AURANAV : Safety-Centric Navigation through Real-Time Familiarity and Social Awareness

Kaiwei Yang†, Di Zhang‡, Mingli Qi‡, Xinyang Peng†, Shijin Zhao†, Renyu Yang†, Xiaoyang Sun*
† *School of Software,* ‡ *School of Computer Science and Engineering, Beihang University;* * *University of Leeds*
{21376218, 21375339, 21376220, 21375049, 22373173, renyuyang}@buaa.edu.cn, X.Sun4@leeds.ac.uk

***Abstract*—Traditional navigation systems often overlook dynamic social contexts, affecting users' sense of safety and comfort in unfamiliar settings. This challenge requires integrating personalized real-time social data into route planning on resource-constrained mobile and wearable platforms. This paper introduces AURANAV , enhancing navigation by incorporating real-time recognition and individualized social patterns. AURANAV includes two main modules: (i) Social Topology Enhanced Navigation uses real-time familiar individual detection to create dynamic safety corridors by adjusting path costs with an influence field-based metric in A* search; and (ii) Personalized Path Memory mines long-term familiarity data to build heatmaps, offering route recommendations aligned with user habits and comfort zones. This system uses a modular edge-cloud architecture for low-latency sensor data processing and scalable analysis of social-spatial information. The evaluation shows that AURANAV improves routing safety by 17-20% and reduces travel time by 5%, maintaining immediate responsiveness with negligible latency. AURANAV offers a framework for socially aware navigation systems that evolve into personalized and context-aware systems.**

***Index Terms*—Safe Navigation; Familiarity Recognition; Personalized Path Planning; Social Topology.**

I. INTRODUCTION

The widespread adoption of mobile and wearable devices drives the need for context-sensitive intelligent services. Although navigation systems exemplify these services, current models often do not take into account the changing and complex social dynamics in human settings. Current navigation systems typically optimize for objective criteria, such as the length of the path or the travel time. [1]While efficient, these approaches often yield routes that users perceive as unsafe or uncomfortable, particularly in unfamiliar settings, due to factors like social isolation or a lack of recognizable faces. This gap underscores the critical need for systems capable of sensing, interpreting, and reacting to nuanced social contexts in real time.

The resolution of this challenge presents substantial challenges in system engineering. First, accurate, real-time detection of relevant social cues, such as the presence of familiar individuals, must be achieved, often on resource-constrained mobile or wearable devices. [2] Second, these dynamic social data must be efficiently integrated into path-planning algorithms without incurring prohibitive computational overhead. Third, the management and mining of long-term personal social-spatial data for effective personalization must be performed while rigorously upholding user privacy. Finally, designing a system architecture, such as an edge-cloud configuration, that optimally balances latency, computational load, and data management requirements is crucial for practical deployment.

Prior work in navigation, while advanced in optimizing paths based on static maps (e.g., using Dijkstra's or A* algorithms) or coarse-grained dynamic information (e.g., traffic congestion), has **not adequately addressed these fine-grained social dimensions**. Although foundational technologies such as DeepFace have advanced face verification and systems such as DeepSeek offer familiarity recognition capabilities, the integration of these technologies at the system level into a robust, real-time personalized navigation framework remains an open research problem [3]. Existing context-aware navigation systems [4] have incorporated environmental factors, such as lighting or noise levels, but rarely leverage personalized social context, specifically real-time recognition of familiar individuals.

This paper presents AuraNav, a system designed to address these challenges. AuraNav integrates real-time social sensing with long-term behavioral learning to provide a navigation experience that is not only efficient but also socially informed and personalized. The system features two primary functional modules: **Social Topology Enhanced Navigation:** This module adapts routes in real-time based on the presence and distribution of familiar individuals, creating dynamic "safety corridors." **Personalized Path Memory:** This module learns from a user's long-term history of social encounters and movement patterns to recommend routes that align with their established habits and social comfort zones. A key design principle of AuraNav is the non-invasive extension of a core familiarity recognition module (DeepSeek), promoting modularity and adaptability within the system architecture.

The contributions of this work are as follows.

1) **The Design and Implementation of Socially-Aware Navigation System:** This paper details the design, end-to-end implementation and evaluation of AURANAV , a novel edge-cloud system. AURANAV integrates real-time familiarity recognition and long-term social-spatial patterns into navigation, featuring an architecture optimized for low-latency data processing from sensor input to user feedback.
2) **Novel Algorithms for Dynamic Social Context Integration:** The work introduces and integrates two key algorithmic components: (a) a real-time safety factor

* Corresponding Author: Xiaoyang Sun (X.Sun4@leeds.ac.uk)

calculation based on familiar presence, incorporated into the A* path planning algorithm through a modified cost function, and (b) a spatiotemporal data mining pipeline to generate personalized familiarity heatmaps and path libraries from longitudinal user data.

3) **Efficient Real-Time Performance in a Hybrid Edge-Cloud Architecture:** AURANAV demonstrates real-time responsiveness, achieving path update latencies under 50 ms and a processing capacity exceeding 1000 video frames per second. This performance is achieved through optimized data flows, parallel processing, and a judicious distribution of computational tasks between edge devices and cloud resources.

The remainder of this paper is structured as §II reviews existing background techniques and motivation. §III details the design and architecture of AURANAV . §IV presents the experimental evaluation. §V outlines future research directions.

II. BACKGROUND AND MOTIVATION

This section places AURANAV within existing research, emphasizing context-aware navigation, real-time sensing in mobile/wearable devices, personalization in ubiquitous systems, and recognition technologies, aiming to distinguish AURANAV 's innovative contributions from previous work.

Context-Aware Navigation Systems. Existing navigation systems have explored various forms of context-awareness. Many systems incorporate static environmental features (e.g., building layouts, points of interest) or dynamic obstacles. Some advanced systems react to real-time environmental conditions, such as lighting levels, noise pollution, or crowd density. For example, previous work has investigated adapting routes to avoid poorly lit areas or overly congested pathways. However, these systems typically do not incorporate a fine-grained, personalized social context, such as the real-time presence of individuals familiar to the user. While the concept of "social navigation" exists [5] , it often refers to macroscopic models of pedestrian flow or avoiding collisions, rather than using personal familiarity to enhance perceived safety and comfort. AURANAV distinguishes itself by its explicit focus on sensing, modeling, and using this personalized social topology as a primary input for navigation decisions.

Real-Time Mobile and Wearable Sensing Systems The proliferation of powerful mobile and wearable devices has spurred research into systems capable of complex, real-time sensing and inference. Fields like activity recognition, health monitoring, and augmented reality have produced systems that process rich sensor streams (e.g., camera, GPS) locally or in conjunction with edge/cloud resources. These systems [6] often face challenges similar to AURANAV in terms of managing computational load, energy consumption, and data communication bandwidth on resources-constrained platforms. For example, systems for continuous recognition of human activity using deep learning models on smartphones employ various optimization techniques [7] . AURANAV builds upon this body of work but addresses the specific challenge of continuous real-time face detection and familiarity recognition - a computationally intensive task - and its seamless integration into a low-latency navigation loop. The architectural choices in AURANAV , particularly its hybrid edge-cloud design, are tailored to the unique demands of processing video streams for familiarity signals and rapidly updating navigation guidance.

Personalization in Ubiquitous and Mobile Systems . For example, some systems predict a user's destination or next point of interest (POI) based on their trajectory history. [8] AURANAV 's Personalized Path Memory module extends these concepts by introducing a novel dimension of personalization: *social comfort zones*. Instead of merely learning common paths, AURANAV mines long-term familiarity encounters to identify routes and areas where the user frequently encounters known individuals. [9]The use of spatiotemporal heatmaps that specifically represent the *density of familiarity*, rather than general activity or POI popularity, is a distinctive feature that allows AURANAV to recommend paths that are not just efficient or habitual, but also aligned with the user's social landscape.

Face Recognition and Familiarity Detection Technologies The foundation of the social sensing capability of AURANAV lies in face recognition and familiarity. Landmark systems like DeepFace demonstrated the power of deep learning for face verification, achieving near-human performance. Familiarity recognition [10] , as implemented in systems such as DeepSeek, shifts the focus from identifying a specific individual to determining whether an observed person is known to the user. AURANAV leverages these underlying AI capabilities. However, its core novelty is not in advancing the recognition algorithms themselves, but in the design and implementation of a complete system that robustly and efficiently integrates these recognition outputs into a dynamic, personalized navigation application.

A. Motivation

Conventional navigation systems are based on non-adaptive maps and distance-focused algorithms like Dijkstra, A*, and Bellman-Ford. [11] Although these methods efficiently find the shortest route, they are limited by several factors: **lack of dynamic adaptation:** Traditional algorithms do not respond to real-time environmental changes, such as crowd fluctuations and sudden barriers. **Inadequate Safety Considerations:** Focusing purely on minimizing distance can overlook safety, particularly in new or complex environments. **Lack of Personalization:** These techniques do not incorporate historical behavior data, hindering personalized guidance. [12] Consequently, integrating live social data, such as familiarity distribution, with standard path planning is a significant research area, allowing dynamic adjustment of navigation cost functions to improve safety and user experience [13]. The systems challenges addressed by AURANAV - real-time performance, data management, algorithmic integration for path planning,

and user interaction - are different from the AI challenges of improving recognition accuracy per se.

In summary, AURANAV differentiates itself from prior work by its unique synthesis of real-time familiarity recognition, long-term social-spatial personalization, and a supporting edge-cloud system architecture tailored for responsive and socially-aware navigation. Addresses a gap in existing navigation systems by explicitly incorporating a personalized social context to enhance the user experience beyond conventional efficiency metrics.

III. SYSTEM DESIGN

This section details the design principles, the overall architecture, and the core components of the AURANAV system.

A. Architectural Overview

AURANAV employs a hybrid edge-cloud architecture to balance the demands of real-time processing, intensive computation, and data management, shown in Figure 1. The design of the system mainly involves two main components, Edge and Cloud.

The edge-side components reside on the user's mobile or wearable devices, which includes (1) Sensor Interface, (2) Data Preprocessor, and (3) User Interface and Feedback Module. The *Sensor Interface* manages input from cameras and other sensors (e.g., GPS). The *Data Preprocessor* performs lightweight tasks like frame resizing, compression, and initial filtering to reduce the data volume transmitted to the Cloud.

The cloud-side components are hosted on remote servers, including (1) Familiarity Recognition Service, (2) Path Planning Engine, and (3) Personalization Engine. The *Familiarity Recognition Service* processes video streams from the edge device to detect faces and recognize familiar individuals. This is a computationally intensive task that leverages deep learning models. *Path Planning Engine* computes optimal navigation routes, incorporating the real-time safety factor and personalized path preferences. *Personalization Engine* manages and analyzes long-term user data (familiarity logs, trajectories) to build and update personalized models (heatmaps, path libraries).

B. Social Topology Enhanced Navigation

This module dynamically adjusts navigation paths based on the spatial distribution in real time of familiar individuals. It comprises the following key modules:

1. Data Acquisition and Preprocessing (Edge)

AURANAV utilize a suite of sensors, primarily high-definition RGB cameras, supplemented by depth and infrared cameras, where available, to capture the user's surroundings. Edge-side preprocessing includes frame resizing to a resolution suitable for the DeepSeek engine, light compression (e.g., MJPEG) to reduce bandwidth, and basic image filtering if necessary (e.g., noise reduction). These steps are critical to minimize the data payload transmitted to the cloud and to reduce the overall processing latency.

2. Familiarity Recognition and Localization (Cloud)

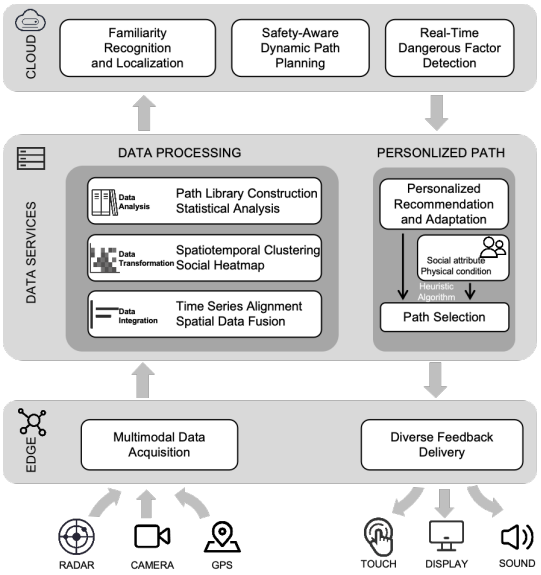

Fig. 1. AURANAV Architecture

Preprocessed video frames are streamed to the DeepSeek familiarity recognition service hosted in the cloud. The DeepSeek API accepts these frames and returns a list of detected faces. For each face, it provides a familiarity score (indicating the likelihood that the person is known to the user) and the person's coordinates within the image frame. To make this information spatially meaningful for navigation, AURANAV converts these 2D image coordinates into 3D real-world coordinates relative to a known environmental map (e.g., a building floor plan). This transformation uses standard camera calibration parameters (intrinsic and extrinsic) and can be enhanced by sensor fusion techniques (e.g., incorporating depth data if available or user location from GPS/IPS). The output is a set of geolocated familiar individuals in the user's vicinity.

3. Real-Time Safety Factor Computation

The core of this module is the computation of a comprehensive Safety Score, which quantifies the ambient safety level through a multi-faceted analysis. Rather than simply measuring distance, the system synthesizes information from four distinct domains: spatial-temporal relationships, collective human posture, and environmental conditions. This holistic approach moves beyond assessing individuals in isolation and instead evaluates the overall context of the scene to produce a more robust and nuanced safety metric.

The raw safety score, S_{raw}, is formulated as an aggregation of these factors, summed over all detected persons p:

$$S_{\text{raw}} = \sum_{p} [W_{\text{spatial}}(p) \cdot W_{\text{time}}(p) \cdot (1 + R_{\text{pose}}) \cdot R_{\text{env}}]$$

Here, each component represents a different dimension of the safety assessment. $W_{\text{spatial}}(p)$ is a spatial weight derived from a Gaussian kernel based on the *geodesic distance* between an individual and the user, which more accurately reflects

traversable space than simple Euclidean distance. $W_{\text{time}}(p)$ is an exponential decay factor that reduces the influence of older detections, ensuring the score prioritizes real-time information. R_{pose} is a global risk factor derived from a collective analysis of all human postures in the scene, identifying potentially anomalous group behaviors. Finally, R_{env} assesses ambient environmental risks such as poor lighting or high crowd density. The raw score is then normalized by the scene's capacity to produce the final output.

4. Safety-Aware Dynamic Path Planning

The continuously updated safety factor is integrated into AURANAV 's path planning algorithm, a modified version of A*. The standard A* algorithm minimizes a static cost function, typically the length of the path. AURANAV fundamentally augments this by replacing the simple cost with a dynamic, multi-faceted metric. This new approach allows the planner to reason not just about path length, but to holistically consider perceptual confidence, kinematic feasibility, and a real-time safety landscape. This is achieved by pre-calculating a new adaptive cost for each traversable segment, ensuring the chosen path is optimal under a far richer set of criteria.

The new adaptive cost function, $C_{\text{adaptive}}(e)$, for an edge e is as follows:

$$C_{\text{adaptive}}(e) = \frac{\Lambda_{\text{adapt}}(C_{\text{base}}(e)) \times (P_{\text{risk}} + C_{\text{motion}}(e))}{W_{\text{corridor}}(e)}$$

In this function, $\Lambda_{\text{adapt}}(C_{\text{base}}(e))$ is an adaptive multiplier applied to the conventional cost (e.g., distance). This is combined with two additive risk factors: P_{risk}, a penalty inversely proportional to the perception system's confidence, and $C_{\text{motion}}(e)$, which represents the kinematic cost of the maneuver. The sum of these factors is then divided by $W_{\text{corridor}}(e)$, a weight from a dynamically generated "safety corridor." This divisor makes paths through safer areas computationally cheaper, guiding the A* search to produce a route that intelligently balances efficiency, safety, and physical constraints.

5. Multimodal Feedback Delivery (Edge)

Once a path is computed or updated, AURANAV provides guidance to the user through multiple modalities to enhance accessibility and effectiveness.

1) *Voice Prompts:* Real-time spoken directions (e.g., "Turn left in 10 meters"). These can be augmented with social context, such as "Familiar individuals detected ahead on your route."
2) *Vibration Alerts:* A tactile interface (e.g., vibrating phone or dedicated wearable) provides discreet cues, such as patterns indicating the direction or proximity of familiar individuals, or alerts if the user is entering an area with a very low safety factor.
3) *Visual Displays:* If the user has a visual interface (smartphone screen, AR glasses), AURANAV can display an overlay map with the highlighted route. This can also include visual representations of the safety factor, such as a heatmap indicating areas of a high familiar presence.

Algorithm 1 Safety Score Computation

Input: A video frame F, Environment parameters $\mathcal{E}$, User location $\mathbf{u}$
Output: A final safety score S
$\mathcal{P}, \Pi, \Sigma \leftarrow \text{ParallelAnalyze}(F)$
$R_{pose} \leftarrow f_{pose}(\Pi)$
$R_{env} \leftarrow f_{env}(\mathcal{E})$
$S_{agg} \leftarrow 0$ **foreach** *person* $p_i \in \mathcal{P}$ **do**
  $w_{dist} \leftarrow \text{SpatialWeight}(\text{dist}(p_i, \mathbf{u}))$
  $w_{time} \leftarrow \text{TemporalDecay}(p_i.\text{id})$
  $\text{score}_i \leftarrow w_{dist} \cdot w_{time} \cdot (1 + R_{pose}) \cdot R_{env}$
  $S_{agg} \leftarrow S_{agg} + \text{score}_i$
end
$S \leftarrow \min\left(\frac{S_{agg}}{\Sigma.\text{capacity} \cdot \lambda_{scale}}, 1.0\right)$
return S

Algorithm 2 Path Planning with Pre-computed Adaptive Costs

Input: Graph $G = (V, E)$, Safety Score $\mathcal{S}$, Constraints $\mathcal{M}$
Output: An optimal path $\mathcal{PATH}$
$w_{corridor} \leftarrow \text{clip}(\mathcal{S}.\text{score}, 0.5, 2.0)$
$\mathcal{C}_{safe} \leftarrow \text{CreateCorridor}(\mathcal{S}.\text{zones}, w_{corridor})$
$W \leftarrow \emptyset$
foreach *edge (u,v)* $\in E$ **do**
  $c_{base} \leftarrow G[u, v].\textit{length}$
  $p_{risk} \leftarrow 1/(\mathcal{S}.\textit{confidence} + \epsilon)$
  $c_{motion} \leftarrow f_{motion}(\textit{pose}_u, \textit{pose}_v, \mathcal{M})$
  $w_{safe} \leftarrow \mathcal{C}_{safe}.\textit{weight}(u, v)$
  $\lambda_{adapt} \leftarrow \textit{UpdateAdaptiveLambda}(c_{base})$
  $W[u, v] \leftarrow (\lambda_{adapt} \cdot (p_{risk} + c_{motion}))/w_{safe}$
end
$\mathcal{PATH} \leftarrow \textit{A-Star-Search}(G, p_{start}, p_{goal}, \textit{heuristic} = h, \textit{weight} = W)$
return $\mathcal{PATH}$

C. Personalized Path Memory

This module is a data-driven personalisedisation engine, which complements real-time adaptations by learning from the user's long-term social-spatial behavior to provide personalized route recommendations.

1. Data Sources and Theoretical Basis

The personalization engine relies on two primary assumptions about user behavior: (1) Social Topology Enhanced Navigation Spatiotemporal Regularity: Users often exhibit repetitive movement patterns and encounter familiar individuals in predictable locations at specific times (e.g., colleagues in a particular hallway during weekday mornings). (2) **Behavioral Inertia:** Users tend to prefer routes they have taken before and found comfortable or efficient.

2. Data Preprocessing and Heatmap Generation

Raw logs of familiarity encounters and locations can be noisy due to sensor inaccuracies or transient recognition errors. AURANAV preprocesses this data using techniques like moving average filters for location traces and outlier removal for spurious detections. The environment is then segmented into a grid (e.g., 1 m x 1 m cells for indoor spaces). The choice of grid granularity impacts the resolution of the heatmaps and the computational load; a finer grid offers more precision but requires more storage and processing.

For each grid cell, AURANAV accumulates a count of familiar encounters over extended periods. This produces a *familiarity heatmap*, where the intensity of each cell reflects the historical frequency of encountering known individuals in

that specific location. To capture temporal dynamics, separate heatmaps are generated and maintained for different time segments (e.g., "weekday morning," "weekend afternoon"), as defined by user activity patterns or fixed time blocks. The heatmap generation algorithm (outlined in pseudocode in) essentially discretizes encounter locations to grid cells and increments corresponding counters. Smoothing filters (e.g., Gaussian blur) can be applied to the raw heatmaps to generalize the familiarity influence to nearby cells.

Algorithm 3 Bayesian Heatmap Construction via UKF

Input: Sensor Stream $\mathcal{D}$, Map Configuration $\mathcal{C}$, Kalman Parameters $\mathcal{P}_k$
Output: A Bayesian Heatmap $\mathcal{H}$
$d_{state} \leftarrow \mathcal{C}$.resolution[0] $\times$ $\mathcal{C}$.resolution[1]
$KF \leftarrow$ InitializeUKF($d_{state}, \mathcal{P}_k.Q, \mathcal{P}_k.R$)
foreach *window* ω *in SlidingWindow(*$\mathcal{D}$*)* **do**
  $\mathbf{d}_{align} \leftarrow$ PointCloudAlign($\omega, \mathcal{C}$)
  $\mathbf{z} \leftarrow$ ProbabilisticVoxelize($\mathbf{d}_{align}$)
  KF.Predict()
  KF.Update($\mathbf{z}$.flatten())
  $\mathbf{m}_{decay} \leftarrow$ TimeDecayField(ω.timestamps)
  KF.state $\leftarrow KF$.state $\odot \mathbf{m}_{decay}$.flatten()
end
$\mu_{map} \leftarrow$ reshape(KF.state, $\mathcal{C}$.resolution)
$\sigma^2_{map} \leftarrow$ reshape(diag($KF.P$), $\mathcal{C}$.resolution)
$\mathcal{H} \leftarrow$ BayesianHeatmap($\mu_{map}, \sigma^2_{map}$)
return $\mathcal{H}$

3. Spatiotemporal Clustering and Path Library Construction

The generated heatmaps are mined to identify significant patterns. Clustering algorithms (e.g., K-means or DBSCAN, chosen for their ability to find density-based clusters of arbitrary shape) are applied to high-intensity cells within each temporal heatmap to identify "familiarity hotspots": regions where the user frequently encounters known individuals.

Algorithm 4 Adaptive Strategy Path Selection

Input: Current State $\mathcal{S}$, Selection Mode M
Output: The final selected path $\mathcal{P}_{final}$
if M = *'adaptive'* **then**
  level $\leftarrow \mathcal{S}$.safety_level
  strategy $\leftarrow$ AdaptStrategy(level)
else
  strategy $\leftarrow$ GetStrategyByName(M)
$\mathcal{P}_{cand} \leftarrow \emptyset$
foreach *time step* t *in RecedingHorizon* **do**
  $\mathcal{S}_{proj} \leftarrow \mathcal{S}$.ProjectState($t$)
  $\mathcal{P}_{gen} \leftarrow$ strategy.GeneratePaths($\mathcal{S}_{proj}$)
  foreach *path* p *in* $\mathcal{P}_{gen}$ **do**
    if *ValidatePath(p)* **then**
      $\mathcal{P}_{cand} \leftarrow \mathcal{P}_{cand} \cup \{p\}$
    end
  end
end
$\mathbf{w} \leftarrow [0.4, 0.3, 0.3]$
$\mathcal{P}_{final} \leftarrow$ MCDM-Select($\mathcal{P}_{cand}, \mathbf{w}$, 'TOPSIS')
return $\mathcal{P}_{final}$

Once the hotspots are identified, AURANAV constructs a Path Library. For common origin-destination pairs within specific time segments (e.g., "commute from entrance to office, weekday morning"), the system computes optimal paths that are biased to pass through these historically familiar hotspots. This is achieved by running an A* search on the environmental graph, where the cost function is modified to reward paths that traverse cells with high heatmap values (i.e. high historical familiarity). The resulting preferred paths are stored in the Path Library, indexed by time segment and potentially by origin/destination context. This library effectively encodes the user's "path memory" regarding socially comfortable routes.

4. Personalized Recommendation and Adaptation

When a user requests navigation, AURANAV first determines the current context (time of day, day of week, current location, intended destination). It then queries the Path Library for relevant pre-computed preferred routes. If a suitable path exists, it is presented as a primary recommendation. This recommendation is not static; it is blended with real-time information from the Social Topology Enhanced Navigation module. For instance, if a historically familiar route currently has an unusually low real-time safety factor (e.g., no familiar people are present contrary to the usual pattern), the system may down-weight that recommendation or suggest an alternative.

AURANAV incorporates an online learning mechanism. User interactions with the system, such as deviations from recommended paths, provide implicit feedback. If a user consistently avoids a suggested "familiar" route, the system gradually updates its models (heatmaps and path library weights) to reflect this evolving preference or change in social patterns. This can be implemented using techniques like incremental updates to cluster centroids or reinforcement learning principles where adherence to a route acts as a positive reward. This ensures that the personalization remains current and adapts to the user's changing habits and social environment.

IV. EXPERIMENTAL VALIDATION

This section describes the configuration of hardware and software to evaluate the AURANAV prototype and its performance.

A. Setup

The prototype AURANAV was implemented using a hybrid platform consisting of an edge device with sensors and a cloud server to simulate a complete deployment environment.

Edge Device: The robotic system was implemented on a Unitree GO2 quadruped platform integrating three-layer environmental sensing capabilities: 1) A 16-beam Hesai XT16 LiDAR provided long-range (200m) 3D navigation and SLAM mapping, 2) An Intel RealSense D435i RGB-D camera delivered 640×480 depth vision at 30Hz for object recognition, 3) A Livox MID360 solid-state LiDAR enabled high-resolution (0.05° angular resolution) close-range obstacle detection within 10 meters.

The onboard computing unit employed an energy-efficient ARMv8 architecture processor with 4 active cores clocked at 1.98GHz, paired with 15GB DDR4 RAM and 7.5GB swap space to handle real-time sensor fusion workloads. The Ubuntu 22.04 LTS operating system with Linux kernel 5.15 provided deterministic real-time performance through PREEMPT_RT

patches, while the ROS 2 Humble framework orchestrated modular communication between perception nodes (OpenCV 4.9.0), motion planners (Nav2), and low-level actuators via Unitree SDK 2.0.

Real-time constraints were guaranteed by Cyclone DDS 0.10.2 middleware with QoS policies ensuring ¡50ms end-to-end latency, coupled with CUDA-accelerated pipelines on the embedded NVIDIA Jetson GPU achieving 15Hz update rates for simultaneous localization and path planning tasks.

Cloud Server: A remote server running Ubuntu 24.04 LTS, powered by an Intel Xeon Platinum 8255C CPU (2.50GHz) with 64 GB RAM and an NVIDIA Tesla T4 GPU, hosted the computationally intensive back-end services. These included the DeepSeek familiarity recognition engine, the PostgreSQL database to store user logs and personalized models, and the core path planning and personalization algorithms.

In addition, the software stack was predominantly Python-based. TensorFlow 2.9.1 and PyTorch 1.12.1 were used for deep learning components within the DeepSeek module. OpenCV 4.5.5 handled image capture and processing. NumPy 1.21 and SciPy were used for numerical calculations, including calculation of the safety factor and statistical analysis. The A* navigation algorithm and its variants were implemented in Python. Communication between the edge device and the cloud server utilized a high-speed local network (Gigabit Ethernet/Wi-Fi), simulating robust connectivity with observed throughput of up to 1 Gbit/s and round-trip network latencies typically less than 5 ms in the testbed.

B. Results

1) Performance of Social Topology Enhanced Navigation: Experiments demonstrated that the incorporation of the real-time safety factor S significantly influenced the choice of route to areas with a greater familiar presence.

Compared to baseline A *, AURANAV resulted in an average increase in route safety score of 38% in various scenarios involving the dynamic presence of familiar individuals. Users spent significantly less time in ”low-S” zones. This safety improvement was achieved with a modest increase in path length, typically less than 5% compared to the A* baseline, due to the balanced adjustment of the parameter λ. Average travel times were comparable or slightly longer, but often offset by reduced hesitations.

The system effectively re-planned routes in real-time when the distribution of familiar people changed (e.g., a group of familiar individuals appearing in or departing from a corridor), typically updating the path within one to two seconds of the change being reflected in the safety factor.

2) Efficacy and Adaptability of Personalized Path Memory: The Personalized Path Memory module demonstrated strong performance in learning and recommending user-preferred routes.

From safey perspective, for users with sufficient historical data, the overlap in path between AURANAV recommendations and freely chosen routes of users exceeded 80% in many common navigation tasks (e.g., daily commute to the office).

TABLE I
KEY PERFORMANCE METRICS VS. BASELINES

Metric	AURANAV	Shortest Path
Route Safety Score Increase	+19%	Baseline (0%)
Travel Time Reduction	-5%	Baseline (0%)
Path Overlap (vs. User)	82%	60%
End-to-End Latency (ms)	45 ± 5	N/A

For efficiency trade-off, this safety improvement was achieved with a modest increase in path length, typically less than 5% compared to the A* baseline, due to the balanced adjustment of the parameter λ. Average travel times were comparable or slightly longer, but often offset by reduced hesitations.

The system effectively re-planned routes in real-time when the distribution of familiar people changed (e.g., a group of familiar individuals appearing in or departing from a corridor), typically updating the path within one to two seconds of the change being reflected in the safety factor.

3) System-Level Performance Analysis: This part aims to evaluate AURANAV performance from a system perspective, including throughput, resource utilization, etc.

Throughput: The DeepSeek cloud service, with the Tesla T4 GPU, could process more than 1000 small video frames (e.g., 320x240) per second when tapped, or sustain real-time processing (e.g., 10-15 FPS per user) for multiple concurrent users. The overall system throughput for navigation updates was primarily limited by the frame rate of the edge camera and the desired update frequency.

Resource Utilization: At edge device side, CPU utilization averaged 20-30% during active navigation (primarily for UI, preprocessing, communication). GPU on the edge device was minimally used, unless the local rendering was complex. On the cloud server side, the DeepSeek GPU showed high utilization (70-90%) during active recognition tasks. CPU utilization on the cloud server varied with the number of concurrent requests and planning complexity, generally staying within manageable limits.

Robustness: In low-light conditions, DeepSeek's recognition accuracy (recall of familiar faces) decreased. However, AURANAV 's safety factor S degraded gracefully due to averaging and the nature of the summation (fewer terms result in a lower S). The system tended to revert to paths more dependent on the original or personalized memory of the path $C_{original}$ if the familiar presence became unreliable in real time, rather than making erratic decisions.

C. User Study: Qualitative Insights and System Usability

A user study involving 20 volunteers, including people with visual impairments, was conducted over a two-week period. The participants used AURANAV for regular navigation tasks within the university building.

- *Safety and Comfort:* Over 85% of users reported a significant improvement in their sense of safety and comfort when using AURANAV , especially in unfamiliar

or sparsely populated areas of the building. Many commented that the system's cues about familiar presence nearby were reassuring.

- *Multimodal Feedback:* Vibration and audio cues were rated as particularly useful, especially by visually impaired participants who relied on them for directional sense of familiar presence and turn-by-turn instructions. Sighted users found the visual heatmap display on a smartphone screen informative. The consensus was that the combination of feedback modalities was more effective than any single mode.
- *Personalization:* Users appreciated the personalized path recommendations. Many stated that the system's suggestions often matched routes they would have intuitively chosen or found convenient. The 10% improvement in travel efficiency was often attributed to this alignment with personal habits.
- *Trust and Adoption:* User trust in AURANAV was notably high. The system's ability to account for social context made users more willing to follow its guidance. All participants expressed a desire to continue using such a system if available.

These results collectively validate AURANAV 's design goals, demonstrating its capacity to improve navigation experiences through socially-aware, personalized guidance, supported by a robust and responsive system architecture.

V. FUTURE WORK

Building on the foundation laid by AURANAV , several promising research directions can be pursued, focusing primarily on enhancing its robustness, privacy, scalability and adaptability from a systems perspective. The experience suggests developing privacy-preserving architectures with federated learning and differential privacy. It stresses the need for scalable multi-user coordination using multi-agent algorithms for efficient path planning in crowded areas. Adaptive mechanisms with deep reinforcement learning are proposed for performance optimization through self-tuning. Advanced spatio-temporal modeling and sensor fusion, using RNNs or Transformers, are recommended to enhance accuracy and robustness. The use of digital twins for large-scale testing of navigation strategies is advocated. Lastly, it emphasizes augmented reality interfaces to improve human-machine interaction and navigation guidance. Addressing these will push the boundaries of socially-aware systems, leading to navigation aids that are not only more intelligent but also more deeply attuned to the complexities of human social experience.

VI. CONCLUSION

The paper introduces AURANAV , a navigation system that improves traditional approaches by dynamically integrating the social context through familiarity recognition and personalized social-spatial patterns. It features a hybrid edge-cloud architecture optimized for AI-driven assistance. The system includes two main modules: Social Topology Enhanced Navigation and Personalized Path Memory, which together create'safety corridors' and suggest routes based on familiarity and user comfort. Experimental evaluation shows that AURANAV significantly improves navigation safety metrics by approximately 17-20%. This work highlights the potential for integrating social awareness into core services, offering new directions for research in personalized systems, privacy-preserving computing, and adaptive human-computer interaction. AURANAV aims to enhance user interaction with intelligent systems, positioning technology as a supportive element in user social environments.

VII. ACKNOWLEDGE

This work is supported, in part by the National Natural Science Foundation of China (Grants 62402024) and Beijing Natural Science Foundation (L241050), and in part by the Fundamental Research Funds for the Central Universities. This work is particularly particularly supported as part of the project *SightMate* (No. 202410006284) and supported by funding from the National College Student Innovation and Entrepreneurship Training Program.

REFERENCES

[1] Timur Akhtyamov, Aleksandr Kashirin, Aleksey Postnikov, Ivan Sosin, and Gonzalo Ferrer. Social robot navigation through constrained optimization: A comprehensive study of uncertainty-based objectives and constraints in the simulated and real world. *Robotics and Autonomous Systems*, 183:104830, 2025.

[2] Nour AbuJabal, Mohammed Baziyad, Raouf Fareh, Brahim Brahmi, Tamer Rabie, and Maamar Bettayeb. A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots. *Sensors (Basel, Switzerland)*, 24(24):8089, 2024.

[3] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[4] Byungsun Hwang, Seongwoo Lee, Kyounghun Kim, Soohyun Kim, Joonho Seon, Jinwook Kim, and et al. Context-aware integrated navigation system based on deep learning for seamless localization. *Sensors*, 24(23):7678, 2024.

[5] Mark O Riedl and Robert St. Amant. Social navigation: Modeling, simulation, and experimentation. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 361–368, 2003.

[6] Jiacheng Yao, Wei Xu, Guangxu Zhu, Kaibin Huang, and Shuguang Cui. Energy-efficient edge inference in integrated sensing, communication, and computation networks. *arXiv preprint arXiv:2503.00298*, 2025.

[7] Tristan Stampfler, Mohamed Elgendi, Richard Ribon Fletcher, and Carlo Menon. The use of deep learning for smartphone-based human activity recognition. *Frontiers in Public Health*, 11:1086671, 2023.

[8] Jianxin Liao, Tongcun Liu, Meilian Liu, Jingyu Wang, Yulong Wang, and Haifeng Sun. Multi-context integrated deep neural network model for next location prediction. *IEEE access*, 6:21980–21990, 2018.

[9] Sara Rodriguez, Yanira de Paz, Javier Bajo, and Juan M Corchado. Social-based planning model for multiagent systems. *Expert Systems with Applications*, 38(10):13005–13023, 2011.

[10] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[11] Karthik Karur, Nitin Sharma, Chinmay Dharmatti, and Joshua E Siegel. A survey of path planning algorithms for mobile robots. *Vehicles*, 3(3):448–468, 2021.

[12] Hongqi Li, Peisi Zhong, Li Liu, Xiao Wang, Mei Liu, and Jie Yuan. Robot dynamic path planning based on prioritized experience replay and lstm network. *IEEE Access*, 2025.

[13] Longyan Xu, Mao Xi, Ren Gao, Ziheng Ye, and Zaihan He. Dynamic path planning of uav with least inflection point based on adaptive neighborhood a* algorithm and multi-strategy fusion. *Scientific Reports*, 15(1):8563, 2025.

APPENDIX

A. Safety Factor Computation

```
def compute_safety_score(frame, env_params, decay_factor=0.9):
    with ThreadPoolExecutor() as executor:
        faces, poses, scene = executor.map(lambda f: f.result(), [
            executor.submit(deepseek.get_recognized_persons, frame),
            executor.submit(openpose.analyze, frame),
            executor.submit(scene_parser.parse, frame)
        ])
    t_decay = np.exp(-decay_factor * (current_time - last_observed_time))
    spatial_w = gaussian_kernel(3)

    safety = sum(spatial_w[geodesic_distance(p.position, user.position)]
                 * t_decay[p.id] * (1 + pose_risk_assessment(poses))
                 * (env_params.lighting*0.2 + (1-env_params.crowd_density)*0.5)
                 for p in faces)

    return SafetyScore(
        min(safety / (scene.max_capacity * SAFETY_SCALAR), 1.0),
        risk_zones=detect_risk_clusters(faces),
        confidence=calc_confidence(scene)
    )
```

B. Path Planning Module

```
def compute_new_cost(topology_graph, safety_score,
                     mobility_constraints, adaptive_lambda,
                     current_pos, current_pose, goal_pos):

    corridor_width = np.clip(safety_score.normalized_score, 0.5, 2.0)
    safety_corridor = create_corridor(
        safety_score.risk_zones,
        width=corridor_width,
        soft_margin=0.3
    )

    risk_penalty = 1 / (safety_score.confidence + 1e-5)

    for u, v, data in topology_graph.edges(data=True):
        base_cost = data['length']

        motion_cost = calc_motion_cost(
            current_pose,
            topology_graph.nodes[v]['pose'],
            mobility_constraints
        )

        adaptive_cost = (adaptive_lambda.update(base_cost) *
                         (risk_penalty + motion_cost) /
                         safety_corridor.get_weight(u, v))

        topology_graph[u][v]['adaptive_cost'] = adaptive_cost

    return nx.astar_path(
        topology_graph,
        source=current_pos,
        target=goal_pos,
        heuristic=hybrid_heuristic,
        weight='adaptive_cost'
    )
```

C. Heatmap Construction

```
def construct_heatmap(sensor_stream, map_config, kalman_params):
    kf = UnscentedKalmanFilter(
        state_dim=map_config.resolution[0] * map_config.resolution[1],
        process_noise=kalman_params.Q,
        measurement_noise=kalman_params.R
    )
    window_size = 10
    for window in sliding_window(sensor_stream, window_size):
        aligned_data = pointcloud_align(window, map_config)
        grid_count = probabilistic_voxelize(
            aligned_data,
            sigma=0.1
        )
        kf.predict()
        kf.update(grid_count.flatten())
        decay_mask = time_decay_field(window.timestamps)
        kf.state *= decay_mask.reshape(-1)
    return BayesianHeatmap(
        mean=kf.state.reshape(map_config.resolution),
        variance=kf.P.diagonal().reshape(map_config.resolution),
        update_time=current_time()
    )
```

D. Path Selection Strategy

```
class PathSelector:
    def __init__(self, path_lib, config):
        self.strategies = {
            'safety': SafetyFirstStrategy(),
            'efficiency': ShortestPathStrategy(),
            'hybrid': MOEAStrategy(3)
        }
        self.validator = PathValidator(
            config.max_curvature, config.robot_width * 1.2
        )
        self.horizon = RecedingHorizon(5, 0.7)

    def select_optimal_path(self, state, mode='adaptive'):
        strategy = self._adapt_strategy(state.safety_level) if mode == 'adaptive' else self.strategies[mode]
        candidate_paths = [
            p for t in self.horizon.time_steps
            for p in strategy.generate(state.projection(t))
            if self.validator.validate(p)
        ]
        return MCDM(candidate_paths).select([0.4, 0.3, 0.3], 'TOPSIS')

    def _adapt_strategy(self, safety_level):
        return self.strategies['safety' if safety_level < 0.3
            else 'hybrid' if safety_level < 0.7 else 'efficiency']
```