P-ISSN: 2987-7261
E-ISSN: 2987-7253

**Jurnal Riset Ilmu Teknik**
Journal homepage: https://jurnaljepip.com/index.php/jurit
**Vol. 4, No. 3, pp. 204-212, 2026**
**DOI: doi.org/10.59976/jurit.v4i3.402**

JURIT

# Feasibility of Crowd-Aware AR Indoor Navigation Using YOLOv8

**Evianita Dewi Fajrianti[1]*, Amma Liesvarastranta Haz[2], Aris Kurniawan[3], Andharu Raffi[4], Hadi Muhammad Hafiiz Hilmi[5], Saniyatul Mawaddah[6], Nirwana Haidar Hari[7]**

[1,2]Human Centric Multimedia Research Laboratory, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia
[3,4,5,6,7]Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia
*Corresponding Author: evianita08@pens.ac.id

**ABSTRACT**

*Augmented reality (AR) indoor navigation provides visual guidance, while crowd-responsive route selection requires current crowd information to be connected to route planning. This study evaluates the integration of camera-based crowd monitoring with AR route adaptation and measures its response time on local infrastructure. The proposed system uses YOLOv8s to detect people, classifies crowd levels from person counts, and uses Unity NavMesh to compute alternative routes displayed through AR on a smartphone. The integration extends static routing by using crowd levels estimated from camera feeds to temporarily exclude highly crowded passages from route planning and update the displayed AR route when an alternative path is available. Evaluation covered detection, crowd-level classification, rerouting, response time, and a static–dynamic routing comparison. Detection mAP@0.5 was 91.92%, and crowd-level classification accuracy was 90.67% across 75 observations. Rerouting succeeded in all 30 controlled events, with an average response time of 2.772 s from the first classification of a zone as highly crowded to the display of the updated AR route. The suitability of this response time for hospital navigation requires further evaluation. In 30 paired trials, dynamic routing avoided the designated crowded zones but increased mean path length and navigation time in all three zones. Testing was limited to one hospital floor with three monitored zones. User comfort, accessibility, and performance in larger buildings require further evaluation.*

***Keywords***: *AR Indoor Navigation, Dynamic Routing, Crowd-Level Monitoring, YoloV8s, NavMesh*

*This is an open-access article under the CC-BY license*

*Manuscript received 23 Agust, 2026; revised 17 September, 2026; accepted 21 October, 2026.*

## INTRODUCTION

Indoor positioning and navigation have been studied using a range of technologies and techniques [1], [2], [3], [4]. Indoor navigation systems are increasingly used in large multi-story buildings such as hospitals, universities, and shopping centers to assist users in reaching their destinations efficiently [5], [6] . Augmented reality (AR) indoor navigation presents directional cues within the user's camera view to guide movement through a building [7], [8]. Accurate localization and AR route visualization alone do not ensure that route selection responds to changing crowd conditions. This requires occupancy information to be connected to the routing process. Parnass et al. [9] reported substantial fluctuations in emergency-department crowding. This variability motivates investigating how crowd information can inform navigation routes. However, at the selected hospital the effects of crowding on travel time and wayfinding difficulty were not measured. This study evaluates the technical integration of crowd monitoring and AR route adaptation.

Previous INSUS studies implemented smartphone-based indoor navigation [10], [11] and investigated location recovery using QR codes and recognition of text on building signs [12], [13]. In particular, Fajrianti et al. [12] combined YOLOv8 sign detection and PaddleOCR text recognition to recover the user's position from existing building signs. These approaches address localization reliability, whereas the present study examines how crowd conditions can influence route accessibility after the user's position has been determined.

Congestion-aware AR routing has also been investigated. Zhang et al. [14] aggregated user locations from AR devices into a population density map and employed a variant of A* to select congestion-aware evacuation routes. Their approach focuses on evacuation and derives crowd information from participating users' devices. The present study instead obtains zone-level person counts from fixed cameras and uses crowd-level classifications to activate NavMesh obstacles and update smartphone AR routes. These studies establish localization recovery and congestion-aware evacuation, but do not establish the technical performance of the fixed-camera-to-NavMesh integration evaluated here. The specific question addressed in this study is whether person-count-based crowd classifications from fixed cameras can be translated into changes in zone accessibility and corresponding smartphone AR route updates within the tested hospital configuration. The evaluation examines detection performance, crowd-level classification accuracy, rerouting success, response time, and differences in path length and navigation time between static and dynamic routing.

YOLOv8-based person detection, a CNN-based model, has been applied to real-time people counting [15], [16]. Unity NavMesh supports route adaptation through obstacle carving, which changes the available navigation space [17]. These capabilities provide a basis for connecting camera-derived crowd levels to route updates. In this study, occupancy analysis runs on a local detection server, while route adaptation and AR visualization run on the smartphone client.

The main contribution is the implementation and technical evaluation of a pipeline connecting fixed-camera crowd monitoring to smartphone AR route updates. A navigation bridge translates zone-level classifications into NavMesh obstacle-state changes through consecutive-state checks. This integration enables monitored passages to be excluded from or reopened to route planning as crowd levels change.

## MATERIALS AND METHODS

The process begins with IP cameras streaming frames to the Flask detection server. YOLOv8s detects people in each monitored zone, and the detected count determines its crowd-level category. The server provides these categories through a REST API, which the Unity client polls every second. After two consecutive High observations, the client activates the corresponding NavMesh obstacle and recomputes the route to avoid the blocked zone. The updated route is displayed as AR directional arrows. After three consecutive non-High observations, the obstacle is deactivated and the route is recomputed with the reopened passage available for path planning. Figure 1 summarizes how camera-based crowd detection leads to obstacle-state changes and updated AR navigation routes.

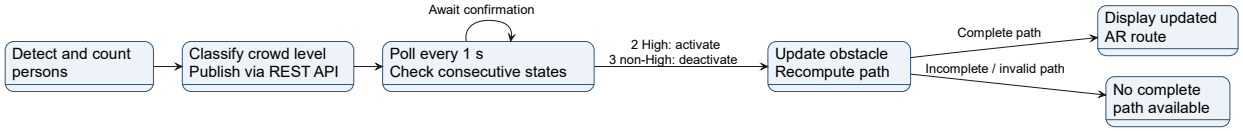



*Figure 1. Workflow of crowd detection, consecutive-state checks, and AR route updates.*

### Camera Acquisition and YOLOv8 Detection

IP cameras are mounted at fixed vantage points covering each defined building zone. The detection server, implemented as a Python Flask application, receives MJPEG frame streams from each camera. The detection application processes frames in temporary buffers and does not save the streams as video recordings. Buffered frames are replaced as new frames are processed. For each frame, YOLOv8s performs bounding box inference on the COCO 'person' class (index 0) and records the detected person count as the zone occupancy estimate. Each camera counts people in its assigned zone. The zone classification controls the corresponding NavMesh obstacle. Counts from different cameras are processed separately and are not summed. No cross-camera identity matching is used. Three cameras monitor separate zones on the fourth floor. Camera 1 monitors the reception area in front of the elevator, covering the passage between the east and west buildings. Camera 2 monitors the west-building waiting-area corridor, covering the passage in front of the seating area toward the west elevator and outpatient clinics. Camera 3 monitors the corresponding waiting-area corridor in the east building, covering the passage toward the east elevator and outpatient clinics. Each monitored zone is linked to a NavMesh obstacle for the corresponding passage. When activated, the obstacle excludes that passage from route computation. Figure 2 shows the detection server, data exchange, and AR navigation components.

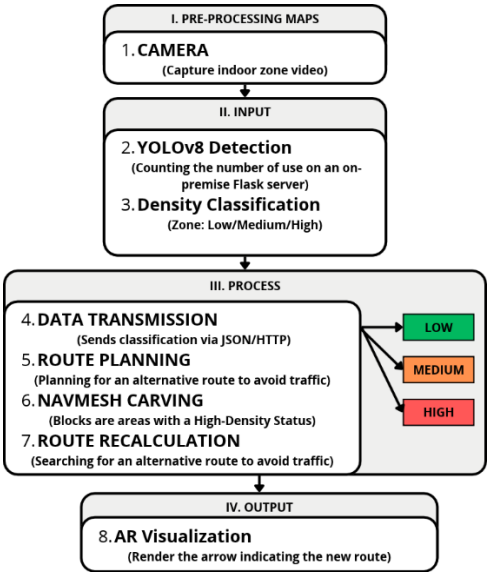



*Figure 2. System architecture of the proposed crowd-aware AR indoor navigation.*

YOLOv8 uses an anchor-free split detection head and provides pretrained models for object detection [18]. Person counting provides the zone-level input used by the crowd-classification and routing components. The system uses Ultralytics YOLOv8s pretrained weights (yolov8s.pt) to detect the COCO person class (class 0). Inference uses an input size of 640 × 640 pixels and a confidence threshold of 0.85. Processing runs on an NVIDIA GeForce GTX 1650 GPU with 4 GB VRAM. Person-detection performance was evaluated using 75 randomly selected images. Persons in these images were manually annotated with bounding boxes, which served as the reference for calculating precision, recall, F1 score, and mean average precision.

**Crowd-Level Classification Thresholds**

Zone crowd levels are classified using detected person counts, without normalization by zone area. Counts provide an operational input to the routing decision for each predefined zone. Table 1 defines the categories and corresponding system actions. Identical counts may represent different crowd conditions in differently sized zones. The thresholds are therefore operational rules for the tested configuration; their suitability for other areas requires calibration against zone dimensions and observed pedestrian conditions.

*Table 1. Crowd Level Classification Thresholds*

| Crowd Level | Person Count | Explanation |
|---|---|---|
| Low | 0-2 | No Action |
| Medium | 3-5 | Advisory Flag Only |
| High | >5 | NavMesh block + reroute |

Crowd-level classification was evaluated using 75 randomly selected frame observations, with 25 from each camera zone. Each observation had a manual person count as the reference. The manual and YOLO counts were converted into crowd-level categories using the thresholds in Table 1. Agreement between the reference and predicted categories was evaluated using a confusion matrix and classification metrics.

**Navigation Engine and AR Visualization**

The AR navigation client is built in Unity. Spatial localization is performed using the Multiset AI SDK, which maps the user's 6-DOF pose within the pre-mapped building coordinate frame. Route computation uses Unity's built-in NavMesh pathfinding to determine the optimal walkable path from the user's current position to the selected destination POI. Routes are visualized as AR directional arrows anchored to NavMesh waypoints and rendered in the smartphone camera view, updated at 15–20 Hz. Multi-floor routing is handled through dedicated staircase and elevator waypoints.

**Dynamic Obstacle Avoidance via NavMesh Carving**

NavMesh represents the walkable floor surface as a polygon mesh generated from the building geometry. Each monitored zone is linked to a NavMeshObstacle component. When the High-state activation requirement is met, the navigation client activates the corresponding obstacle with carving enabled. This excludes the passage from the walkable surface, and the client recomputes the route to avoid the blocked area [17] . The updated route is then displayed in the smartphone AR interface.

When the Low or Medium state satisfies the deactivation requirement, the obstacle is deactivated and the passage becomes available for route computation again. The client recomputes the path and selects a shorter route if one is available. If all routes to the destination are blocked, no complete path can be computed. Successful rerouting therefore requires an available alternative path. The all-routes-blocked condition was not evaluated in the reported experiments.

**Rerouting Trigger Logic (The Navigation Bridge)**

The Flask detection server processes camera frames and provides each zone's person count and crowd level through a REST API. The Unity AR client polls these data every second through the navigation bridge. The bridge applies consecutive-state checks and controls the corresponding NavMesh obstacle. When the obstacle state changes, the client recomputes the path and updates the AR route display.

A zone obstacle is activated after two consecutive High observations (K = 2) and deactivated after three consecutive Low or Medium observations (M = 3). Each obstacle-state change triggers route recomputation. With one-second polling, activation requires one further polling interval after the first High observation, while deactivation requires two further intervals after the first non-High observation. A server-side classification change may also wait for the next client poll. Communication, NavMesh processing, and AR rendering contribute additional delay. These consecutive checks reduce route changes caused by brief crowd-level fluctuations.

## RESULTS

This section reports person-detection performance, crowd-level classification accuracy, rerouting outcomes, the static–dynamic routing comparison, and timing measurements. Experiments were conducted on the 4th floor of RSI A. Yani Surabaya, comprising three defined navigation zones covered by three IP cameras. The detection and classification server ran on a local machine equipped with an NVIDIA GTX 1650 (4GB VRAM) and 16GB RAM, while the AR navigation client was tested on a Samsung Galaxy A16 (Android 15) smartphone. Each subsection concludes with a discussion interpreting the significance of the findings and their implications for practical deployment.

**YOLOv8s People Detection Accuracy**

Table 2 presents person-detection performance evaluated against manually annotated bounding boxes, together with the recorded per-frame inference times. The mAP@0.5 of 91.92% and mAP@0.5:0.95 of 89.68% reflect robust bounding-box localization accuracy that holds up even under stricter intersection-over-union criteria, a property essential for reliably counting persons within spatially constrained corridor zones.

The closely matched Precision (92.55%) and Recall (92.47%) values show that the model maintains a well-balanced trade-off between minimizing false positives and false negatives. This balance matters operationally: too many false positives would inflate the estimated crowd level and trigger rerouting that isn't actually needed, while too many false negatives would understate zone occupancy and prevent rerouting from activating when it genuinely should. The resulting F1 score of 0.9251 captures this balance in a single quantitative measure.

*Table 2. YOLOv8s People Detection Performance*

| Metric | Achieved |
|---|---|
| mAP@0.5 | 91.92% |
| mAP@0.5:0.95 | 89.68% |
| Precision | 92.55% |
| Recall | 92.47% |
| F1 Score | 0.9251 |
| Avg. Inference Time (per frame) | 23.69 ms |
| Min. Inference Time (per frame) | 16.51 ms |
| Max. Inference Time (per frame) | 166.7 ms |

YOLOv8s inference averaged 23.69 ms per frame. This measures model inference rather than the complete navigation response, which also includes communication, consecutive-state checks, and route updates. Inference times ranged from 16.51 to 166.7 ms, indicating variation across the recorded measurements.

**Zone Crowd-Level Classification Accuracy**

Zone crowd-level classification was evaluated using 75 observations, with 25 observations from each camera zone. Manual person counts were converted into reference categories using the thresholds in Table 1, yielding 24 Low, 27 Medium, and 24 High observations. Figure 3 presents the confusion matrix. The system correctly classified 68 observations, achieving an overall accuracy of 90.67%. Seven classification errors occurred: one Low observation was predicted as Medium, two medium observations as Low, one medium observation as High, and three High observations

as Medium. These errors occurred when differences between manual and detected counts crossed a category boundary.

High-class precision and recall were 95.45% and 87.50%, respectively. The three missed High observations could delay rerouting activation, while the medium observation predicted as High could contribute to unnecessary activation. The effect on routing depends on whether these classifications persist through the consecutive-state checks. These results highlight the need for further evaluation of classification stability near the category boundaries.

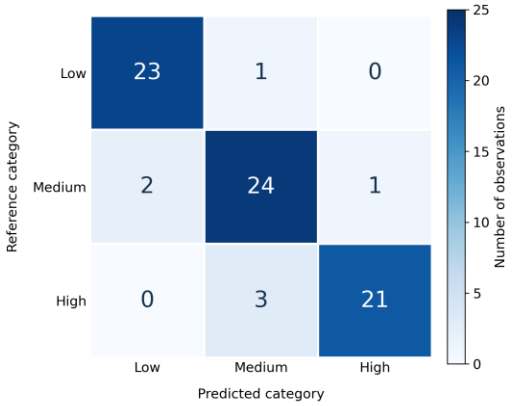



*Figure 3. Confusion matrix for zone crowd-level classification.*

**Dynamic Rerouting Trigger Evaluation**

Dynamic rerouting was evaluated over 30 controlled High-crowd-level trigger events, with ten events in each camera zone. A reroute was considered successful when NavMesh computed a complete alternative path that avoided the blocked zone and the path was displayed correctly in the smartphone AR interface.

All 30 events produced successful reroutes. Obstacle deactivation and complete-path computation after deactivation succeeded in all 30 events, while the original route was restored in 29 events. Table 3 summarizes the outcomes. These results support repeatable operation within the tested configuration but do not establish reliability across broader operating conditions.

*Table 3. Dynamic Rerouting Trigger Results*

| Metric | Value |
|---|---|
| Total High-crowd-level trigger events | 30 |
| Events per camera zone | 10 |
| Successful reroute | 30/30 (100.0%) |
| Failed reroute (no alt. path) | 0/30 (0.0%) |
| Successful obstacle deactivations | 30/30 (100%) |
| Complete path after obstacle deactivation | 30/30 (100%) |
| Return to the original route | 29/30 (96.7%) |
| Rerouting response time, mean ± SD | 2.772 ± 0.172 s |
| Minimum rerouting response time | 2.491 s |
| Maximum rerouting response time | 3.045 s |

After obstacle deactivation, complete paths were available in all 30 events. The original route was selected again in 29 events. In the remaining event, the system maintained a valid alternative route. This event was recorded as no return to the original route, although obstacle deactivation and complete-path computation succeeded. Across 30 events, the mean rerouting response time was 2.772 s (SD = 0.172 s), ranging from 2.491 to 3.045 s. This duration was measured from the first High classification to the display of the updated AR route and includes polling and consecutive-state checks. Figure 4 shows the original AR route and the turn-back instruction displayed after rerouting.

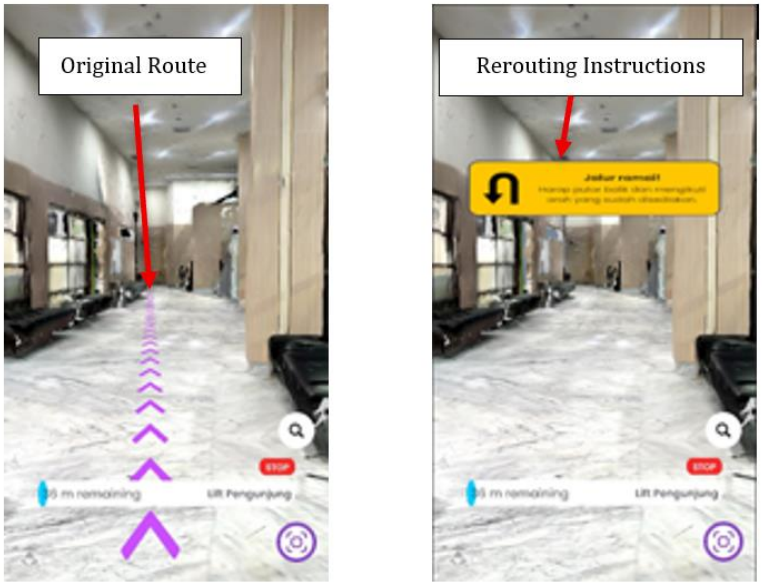



*Figure 4. Smartphone AR navigation views: (a) the original route indicated by purple directional arrows; (b) a turn-back instruction displayed after crowd-triggered rerouting.*

**Comparison with Static Routing**

Static and dynamic routing were compared in 30 paired trials, with ten pairs per camera zone. The navigation runs were performed by the authors. The trials assessed route completion, crowd-zone avoidance, path length, and navigation time. Patient comfort and usability were not evaluated. The same application was used with crowd-triggered routing disabled for the static baseline and enabled for the dynamic mode. Path length was obtained from NavMesh. Navigation time was measured from navigation start until arrival at the destination. Table 5 demonstrate the results on the static and dynamic comparison between east and west wing.

*Table 4. Static and Dynamic Routing Comparison*

| Camera | Pairs | Mean static path (m) | Mean dynamic path (m) | Mean static navigation time (s) | Mean dynamic navigation time (s) |
|---|---|---|---|---|---|
| Camera 1 | 10 | 61.4 | 65.1 | 201.7 | 267.7 |
| Camera 2 | 10 | 25.6 | 37.2 | 109.1 | 145.0 |
| Camera 3 | 10 | 26.2 | 37.8 | 101.2 | 139.4 |

Each mode produced complete paths in all 30 runs, giving 60 runs across the 30 paired trials. This indicates successful route computation, not improved travel performance. Dynamic routes avoided the High-crowd-level zone in all trials, whereas static routes crossed it. However, dynamic routing increased mean path length and navigation time in all three zones. The results demonstrate crowd avoidance, but not faster or shorter travel.

**Pipeline Stage Latency**

Pipeline-stage timing was recorded for the same 30 rerouting events. Table 5 reports the mean duration of each measured stage. Their mean durations sum to 390.41 ms, excluding the wait for consecutive High-state checks. This aggregate differs from the average rerouting response time of 2.772 s across 30 events, measured from the first High classification to the updated AR route display, including polling and hysteresis.

*Table 5. Pipeline Stage Latency Measurements*

| Pipeline Stage | Avg (ms) |
|---|---|
| Camera → YOLOv8 detection | 162.41 |
| Detection → Crowd-level classification | 0.83 |
| Classification → API response | 6.47 |
| API poll → NavMesh carve | 219.63 |
| NavMesh carve → AR route update | 1.07 |
| **Sum of stage means** | **390.41** |

The largest reported stage duration was API poll to NavMesh carve, averaging 219.63 ms. Camera acquisition through YOLOv8 detection averaged 162.41 ms. This stage covers more than model inference alone, which averaged 23.69 ms per frame. Crowd-level classification averaged 0.83 ms, and the final stage from NavMesh carve to AR route update averaged 1.07 ms. The final-stage value does not represent the full rerouting process. The mean rerouting response time includes polling and consecutive-state checks. Reduced polling intervals or event-driven communication may improve response time, but these changes were not tested.

## DISCUSSION

The evaluation supports the operation of crowd-aware rerouting within the tested configuration. Crowd-level classification achieved 90.67% accuracy across 75 observations, and all 30 controlled trigger events produced complete alternative paths displayed in AR. The mean rerouting response time was 2.772 s (SD = 0.172 s), including polling and consecutive-state checks. These results demonstrate functional operation, while the suitability of this response time for hospital navigation requires further evaluation.

Mapping camera zones to NavMesh obstacles allows crowd-state changes to affect the relevant passages. Hysteresis limits responses to brief classification fluctuations but introduces a trade-off between stability and responsiveness. The additional confirmation required for deactivation also delays reopening a passage after the crowd level falls.

Classification errors occurred at both the Low–Medium and Medium–High boundaries. Differences between manual and detected person counts caused seven observations to cross a category boundary. High-class precision was 95.45%, while recall was 87.50%. Missed High classifications could delay obstacle activation, and a medium observation classified as High could contribute to unnecessary activation if the error persists through the consecutive-state checks. Partial visibility or occlusion may contribute to counting errors, although their specific causes were not verified. The person-count thresholds are operational rules and require calibration for zones with different dimensions and occupancy patterns.

The latency results show that response time depends on both pipeline-stage durations and the waiting introduced by polling and consecutive-state checks. Camera acquisition and detection, together with the API-poll-to-NavMesh-carve stage, account for the largest recorded stage durations. Optimization therefore requires evaluating processing, communication, and state-confirmation delays together. Shorter polling intervals or event-driven communication remain possible improvements whose effects require experimental validation.

Previous INSUS research [12] evaluated location recovery through sign detection and text recognition. The present study examines the subsequent routing stage, where camera-derived crowd levels determine passage availability. Zhang et al. [14] investigated congestion-aware AR evacuation using crowd information aggregated from users' devices. Differences in crowd sensing, routing objectives, building layouts, and evaluation procedures limit direct numerical comparison with previous studies.

The paired static–dynamic comparison demonstrates the trade-off introduced by crowd avoidance. Across 30 paired trials, dynamic routing avoided the designated crowded zones but increased mean path length and navigation time in all three zones. Excluding a crowded passage can require a longer detour, so successful crowd avoidance does not necessarily improve travel efficiency. For elderly patients, wheelchair users, and people with limited mobility, these detours may increase physical effort. Route selection also needs to consider corridor width, step-free access, and elevator availability. User comfort and accessibility require evaluation with these groups.

In real buildings, the integration allows navigation guidance to respond to crowd conditions in monitored passages. A passage can be temporarily excluded when its crowd level is High and made available again after the level falls and the deactivation requirement is met. Hospital implementation requires suitable camera coverage, accurate zone-to-NavMesh mappings, and reliable communication between the detection server and navigation client. Camera, server, or network interruptions may prevent crowd-state updates. The detection application processes camera frames temporarily and does not retain video recordings. Access to live streams and the detection server still requires appropriate controls. These controls were not assessed in the present evaluation, so privacy protection requires further assessment before wider deployment.

The evaluation covered one floor and three camera-monitored zones. The 75 classification observations and 30 rerouting events provide evidence within this restricted setting. Larger buildings introduce additional camera streams, monitored zones, and routes across floors, which may increase processing load and response time. Further work should evaluate these scaling effects, improve classification near category boundaries, test failure handling, and assess route suitability with hospital users.

## CONCLUSION

This study evaluated an integration linking camera-derived crowd levels to AR route updates. Person detection achieved an mAP@0.5 of 91.92%, while crowd-level classification accuracy was 90.67% across 75 observations. All 30 controlled rerouting events succeeded, with a mean response time of 2.772 s from the first High classification to the updated AR route display. In 30 paired trials, dynamic routing avoided designated crowded zones but increased route length and navigation time. The findings are limited to one hospital floor, three monitored zones, and the reported sample sizes. Future work should prioritize response-time reduction, testing across larger layouts and failure scenarios, and evaluation of navigation performance, comfort, and accessibility with hospital users.

## ACKNOWLEDGMENT

The authors would like to thank Politeknik Elektronika Negeri Surabaya (PENS) for supporting this research through the Internal Research Grant 2026, under Contract No. 012/SP2H/PENS/PPK/2026. The authors also express their appreciation to RSI A. Yani Surabaya for providing the research location and supporting the implementation and evaluation of the proposed crowd-aware AR indoor navigation system.

## CONFLICT OF INTEREST

The authors declare that they have no known financial or personal conflicts of interest that could have influenced the research, analysis, or findings presented in this study.

## DATA AVAILABILITY STATEMENT

The data supporting this study include person-detection results, crowd-level classification records, rerouting measurements, and navigation performance data. These data may be requested from the corresponding author, subject to institutional approval and privacy requirements. Camera streams were processed temporarily and were not stored as video recordings to protect the privacy of hospital visitors.

## FUNDING

This research was supported by the Internal Research Grant of Politeknik Elektronika Negeri Surabaya (PENS) in 2026, under Contract No. 012/SP2H/PENS/PPK/2026.

## DECLARATION OF AI TOOL USAGE

No generative AI tools were used for field data collection. YOLOv8s was used as the computer vision method for detecting and counting people.

## REFERENCES

[1] W. C. S. S. Simões, G. S. Machado, A. M. A. Sales, M. M. de Lucena, N. Jazdi, and V. F. de Lucena, "A Review of Technologies and Techniques for Indoor Navigation Systems for the Visually Impaired," *Sensors*, vol. 20, no. 14, 2020, doi: 10.3390/s20143935.

[2] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007, doi: 10.1109/TSMCC.2007.905750.

[3] J. Drewlow, M. Dappen, and M. Lehmann, "Navigation with Augmented Reality in a Hospital," *Stud. Health Technol. Inform.*, vol. 292, pp. 111–114, 2022, doi: 10.3233/SHTI220335.

[4] B. C. Huang, J. Hsu, E. T. H. Chu, and H. M. Wu, "ARBIN: Augmented Reality Based Indoor Navigation System," *Sensors 2020, Vol. 20,* vol. 20, no. 20, pp. 1–20, Oct. 2020, doi: 10.3390/S20205890.

[5] H. Iftikhar, P. Shah, and Y. Luximon, "Human wayfinding behaviour and metrics in complex environments: a systematic literature review," *Archit. Sci. Rev.*, vol. 64, no. 5, pp. 452–463, Sep. 2021, doi: 10.1080/00038628.2020.1777386.

[6] E. D. Fajrianti, A. L. Haz, Y. A. Sari, S. Sukaridhoto, Z. M. Achmad, and R. P. N. Budiarti, "Comparative Analysis of Automated Cloud-Based Mapping and Manual 3D Modeling for AR-Based Indoor Navigation," *Journal of Imaging 2026, Vol. 12, Page 452*, vol. 12, no. 9, p. 452, Sep. 2026, doi: 10.3390/JIMAGING12090452.

[7] B. Rokesh Maran, L. Giridharan, and R. Krishnaveni, "Augmented Reality-based Indoor Navigation using Unity Engine," in *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, 2023, pp. 1696–1700. doi: 10.1109/ICSCSS57650.2023.10169855.

[8] Z. Qiu, A. Mostafavi, and S. Kalantari, "Use of augmented reality in human wayfinding: a systematic review," *Virtual Reality 2025 29:4*, vol. 29, no. 4, pp. 154-, Sep. 2025, doi: 10.1007/S10055-025-01226-W.

[9] G. Parnass, O. Levtzion-Korach, R. Peres, and M. Assaf, "Estimating emergency department crowding with stochastic population models," *PLoS One*, vol. 18, no. 12, p. e0295130, Dec. 2023, doi: 10.1371/JOURNAL.PONE.0295130.

[10] E. D. Fajrianti, A. L. Haz, N. Funabiki, and S. Sukaridhoto, "A Cross-Platform Implementation of Indoor Navigation System Using Unity and Smartphone INSUS," in *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 2023, pp. 249–254. doi: 10.1109/ICVEE59738.2023.10348217.

[11] E. D. Fajrianti *et al.*, "INSUS: Indoor Navigation System Using Unity and Smartphone for User Ambulation Assistance," *Information*, vol. 14, no. 7, 2023, doi: 10.3390/info14070359.

[12] E. D. Fajrianti, Y. Y. F. Panduman, N. Funabiki, A. L. Haz, K. C. Brata, and S. Sukaridhoto, "A User Location Reset Method through Object Recognition in Indoor Navigation System Using Unity and a Smartphone (INSUS)," *Network 2024, Vol. 4, Pages 295-312*, vol. 4, no. 3, pp. 295–312, Jul. 2024, doi: 10.3390/NETWORK4030014.

[13] E. D. Fajrianti, N. Funabiki, A. L. Haz, and S. Sukaridhoto, "A Proposal of OCR-based User Positioning Method in Indoor Navigation System Using Unity and Smartphone (INSUS)," in *Proceedings of the 2023 12th International Conference on Networks, Communication and Computing*, in ICNCC '23. New York, NY, USA: Association for Computing Machinery, 2024, pp. 99–105. doi: 10.1145/3638837.3638852.

[14] Z. Zhang, H. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, "Congestion-aware Evacuation Routing using Augmented Reality Devices," in *ICRA*, 2020.

[15] A. Elaoua, M. Nadour, L. Cherroun, and A. Elasri, "Real-Time People Counting System using YOLOv8 Object Detection," in *2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, 2023, pp. 1–5. doi: 10.1109/IC2EM59347.2023.10419684.

[16] A. Cortesi *et al.*, "Integration of Deep Learning into the IoT: A Survey of Techniques and Challenges for Real-World Applications," *Electronics 2023, Vol. 12, Page 4925*, vol. 12, no. 24, p. 4925, Dec. 2023, doi: 10.3390/ELECTRONICS12244925.

[17] Unity Technology, "Unity - Manual: NavMesh Obstacle." Accessed: Jul. 13, 2026. [Online]. Available: https://docs.unity3d.com/530/Documentation/Manual/class-NavMeshObstacle.html

[18] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics