MSSV: 23521555

Họ và tên: Huỳnh Diên Thục

Final Project: Decision support system for students facing elevator congestion

Background and Context Paragraph

For my final project, I am assuming the role of a system developer tasked with designing a decision support system to address elevator congestion at the University of Information Technology, Vietnam National University - Ho Chi Minh City (VNUHCM). During peak hours, particularly from 7:30-8:00 AM, the elevator waiting area on the ground floor of Building B experiences severe congestion, with students often waiting 5-10 minutes to access elevators. This delay directly impacts students' ability to attend classes on time, have breakfast, and engage with early class content, leading to stress, reduced academic performance, and a diminished campus experience. The system aims to provide real-time notifications to students about queue status, helping them decide whether to wait for the elevator or take alternative routes like the stairs, while also reducing congestion in the waiting area.

Project Justification - Iteration 1

Problem Identification

The problem was identified through observations and reports of elevator congestion at the ground floor of Building B, University of Information Technology, VNUHCM, particularly during peak hours (7:30-8:00 AM). Students face delays of 5-10 minutes, impacting their ability to attend classes on time, have breakfast, and engage with early class content, which leads to stress and reduced academic performance. This issue was recognized by

analyzing the frequency and impact of congestion, as well as the need for a system to provide real-time information to help students decide whether to wait for the elevator or use alternative routes like stairs. The problem was further refined by considering the technical capabilities of RGB-D cameras and the elevator's specifications, ensuring the system could address the issue effectively.

Decomposition

To address the elevator congestion problem, it was broken down into manageable components: (1) detecting and counting people in the elevator waiting area and inside the elevator using RGB-D cameras, (2) estimating the occupied area to assess elevator capacity, (3) processing real-time video and depth data to determine queue status, and (4) delivering personalized notifications ("Full," "Available," or "Possibly Full") to students. Decomposition helped by isolating specific tasks, such as human detection and capacity estimation, which allowed for targeted development of algorithms and system modules. For example, separating the task of processing depth maps from notification delivery enabled efficient real-time calculations and ensured the system could handle each component within the 2-second response time requirement.

Pattern Recognition

Patterns were identified in the data collected from RGB-D cameras, particularly in the behavior of the queue and elevator occupancy. Analysis of video and depth data revealed recurring patterns, such as peak congestion during specific times (7:30-8:00 AM), consistent queue formation in the waiting area, and predictable elevator capacity limits based on the number of people and occupied area. These patterns helped in designing algorithms to predict queue status for the next elevator trip. For instance, recognizing that the queue often reaches maximum capacity when the elevator is full and no occupants exit allowed the system to issue "Possibly Full" notifications accurately, helping students make informed decisions about waiting or taking the stairs.

Abstraction

Abstraction was applied by focusing on essential information needed to solve the congestion problem while ignoring irrelevant details. The system abstracts complex video data into key metrics: the number of people in the queue, the occupied area in the elevator, and the elevator’s maximum capacity. By filtering out unnecessary details, such as facial recognition or behavioral analysis (explicitly out-of-scope), the system concentrates on delivering real-time queue status notifications. This abstraction simplified the development process, ensuring the system could process data quickly (within 2 seconds) and provide clear, user-friendly outputs ("Full," "Available," "Possibly Full") that align with the functional requirements and support student decision-making.

Project Justification

Iteration 2

Problem Identification

In the second iteration, the problem was further refined to address specific challenges in real-time processing and user notification accuracy. The initial system design revealed that occasional delays in processing video and depth data from RGB-D cameras could exceed the 2-second response time requirement, particularly during peak congestion periods. Additionally, feedback from a preliminary user survey indicated that some students found notifications unclear when the queue status changed rapidly (e.g., from "Available" to "Full" as new students joined). The problem was thus redefined to focus on optimizing processing speed and enhancing notification clarity to ensure students receive timely and accurate information to decide between waiting for the elevator or using the stairs.

Decomposition

The problem was broken down into sub-components to address the identified issues: (1) optimizing the processing pipeline for real-time video and depth data analysis, (2) improving human detection algorithms to handle crowded scenes, (3) refining the notification system to account for rapid queue status changes, and (4) implementing a backup mode to handle system errors (e.g., camera disconnections). Decomposition allowed for targeted improvements, such as parallelizing data processing tasks to meet the 2-second response time and designing a dynamic notification update mechanism to reflect real-time changes in queue status. This modular approach ensured that each issue could be tackled independently while maintaining system coherence.

Pattern Recognition

Analysis of additional data from RGB-D cameras revealed patterns in queue dynamics, such as frequent fluctuations in the number of people in the waiting area during peak hours and consistent overcrowding inside the elevator when no passengers exited. These patterns informed the development of a predictive model that anticipates queue status changes based on the rate of new arrivals and elevator occupancy trends. For example, recognizing that a sudden influx of students often led to a "Possibly Full" status within 30 seconds helped the system prioritize preemptive notifications, reducing confusion and enabling students to make quicker decisions about alternative routes.

Abstraction

Abstraction was applied by focusing on critical data points—queue length, elevator occupancy, and rate of student arrivals—while ignoring extraneous details like individual

student identities or non-human objects in the scene. This iteration abstracted the complex dynamics of queue fluctuations into a simplified state machine with three states ("Full," "Available," "Possibly Full") updated based on real-time data thresholds. By abstracting away irrelevant environmental factors (e.g., background noise or lighting variations), the system improved processing efficiency and ensured notifications remained clear and actionable, aligning with the requirement for an intuitive, user-friendly interface.

Iteration 3

Problem Identification
The third iteration focused on scalability and privacy concerns raised during system testing. Initial deployments showed that the system struggled to process multiple video streams simultaneously when extended to other buildings with similar congestion issues. Additionally, students expressed concerns about the privacy of video data collected by RGB-D cameras, fearing potential misuse despite assurances of encryption. The problem was redefined to enhance the system's ability to handle multiple data streams across large buildings while ensuring compliance with privacy standards, such as ISO/IEC 27001, to build user trust and meet legal requirements.

Decomposition
The problem was decomposed into: (1) developing a scalable architecture to process multiple video and depth data streams concurrently, (2) implementing robust encryption and data deletion protocols to protect user privacy, (3) designing a user consent mechanism to ensure compliance with privacy regulations, and (4) creating an error-handling system to maintain service continuity across multiple locations. Decomposition allowed for focused development, such as adopting cloud-based processing for scalability and integrating secure data-handling modules to address privacy concerns, ensuring the system could expand without compromising performance or user trust.

Pattern Recognition
Patterns were identified in the system's performance across different buildings, revealing that processing bottlenecks occurred when multiple cameras streamed high-resolution (1080p, 30 FPS) video simultaneously. Additionally, privacy concerns followed a pattern where students were more likely to consent to data collection if informed about data deletion timelines and encryption measures. These patterns guided the development of a load-balancing algorithm to distribute processing tasks across servers and a transparent consent interface that clearly communicated data usage policies, improving both scalability and user acceptance.

Abstraction

Abstraction was applied by focusing on high-level system requirements—simultaneous multi-stream processing and privacy compliance—while ignoring low-level details like specific camera models or minor variations in building layouts. The system abstracted video stream processing into a generalized pipeline that could handle varying numbers of cameras and abstracted privacy requirements into a standardized protocol for encryption and data retention. This approach simplified the scaling process, ensuring the system could be deployed across multiple buildings while maintaining compliance with international security standards and delivering consistent, real-time notifications to students.

// Algorithm: Elevator Congestion Decision Support System // Purpose: Process real-time RGB-D camera data to manage elevator queue and provide notifications // Inputs: Real-time video and depth data from RGB-D cameras (waiting area and inside elevator), // elevator specifications (max capacity in people, area in m^2), // coordinate data for queue and waiting area detection // Outputs: Notifications ("Full", "Available", "Possibly Full") to students in real-time // Constraints: Process within 2 seconds, handle 1080p video at 30 FPS, detect only humans

BEGIN ALGORITHM

1. Initialize System
    - Connect to RGB-D cameras in elevator waiting area and inside elevator
    - Load elevator specifications: max_capacity (people), elevator_area (m^2)
    - Define queue_area_coordinates and waiting_area_coordinates from camera data
    - Initialize notification system for real-time updates
    - Set error_handling_mode to backup if connection fails
2. Real-Time Data Acquisition WHILE system is running:
    - Capture video_frame_waiting from waiting area camera (1080p, 30 FPS)
    - Capture video_frame_elevator from elevator camera (1080p, 30 FPS)
    - Capture depth_map_waiting from waiting area camera

- Capture depth_map_elevator from elevator camera IF camera connection fails:
    - Trigger error_handling_mode
    - Alert system administrator
    - Switch to backup mode within 2 seconds ENDIF

3. Human Detection and Counting
    - Process video_frame_waiting and depth_map_waiting:
        - Apply human detection algorithm to identify people in queue_area_coordinates
        - Count num_people_queue (number of people in queue)
    - Process video_frame_elevator and depth_map_elevator:
        - Apply human detection algorithm to identify people inside elevator
        - Count num_people_elevator (number of people in elevator)
    - Filter non-human objects using depth data (ignore objects outside human size range)
4. Capacity Estimation
    - Calculate occupied_area_elevator using depth_map_elevator:
        - Sum pixel distances to estimate area occupied by num_people_elevator (m^2)
    - Compare occupied_area_elevator with elevator_area
    - Calculate available_capacity = max_capacity - num_people_elevator
    - Estimate queue_capacity = num_people_queue that can fit in available_capacity
        - Use average human area (e.g., 0.5 m^2 per person) for estimation
5. Queue Status Prediction
    - Initialize notification_status as "Available"
    - IF num_people_elevator >= max_capacity:

        - Set notification_status to "Full"
    - ELSE IF (num_people_elevator + num_people_queue) >= max_capacity AND no_exit_detected:
        - Set notification_status to "Possibly Full"
    - ELSE:
        - Set notification_status to "Available"
    - ENDIF
    - no_exit_detected = TRUE if no passengers exit elevator in current cycle
6. Notification Delivery
    - Send notification_status to display interface for students in queue_area_coordinates
    - Ensure notification is updated within 2 seconds of new person entering queue
    - Format notification for clarity (e.g., "Full: Please use stairs", "Available: Space for X people")
    - IF queue status changes rapidly (within 30 seconds):
        - Update notification dynamically to reflect latest status
7. Privacy and Data Management
    - Encrypt video_frame_waiting, video_frame_elevator, depth_map_waiting, depth_map_elevator
    - Store data temporarily for processing (delete after 24 hours)
    - Ensure compliance with ISO/IEC 27001 standards
    - Obtain user consent for data collection before processing
8. Scalability and Error Handling
    - IF multiple video streams are detected (from other buildings):
        - Distribute processing tasks across servers using load_balancing_algorithm
    - IF system error occurs (e.g., module disconnection):

        - Switch to backup_mode within 2 seconds
        - Log error for administrator review
    - Update system status continuously to ensure real-time performance
9. Loop Control
    - Repeat steps 2-8 every frame (30 FPS) to ensure real-time processing
    - IF system shutdown is requested:
        - Save system state
        - Disconnect cameras and clear temporary data
        - Exit ENDIF

END ALGORITHM