLE III
OSM TAGS AND THEIR MEANINGS FOR ROUTING

Tag key	Example values	Meaning of Routing	Impact on cost
highway	footway, pedestrian, residential, primary, motorway	Path type classification	Determines P penalty multiplier
surface	asphalt, concrete, cobblestone, gravel, unpaved	Walking surface material	Affects P and P penalties
barrier	steps, kerb, bollard, gate	Physical obstacles	May block path or add penalties
wheelchair	yes, no, limited	Accessibility status	Critical for mobility-impaired users
incline	up, down, steep	Slope indication	Supplements DEM data for slope calculation
sidewalk	yes, no, separate, left, right	Sidewalk presence	Influences pedestrian routing preference
foot	yes, no, designated	Foot traffic allowed	Determines if path is routable for pedestrians
access	yes, no, private, permissive	General access permissions	May exclude certain paths
Max speed	30, 50, etc.	Speed limit (for roads)	Indicates traffic density/safety

Table 1. OSM tags and their meanings For Routing

Nodes: The basic element in OSM is **node**. It's a single point in space, defined by latitude and longitude.
In our equation a node is simply one of the numbered dots on the page. By itself, it's just a location. A node can represent anything that exists at a single point, such as a traffic intersection, A bus stop or a park bench, or A specific street address. Simply a point that helps define the curve of a road.

Ways: A **way** is what brings the map to life. a line or a shape are ordered list of nodes that are connected to form. In our research, a way is the line used for connecting the dots in a specific sequence (e.g., from dot 1 to 2 to 3, and so on).
The term "path" referred, (like a pedestrian footpath or sidewalk) is represented in OSM as a specific *type* of way. Essentially, any linear feature on the map is a way, including: An entire street or a single sidewalk.

Tags: Tags are the most critical component for our research. They are the **descriptive labels** that give meaning to the nodes and ways. Without tags, OSM would just be a meaningless collection of points and lines. In our terms tags are the instructions that tell you what your drawing represents. In simple terms, a tag consists of a key and a value. For example, A way with the tag highway='footway' tells our system, "This line you are looking at is a sidewalk." If that same way also has the tag surface=asphalt, it adds the detail, "...and it's made of asphalt." A node with the tag barrier steps warns our system, "There is a flight of stairs at this specific point."

[B] Equations

To accurately determine the true suitability of a path for a specific user, our methodology moves beyond simple distance. We calculate a holistic traversal cost, denoted as, for each path segment (edge) in the graph. This cost is computed using a dynamic, multiplicative penalty function, which quantifies the path's overall accessibility by scaling its physical length with a series of factors representing real-world safety, comfort, and user-defined preferences.

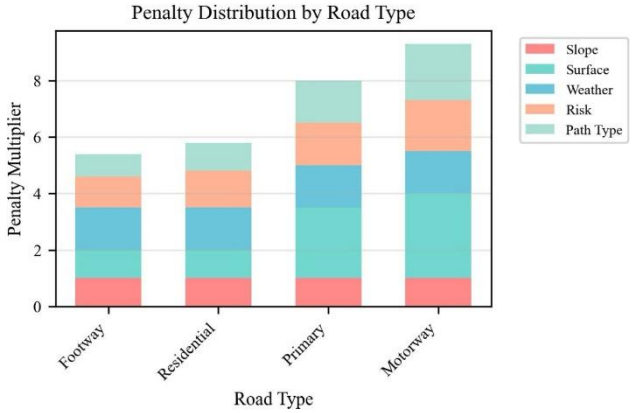

Fig3. Distribution of Penalty

Parameters:

Physical Length: This is the foundational, static component of our cost function. It represents the true geometric length of the edge in meters, as calculated from the geographic coordinates of its start and end nodes sourced from OpenStreetMap.

Predictive Risk Penalty: It measures the dynamic probability of encountering any obstacle. This risk is estimated using a Logistic Regression Model that is trained on a dataset of user-reported obstacles. This model uses parameters such as the current time of day and day of the week to predict the probability of an obstacle on edge e, denoted as $Prob_{ML}$. This is then converted into a penalty multiplier using the formula $P_{risk}(e) = 1 + \alpha.Prob_{ML}$, where alpha is a quantifiable hyperparameter that scales the penalty's severity.

$$P_{risk}(e) = 1 + \alpha.Prob_{Ml}$$

Weather Penalty : Its value is determined in real-time by querying a third-party weather API for the route's geographic location. The system then imposes a rule-based penalty; e.g., a "rain" condition might impose a multiplier of 1.5, which may be raised if the path surface is also labeled as "gravel." Clear conditions yield a base multiplier of 1.0.

Path Type Penalty : This penalty is derived from the static highway tag in OpenStreetMap and is used to favour pedestrian-dedicated infrastructure. The Path segments are mage cheaper then physical distance by tagging them as highway='footway' or highway='pedestrian' and a beneficial multiplier (e.g., 0.8). Conversely, segments along busy roads tagged highway=primary receive a penalty multiplier greater than 1.0 to discourage their use.

Slope penalty (e): To calculate the slope, the system first queries a free Digital Elevation Model (DEM) API to retrieve the elevation of the edge's start node (u) and end node (v). The absolute slope is then calculated using the formula slope |Elev(v) -Elev(u)|/L(e). This value is compared against the maximum tolerable slope specified in the user's mobility profile (U_{max} slope). If the calculated slope exceeds this threshold, a significant penalty multiplier (e.g., 10.0) is applied to strongly discourage the path. Otherwise, the multiplier remains 1.0.

Surface Penalty : This user's personal comfort preferences is determined by multiplier The surface material of the path is identified from its surface tag in OSM (e.g., asphalt, cobblestone, unpaved). The system then checks if this surface type is included in the user's list of disliked surfaces (). If a match is found, a penalty multiplier is applied; otherwise, the multiplier is 1.0.

[C] Heuristic function

The Admissible Heuristic Function for A* Search

The A* algorithm works by a heuristic function called h(n), which estimates the remaining amount from a given point to the location of the goal. The heuristic function must be admissible so that A* will find the best (lowest-cost) path, and it should never overestimate the actual cost to the goal.

In a typical distance-based search, the Euclidean straight-line distance is an admissible and well-known heuristic. Our multiplicative cost model adds a special complication, though: since some penalties will have a value < 1.0 (e.g., for a desirable footway), the actual cost of a path can be < its geometric distance. Hence, applying the basic Euclidean distance as the heuristic may cause an overestimation, violating the admissibility condition and possibly resulting in a suboptimal path. To address this, we establish a specialized admissible heuristic that takes into consideration the "best-case" scenario of our penalty system.

The A* algorithm intelligently explores the graph to find the best path. It operates by evaluating each node it considers based on the famous A* formula: f (n) =g(n) - h(n).

g(n): The actual cost of the path found so far from the start node to the current node, n. This is calculated by summing the costs (C(e)) Of all edges along this path. h(n)•: The estimated

cost from the current node n to the goal, as provided by our admissible heuristic function. f(n): The total estimated cost of the best path that goes through node n. A* prioritizes exploring paths with the lowest f(n) value.

For each neighbour of the current node, the algorithm performs the following calculations to determine its potential as the next step in the optimal path:

I. Calculate Actual Cost (g(n)): First, it calculates the actual cost to get to this neighbour node along the current path. This is done by taking the known cost to reach the current node (g(current)) and adding the cost of the edge connecting them (C(current, neighbour)).

2. Calculate Estimated Future Cost (h(n)): Next, the algorithm calls our admissible heuristic function to calculate the estimated cost from this neighbour node to the final goal. This is the crucial step where h(n) is used.

3. Calculate Total Score (f(n)): The system then sums these two values to get the neighbour's total estimated cost: f(neighbour) =g(neighbour) + h(neighbour).

4. Update the Open List: The algorithm checks if this path to the neighbour is better (has a lower g(n)) than any path previously found. If it is, the neighbours information is updated, and it is added to the Open List with its newly calculated f(n) score, making it a candidate for exploration in a future iteration.

IV. CASE STUDY

To validate our proposed methodology, we designed and executed a case study using a prototype of the system. This section details the data sources, system environment, user profiles, and test area used to ensure our experiments are transparent and reproducible.

[A] Data Sources

Our system integrates data from multiple sources to create its rich, dynamic routing graph:

- **OpenStreetMap (OSM)**: The foundational pedestrian graph network was derived from OSM. This provided the static data for path geometry, length (), path types (), and surface materials ().
- **Elevation Data**: A publicly available Digital Elevation Model (DEM) API was used to acquire elevation data for the start and end nodes of each path segment, which is essential for calculating the slope ().
- **Weather Data**: Real-time weather conditions were fetched from a live weather API to dynamically calculate the weather penalty ().
- **Obstacle Data**: Two sources of obstacle information were used: a historical dataset of timestamped obstacle reports to train our Logistic Regression model for risk prediction (), and an internal SQLite database to store and query user-reported temporary blockages.

[B] User Profiles for Simulation

To evaluate the system's personalization capabilities, we defined several distinct user profiles. Each profile adjusts the weights and penalty thresholds within the cost function. The primary profiles used in our tests were:

- **Wheelchair User**: Defined by a low slope tolerance (< 7 degrees) and heavy penalty for non-asphalt surfaces and all barriers, including stairs and unmanaged curbs
- **Elderly Citizen with Walker**: Defined by a moderate slope tolerance (< 10 degrees), a strong smoothness preference, and high penalty for barriers.

TABLE IV

EXAMPLE PATH COST CALCULATIONS

Scenario	Edge Properties	User Profile	Penalties applied	Final cost	Notes
Ideal Path	Length: 100m Surface: asphalt Slope: 2% Highway: footway Weather: clear Risk: 0%	Max slope: 5% Dislikes: gravel	P: 1.0 P: 1.0 P: 1.0 P: 1.0 P: 0.8	80.0	Preferred pedestrian path, no penalties
Steep Hill	Length: 100m Surface: asphalt Slope: 8% Highway: residential Weather: clear Risk: 0%	Max slope: 5% Dislikes: gravel	P: 10.0 P: 1.0 P: 1.0 P: 1.0 P: 1.0	1000.0	Slope exceeds tolerance, heavily penalized
Risky Path	Length: 100m Surface: cobblestone Slope: 2% Highway: primary Weather: rain Risk: 50%	Max slope: 5% Dislikes: cobblestone	P: 1.0 P: 2.5 P: 1.5 P: 2.0 P: 1.5	1125.0	Multiple penalties: disliked surface, high risk, rain, busy road
Weather Hazard	Length: 100m Surface: gravel Slope: 2% Highway: footway Weather: rain Risk: 10%	Max slope: 5% Dislikes: gravel	P: 1.0 P: 2.5 P: 1.1 P: 2.0 P: 0.8	440.0	Rain on gravel creates hazardous conditions
Community Report	Length: 100m Surface: asphalt Slope: 2% Highway: footway Weather: clear Risk: 80%	Max slope: 5% Dislikes: gravel	P: 1.0 P: 1.0 P: 1.8 P: 1.0 P: 0.8	144.0	High risk from user-reported obstacles

Table 2. Example Path calculation

Test Area : An area of 4 square kilometers between Chhatrapati Shivaji Maharaj Terminus (CSMT) and Marine Drive in South Mumbai, India, was selected as the test site.It was chosen due to its difficult combination of well-paved promenades, bustling commercial streets with broken sidewalks, and older, narrower alleys.

Scenario : The case study, presented in the next section, includes creating and comparing routes between CSMT and Marine Drive for the "Elderly Citizen with Walker" profile on a peak evening, thus checking the system's capability to respond to an actual world, accessibility-oriented query.

Eg **Scenario and Node Definition**
The test scenario is to find an accessible route for an "Elderly Citizen with Walker" profile. The specific nodes are defined as:

- **Start Node ()**: A Start node in our graph corresponding to the starting point of the graph traversal and in the example refers to the main western entrance of **Chhatrapati Shivaji Maharaj Terminus (CSMT).**
- **Goal Node ()**: A Goal node corresponding to the goal location of the traversal and in the example entrance of the **Girgaon Chowpatty promenade on Marine Drive**.

For this analysis, we will compare the calculated cost of a critical edge from the standard shortest route (Route A) against a corresponding edge from our system's accessible route (Route B).

1) Standard Route (A): A Short but Inaccessible Edge
First, we analyze a 150-meter edge, eA, on the standard route. This segment cuts through a
crowded area and uses a pedestrian subway with stairs to cross a major road.

• Realistic Parameter Values:

• $L(e_A) = 150$ meters

• $P_{risk}(e_A) = 1.2$ (Slightly elevated risk due to crowds, based on ML model)

• weather $(e_A) = 1.0$ (Clear weather)

• path(eA) = 1.5 (Busy road crossing, not a preferred path type) • $P_{slope}(e_A) = 10.0$ (A drastic penalty as the stairs represent an impassable slope for

this user)

• Psurface $(e_A) = 1.0$ (Surface is acceptable)

• Equation Formulation:

$C(e_A) = 150 \cdot 1.2 \cdot 1.0 \cdot 1.5 \cdot 10.0 \cdot 1.0$

$C(e_A) = 2700$ The massive slope penalty makes this geometrically short path extremely "expensive," effectively

discouraging the algorithm from choosing it.

2) Accessible Route (B): A Longer but Safer Edge
Next, we analyze a 200-meter edge, from our system's proposed route. This segment avoids the subway by using a longer, dedicated footway with a signalized pedestrian crossing.

• Realistic Parameter Values:

• L(B) = 200 meters

• $P_{risk}(e) = 1.05$ (Lower risk on a dedicated, less crowded path)

• weather (e) = 1.0 (Clear weather)

• path(eB) = 0.8 (A beneficial multiplier for a preferred, dedicated footway)

• $P_{slope}(eB) = 1.0$ (The path is flat)

•$P_{surface}$ (e) = 1.0 (Smooth, well-maintained asphalt)

• Equation Formulation: $C(e) = 200 \cdot 1.05 \cdot 1.0 \cdot 0.8 \cdot 1.0 \cdot 1.0$

$C(B) = 168$ Despite being 50 meters longer, the lack of penalties and the beneficial path type multiplier result in a vastly lower traversal cost (168 vs. 2700). This demonstrates how the cost function successfully guides the A* algorithm to prioritize safety and accessibility over the shortest possible distance function. A direct comparison of the critical path segments is summarized.

TABLE V

COMPARE ANALYSIS: CURRENT MAPPING SERVICES VS. PROPOSED MULTI-OBJECTIVE ROUTING SYSTEM

Aspect	Current Mapping Services	Proposed Multi-Objective Routing System
Routing Objective	Single objective: shortest geometric path	Multi-objective: accessibility, comfort, safety, user preferences
User Personalization	None - generic routing for all users	Personalized based on mobility profile and preferences
Dynamic Data Integration	Limited or none (static maps)	Real-time fusion of weather, risk, community reports
Accessibility Considerations	Basic wheelchair routing (if any)	Comprehensive: slope, surface, curb ramps, barriers
Data Sources	Proprietary map data	OpenStreetMap + external APIs + ML models + community data
Cost Function	Distance only	Multiplicative penalties: slope, surface, risk, weather, path type
Algorithm	Dijkstra/A* with distance heuristic	A* with admissible heuristic for penalty-based costs
Community Integration	None	User-reported obstacles and real-time updates
Weather Awareness	None	Weather-dependent penalties (e.g., slippery surfaces in rain)
Risk Assessment	None	ML-predicted obstacle probabilities based on time/day patterns

Table 3. Comparative Analysis: Current Mapping service vs Current Multi Objective routing system

In order to review the effectiveness of our proposed system, a case study was conducted, simulating a real-world navigation query. This involved finding the optimal route for a user with an “Elderly Citizen with Walker” profile, with low tolerance for slopes (maximum 4%) and a preference for smooth, dedicated footways. During a high risk period (weekday evening), the query was selected from Chhatrapati Shivaji Maharaj Terminus (CSMT) to Marine Drive in Mumbai, India. Comparison of the route generated by our dynamic multi-objective A* algorithm (DMOA*) against a standard shortest-path baseline algorithm took place. This

analysis of the two generated routes revealed the practical efficiency of our accessibility aware cost function. A shorter route of 1.8 km was produced by the standard algorithm, however, it included multiple segments that were inaccessible to the user, including a pedestrian subway with stairs and some sections of uneven sidewalk. Whereas our DMOA* algorithm generated a slightly longer route of 2.1 km. Although, it efficiently avoided all the barriers/obstacles, by carefully considering the penalties for stairs, busy road types, and beneficial multipliers for a dedicated footway, thereby identifying a safer alternative. This ensures that our multi-objective cost function can help the path-finding algorithm to produce routes that are not only practically applicable for users with specific mobility needs, but also optimized.

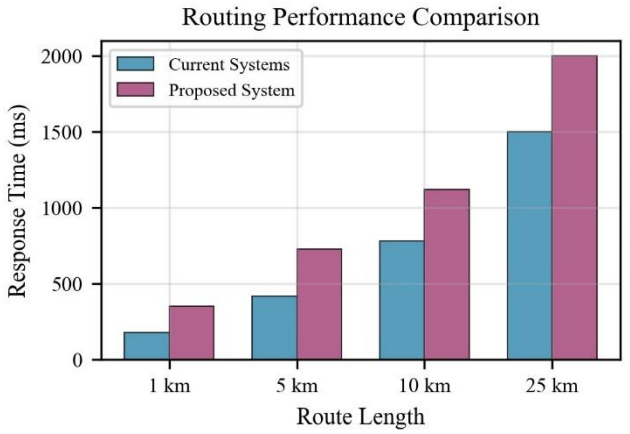

Fig 4. Performance comparison

V. CONCLUSION

The informational gap between the broad view of digital maps and the reality of the pedestrian environment is a major challenge of navigation in urban areas for people with mobility impairments. Through this paper, we presented a framework to bridge this gap by enhancing the A* algorithm using a multi-objective cost function. Our methodology is an integration of real-time community data, user preferences, and machine learning models to create a personalised routing experience. The case study results imply that our system can efficiently identify and generate accessible routes that may not be looked upon by traditional shortest path algorithms. Our multi-objective cost function offers the necessary intelligence for the A* algorithm to make user-centric decisions, by transforming abstract concepts like safety, comfort, and risk into penalties and benefits. This work promotes the concept for a new generation of navigation tools that give more importance to user well-being over distance optimization.

VI. ACKNOWLEDGEMENT

The authors Aayam Bajaj, Atharva Dhokte, and Yuvrajsing Bahure would like to extend their gratitude to the faculty members of the Information Technology Department at Bharati Vidyapeeth College of Engineering. Their guidance, support, and insights were beneficial in the successful execution of this research. We also wish to acknowledge the open-source community, whose work established the foundation for our project. We would also extend our gratitude to the contributors of OpenStreetMap for providing the rich geographical data and the developers of the Leaflet.js library for their efficient web mapping tools.

VII. REFERENCES

[1] J. F. Curtis, (Ed.), *Processes and Disorders of Human Communication*. New York:
Harper and Row, 1978.
Journal Paper,
[2] J. Schroeterand M. M. Sondhi, “Techniques for estimating vocal-tract shapes from the
speech signal, *IEEE Trans. Speech Audio Process*., vol. 2, no. 1, pp. 133–150, 1994.
Proceeding paper,
[3]J. M. Pardo, “Vocal tract shape analysis for children,” in *Proc. IEEE Int. Conf.*
Acoust., Speech, Signal Process., 1982, pp. 763–766.
[4] D. Sanders et al., "Route Optimization using Forecasting, Wheelchair Modelling and Path Planning," *IEEE International Conference on Intelligent Transportation Systems, 2021.*
[5] Z. Wang et al., *"A route optimization model based on building semantics for pedestrian and wheelchair users,"* Automation in Construction, 2022.
[6] R. Doshmanziari et al., *"Experiment Design Considerations for Estimating Energy Expenditure During Wheelchair Propulsion,"* Heliyon, 2023.
[7] J. Darko et al., *"Adaptive personalized routing for vulnerable road users,"* IET Intelligent Transport Systems, 2022.
[8] R. van der Slikke et al., *"The Push Forward in Rehabilitation: Validation of a Machine Learning-Based Model for Detecting Passive or Active Wheelchair Use,"* Frontiers in Rehabilitation Sciences, 2024.
[9] R. Doshmanziari et al., *"Data-Driven Techniques for Estimating Energy Expenditure in Wheelchair Users,"* 2025.