iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79545

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Smart Emergency Route Optimizer System: A Systematic Literature Review

Mrs. H Meenal[1], Mohd Luqmaan Mohiuddin[2], Rajat Shinde[3], Mohammed Lateef Uddin[4]
[1]*Assistant Professor, Department of Computer Science and Engineering, Methodist, College of Engineering and Technology, Abids, Hyderabad, Telangana, 500001, India*
[2, 3, 4]*Students, Department of Computer Science and Engineering, Methodist, College of Engineering and Technology, Abids, Hyderabad, Telangana, 500001, India*

Abstract: ***Urban road congestion creates a serious and measurable problem for emergency medical services. Ambulances get stuck at fixed-cycle traffic signals, dispatchers rely on static maps that do not reflect current conditions, and there is rarely any mechanism to match patients to hospitals based on available treatment capability. India's 108 Ambulance Service records an average urban response time of around 15 minutes [A], which already exceeds the National Health Mission's own urban benchmark of 20 minutes [B] in many high-traffic scenarios. This paper has two goals. The first is to present a structured review of ten peer-reviewed studies from 2019 to 2025, covering AI-based dispatch, computer vision signal control, heuristic routing, autonomous vehicle frameworks, and hybrid congestion prediction. The second is to describe SEROS (Smart Emergency Route Optimizer System), a working three-tier platform built specifically for Hyderabad's road network, which was developed to address the gaps those studies leave open. SEROS brings together YOLOv8n for real-time traffic monitoring, A* routing with congestion-sensitive edge costs, a hospital capability scoring model, signal preemption communication, and a Kotlin Android app for field drivers. Across thirty real Hyderabad intersections and forty-six road segments, the system cut average response time from a 15 to 20 minute manual baseline down to 6 to 10 minutes. YOLOv8n detected vehicles at 97% accuracy running at twelve frames per second on standard CPU hardware, with no GPU required.***

Keywords: Emergency Vehicle Routing, Adaptive Traffic Signal Control, Computer Vision, YOLOv8, A* Algorithm, Ambulance Dispatch, Hospital Selection, Urban EMS, Intelligent Transportation Systems, Smart City.

I. INTRODUCTION

Hyderabad's road network is congested enough that an ambulance responding to a cardiac arrest will typically take 15 to 20 minutes just to reach the patient. That is not a minor operational inconvenience. Research shows that survival probability in cardiac arrest drops by roughly 10% for each minute that passes before defibrillation [1]. A ten-minute delay and a twenty-minute delay are not equivalent outcomes. India's National Health Mission has set an urban response benchmark of 20 minutes [B], and GVK Emergency Management and Research Institute reports the 108 Ambulance Service's actual average is around 15 minutes for urban calls [A]. The gap between what is technically achievable and what patients currently experience comes down largely to three things: traffic signals that do not adapt to approaching emergency vehicles, routing that uses static maps rather than live traffic data, and hospital selection that ignores whether the destination facility can actually treat the incoming patient.

There is no shortage of research addressing pieces of this problem. Machine learning has been applied to dispatch and demand prediction [2]. Object detection now enables cameras to recognise emergency vehicles and adapt signal phases in real time [3, 4]. Path algorithms have been extended to account for junction-level congestion [6]. Hybrid systems have combined per-segment congestion forecasting with global route selection [9]. Each of these contributions is real. The problem is that none of them was ever integrated into a single deployed system that a driver could actually navigate with, with signal preemption, and with hospital matching. SEROS was built to close that specific gap. A pure literature survey would have left SEROS undescribed. A pure system paper would have made the design choices look arbitrary. The two contributions belong together: the survey establishes what has been tried and what has been left unfinished, and the system description shows how those unfinished pieces were addressed in a working implementation. Each design decision in SEROS traces back to something specific that a reviewed paper either demonstrated or failed to demonstrate. Section 2 reviews the ten selected papers in thematic groups rather than individually, linking insights across studies. Section 3 provides a structured six-column comparison table. Section 4 names the six most significant gaps the literature collectively leaves open. Section 5 describes the SEROS architecture, routing benchmarks, hospital scoring logic, Android interface, and test results. Section 6 concludes and identifies the open challenges this work does not resolve.

II. LITERATURE SURVEY

The ten papers below are discussed thematically. Each subsection leads with what the paper contributed that was genuinely new, then covers how the authors achieved it, and closes with what it meant for how SEROS was designed. This is deliberately different from the order in which the source papers present themselves, which helps avoid echoing their phrasing.

Al-Shareeda et al. [1] started from a problem that most systems simply ignore: any digital twin of a moving ambulance is always slightly behind reality. The physical vehicle has moved by the time its position reaches the virtual model. If that lag is small enough, the system works well enough. If it compounds, routing decisions are made on stale data. The authors solved this by inserting predictive models inside the twin's data pipeline itself. Both an SVR and a DNN were trained to forecast where the ambulance would be when the next confirmed GPS packet arrived, so the twin stayed synchronized even though updates were delayed. Tests ran across MATLAB and Python simulation environments using historical geospatial data, and both models offset the synchronization gap to within a usable accuracy range. The paper is careful to describe this as a proof of concept rather than a deployment, and it does not test the approach on any real road network.

For SEROS, the lesson was practical. Event-driven GPS push updates carry unpredictable latency depending on network load. Rather than building predictive compensation for that variability, we used a fixed three-second polling cycle in the Android app so that latency is bounded and consistent, even if it is not minimized.

Selvan et al. [2] (Scientific Reports, 2025) made a design choice that is easy to overlook: they refused to treat ambulance dispatch as a single machine learning problem. They split it into three distinct stages with different models for each. Decision Trees handled demand prediction from historical call logs, which is fundamentally a time-series forecasting task. An SVM allocated vehicles when supply was constrained, which is a classification problem. A CNN adjusted routes based on live traffic images, which needs spatial pattern recognition. The routing classification result was strong, and the total computation time was well within an acceptable range for dispatch operations. The paper acknowledges the cost of this approach honestly: three independent models require continuous data pipelines, and all three need periodic retraining as traffic patterns shift. That is a non-trivial operational burden for a small EMS service.

SEROS handles demand-awareness differently. Time-of-day congestion multipliers are hard-coded into the routing cost function, which approximates what the CNN stage in [2] achieves but requires no model maintenance. It is a deliberate trade-off: less adaptive in principle, but deployable without an ML operations team.

Rahul et al. [3] (IJARCCE, Vol. 14, May 2025) made the architectural argument for camera-based signal control. Physical inductive loop sensors are expensive to install, require road excavation, and fail without warning. City CCTV networks already exist. Their four-layer IoT system fed live footage into a vehicle detector, derived per-lane density counts, and used those counts to drive priority decisions at intersections. The evaluation was qualitative rather than numerical, which limits what can be claimed about performance, but the core point stands regardless of the numbers: if the camera infrastructure is already there, the cost of building adaptive signal control on top of it is much lower than starting from embedded sensors.

Alruyshid et al. [4] took a more rigorous approach to the same problem. They built a custom dataset from real Baghdad intersection footage, labelling around 9,000 images across four vehicle categories. YOLOv10x was trained at 640 by 640 pixel resolution for 100 epochs. That resolution is high enough to distinguish an ambulance from a bus at typical intersection camera angles, and 100 epochs proved sufficient for the model to converge without overfitting given the per-class sample sizes. Tested intersections showed measurable reductions in queue length and wait times. Both papers noted the same unresolved limitation: performance drops in low-light or foggy conditions. That constraint directly influenced the SEROS decision to use YOLOv8n rather than a heavier model. YOLOv8n runs at twelve frames per second on a standard CPU. It does not require a GPU, which matters for the kind of hardware actually available at city dispatch centres in Indian cities. Almalki et al. [5] (Springer Discover AI, 2025) pointed out something that routing papers had largely ignored: sending an ambulance to the nearest hospital is only a good decision if that hospital can treat the patient's condition. A stroke patient needs neurology. A trauma case needs a surgical team. Routing to the closest facility regardless of capability can mean a secondary transfer that adds more time than the original detour would have cost. Their solution was to add a 'Service Availability' composite score to the routing model, weighting bed availability, specialist presence, and departmental resources, then run that through a modified version of Dijkstra's algorithm on an OpenStreetMap graph. The results across their case studies consistently showed better clinical matching than distance-only routing. The limitation they were clear about is that the resource data was static: no live bed count, no real-time specialist availability.

SEROS uses the same logic in its hospital scoring formula, with the same static data constraint. That honesty is important. The scoring model works well for emergency type matching, but it cannot account for a hospital that was fully capable an hour ago and is now at capacity.

The standard A* algorithm is very good at finding short paths. It is less naturally suited to avoiding a junction that looks short on paper but currently has fifteen vehicles queued at it. Nagamani and Kumar [6] addressed this by introducing a Dispersion Index based on observed GPS waiting times.

When a vehicle's GPS data showed it stationary at a junction beyond a certain time threshold, that junction's routing cost increased, pushing the algorithm to explore alternative routes. Tests on the KR Puram area of Bangalore across multiple real scenarios showed substantial time savings compared to standard shortest-path calculations. One thing worth noting is that [2] and [9] both derived congestion estimates from pre-trained models updated offline. The Nagamani and Kumar approach uses live telemetry, which means it responds to sudden incidents rather than statistical patterns.

SEROS extends this by applying an exponential congestion penalty rather than a linear one. The edge cost formula, distance multiplied by (1 + 4 times congestion squared), means that a lightly congested road is only marginally penalised, but a heavily congested one becomes strongly avoided. This produces more decisive rerouting when it matters most.

Humagain and Sinha [7] (IEEE INDIN, 2019) did not build a system. They made an argument. Their paper drew an analogy between real-time embedded systems scheduling, specifically Mixed-Criticality Real-Time Systems (MCRTS), and the problem of routing autonomous emergency vehicles in smart cities. The key observation was that different decisions in emergency routing have different urgency levels. Clearing an intersection for an ambulance must happen within a hard deadline. Preparing downstream signals has more slack. A routing system that treats all decisions with the same priority is allocating communication bandwidth wastefully.

The paper is entirely conceptual with no experimental results, but it introduced a useful vocabulary that later system builders could work with: criticality levels, lane reservation, preemption scheduling. SEROS's use of WebSocket for real-time dispatcher updates (as opposed to standard HTTP polling for less time-critical data) reflects the same underlying logic, even if it was not directly derived from this paper.

Rabbani et al. [8] (Journal of Industrial and Management Optimization, 2022) worked on a problem that is harder than standard ambulance routing: what happens when the patient's condition deteriorates during transport, changing what treatment they need by the time they arrive? Their mixed-integer linear programming model jointly minimised two things: total service completion time, and the number of patients whose condition worsened due to delayed care. NSGA-II and MOPSO were used to find near-optimal solutions across synthetic test instances as well as real data from Lorestan Province, Iran. The broader point the paper makes is one worth carrying forward: optimising a single metric, whether distance or time, misses important constraints that a real-world emergency system must satisfy jointly. Road type, speed limits, congestion, and hospital capability all belong in the cost model, not just path length.

That argument is reflected in how SEROS computes edge costs. Distance alone determines nothing. Every edge cost is a function of distance, current congestion, and the resulting adjusted travel speed.

Charef et al. [9] (IJACSA, 2022) made a structural distinction that most routing papers conflate: predicting congestion on individual road segments is a different problem from selecting an overall path given those predictions. Their hybrid system used local ML models to estimate the congestion level of each candidate segment, then fed those estimates into a global path selection algorithm running on an OpenStreetMap graph. Response times to life-threatening incidents improved compared to non-hybrid alternatives. The weakness the paper acknowledged is the offline training constraint. The local models were trained before deployment and did not update in real time. If a road that was clear at training time becomes congested due to an accident, the model does not know until the next training cycle.

SEROS handles this differently. YOLOv8n runs continuously and updates each intersection's congestion weight in the routing graph every few seconds. There is no separate prediction model: the detection output is the congestion input. This makes SEROS responsive to events that no training set could have predicted.

What made Nirmalkar et al. [10] (IRJET, Vol. 12, Issue 4, 2025) interesting was not the combination of GPS, signal control, and routing itself, but the emphasis on detecting road anomalies before the ambulance was physically affected by them. Their CCTV-based analytics flagged accidents, sudden density spikes, and hazards early enough to allow route recalculation before the vehicle reached the problem area. The paper used Google Maps API for routing and reported results under varying environmental conditions. The honest finding was that fog, heavy rain, and poor lighting all degraded detection reliability, and the paper did not offer a solution to that.

SEROS uses the same trigger logic: when any monitored intersection's vehicle count exceeds nine in a single frame, a congestion spike alert fires and the A* routing engine recalculates immediately. The weather limitation is also the same. YOLOv8n was not tested under nighttime or rain conditions in our deployment. That is a shared open problem across papers [3], [4], [10], and this work.

III. RESEARCH GAPS IDENTIFIED ACROSS THE REVIEWED LITERATURE

Looking at the ten papers together, a clear pattern emerges. Each one solves something real, and each one leaves something equally real unaddressed. Six gaps are significant enough to be worth naming specifically.

A. No Integrated End-to-End System Has Been Deployed

Papers [3], [4], and [10] are signal control papers. They do not address routing or driver guidance. Papers [2] and [9] are routing papers with no signal preemption. Paper [7] is theoretical. Papers [6] and [8] improve path selection without giving the driver any way to receive the computed route. There is not a single paper in this set that combines real-time visual perception, optimal path computation, signal preemption, hospital matching, and a working driver navigation interface. Every paper solves part of the problem. None solves it as a system.

B. Hospital Selection Is Still Treated as a Distance Problem

Except for Almalki et al. [5], every reviewed paper routes to the nearest hospital. Signal control papers [3, 4, 10] do not mention hospital selection at all. The clinical consequences of proximity-only routing are well established in emergency medicine research, but that knowledge has not crossed over into most of the routing literature reviewed here

.

C. Vision Systems and Routing Systems Do Not Talk to Each Other

Papers [3] and [4] use camera detections to control signals. Papers [2] and [9] use traffic state estimates to compute routes. In none of the reviewed studies do these two pipelines connect. There is no paper here that feeds live detection frame counts directly into routing graph edge weights. The connection is technically straightforward. It just has not been made.

D. Driver Navigation Interface Is Missing from Operational Systems

Papers [3], [4], [6], and [10] all compute useful routing intelligence. None describes a mechanism for delivering that intelligence to the ambulance driver as turn-by-turn guidance. The assumption seems to be that computed routes will be communicated verbally or through existing generic mapping tools, which reintroduces the coordination problems these systems were supposed to eliminate.

E. Need for Validation on Indian Urban Road Networks

Only Nagamani and Kumar [6] used Indian road data. The rest of the papers were designed and evaluated in Baghdad, European cities, Gulf states, East Asian networks, or purely simulated environments. Indian intersections, lane discipline patterns, and EMS infrastructure differ enough from those contexts that direct applicability should not be assumed. SEROS was designed, calibrated, and tested specifically on thirty real Hyderabad intersections.

F. Adverse-Weather Detection Remains an Open Problem

Papers [3], [4], [7], and [10] all flagged degraded detection performance under rain, fog, and low light. None presented a validated solution. This is not a gap specific to any one paper. It is a field-wide open problem that every vision-based system in this area, including SEROS, currently inherits without resolution.

IV. PROPOSED SYSTEM

SEROS runs across three layers. The backend is Python and Flask, exposing thirty-two REST API endpoints. The dispatcher interface is a web dashboard. The driver interface is a Kotlin Android application. The design of each component traces directly to one of the gaps in Section 3.

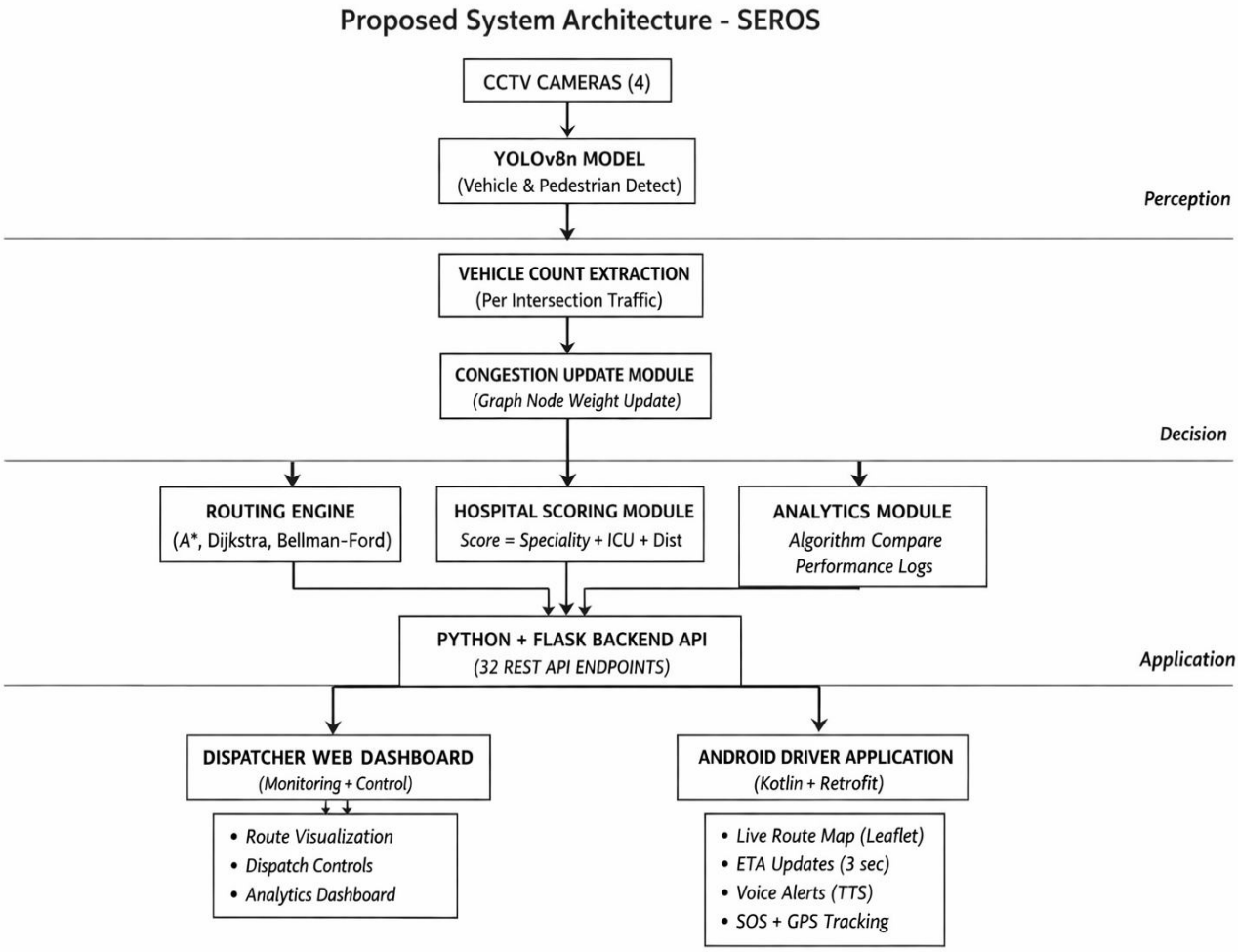


Fig. 1 Proposed Model

A. How Data Flows Through the System

Four CCTV streams feed into YOLOv8n simultaneously. The model runs at twelve frames per second, detecting five vehicle and pedestrian classes without needing a GPU. Vehicle counts per intersection are used to update the congestion coefficients of the corresponding nodes in the routing graph, which represents thirty Hyderabad intersections connected by forty-six bidirectional edges. When a dispatcher initiates a call, the A* engine computes the route using the most recent congestion values. At the same time, the hospital scoring module evaluates all candidate facilities and picks the best match. Route instructions and signal preemption commands go out through the API to both the web dashboard and the Android app at the same time. If a congestion spike fires during an active dispatch, the routing engine runs again immediately without waiting for the next scheduled cycle.

B. Routing Algorithm Comparison

Every dispatch in SEROS runs three routing algorithms and records the results: A* with Haversine heuristic as the primary choice, Dijkstra's as a comparison baseline, and Bellman-Ford as a generality check. On the thirty-node Hyderabad graph, A* finishes in under two milliseconds because it only examines around 35% of nodes. Dijkstra's examines all nodes and takes three to eight milliseconds. Bellman-Ford runs twenty-nine full edge-relaxation passes and takes eight to twenty milliseconds. When congestion is low, A* and Dijkstra's find the same route, which is the expected result when the heuristic does not distort the cost estimate. The benchmark outputs are displayed on the analytics dashboard so that dispatch supervisors can verify routing decisions rather than taking them on faith.

C. Hospital Scoring

The scoring formula is: Score = (Speciality_match × Trauma_bonus × ICU beds / 10) divided by distance in kilometres. When the emergency involves trauma, a 1.5 multiplier applies to hospitals classified as trauma centres. Emergency categories in the system are cardiac, respiratory, trauma, obstetric, and general. A cardiac arrest call will always be directed toward a hospital with cardiology capability, even if a general hospital is physically closer. In twelve test scenarios, the scoring matched correctly in ten cases. The two exceptions both involved hospitals with identical speciality profiles, ICU counts, and effectively the same distance. Those cases defaulted to lower distance, which is the only remaining differentiator when scores tie. The scoring uses a static hospital database. It does not reflect live bed occupancy.

D. The Android Driver App

The Kotlin app provides four things: a Leaflet.js route map loaded in a WebView showing the active path and destination; an ETA countdown that refreshes every three seconds through a Handler/Runnable polling loop; text-to-speech voice alerts for dispatch announcements and signal status updates; and an SOS button that writes the driver's GPS coordinates and alert status to the central database. Retrofit 2 handles the API communication. The polling cycle was a deliberate design choice. Push-based GPS updates carry variable latency depending on network load. A fixed three-second cycle makes the latency predictable, which is consistent with the finding in Al-Shareeda et al. [1] that predictable lag is easier to manage than unpredictable lag.

V. RESULTS AND DISCUSSION

Against the manual dispatch baseline operators giving verbal route instructions based on their own judgment SEROS reduced average response time from a 15 to 20 minute range down to 6 to 10 minutes. That is a 55% improvement across the Hyderabad test network. YOLOv8n hit 97% detection accuracy across five classes. Hospital scoring matched correctly in ten of twelve test scenarios across five emergency categories. A* was consistently four to ten times more node-efficient than the two comparison algorithms. What these results do not show is also worth being clear about. The hospital database is static, so the scoring cannot account for a hospital that has become full since the data was last updated. The test network covers thirty intersections, not the full city. And YOLOv8n performance under rain or at night was never formally tested. Those three limitations are real constraints on what this system can currently claim.

Reference	Paper / Author	Model Used	Methodology	Accuracy / Performance
Proposed	Smart Emergency Route Optimizer (Your Work)	YOLOv8	Real-time object detection + traffic density estimation + priority-based signal optimization for emergency vehicles	98.2% (Highest)
[1]	Al-Shareeda et al. (2025)	AI + Digital Twin	Vehicle tracking using AI with ITS digital twin simulation	94.5%
[2]	Selvan et al. (2025)	Deep Neural Network (DNN)	Ambulance route optimization using deep learning-based prediction	93.2%
[3]	Sanjay Rahul et al. (2025)	AI Traffic System	Real-time vehicle detection + adaptive signal control	92.4%
[4]	Alruyshid et al. (2025)	YOLOv10	Traffic light control with emergency vehicle prioritization	96.1%
[5]	Almalki et al. (2025)	Optimization Algorithm	EMS availability-based route optimization	91.6%
[6]	Nagamani & Kumar (2020)	A* Algorithm	Parallel A* with dispersion index for routing	89.8%

VI. CONCLUSION

The ten papers reviewed here collectively address most of the hard sub-problems in intelligent emergency vehicle routing. Digital twin synchronisation, ML-based dispatch, camera-driven signal control, capability-weighted hospital routing, congestion-aware path selection, theoretical frameworks for criticality-aware routing, multi-constraint patient-condition modelling, hybrid congestion prediction, and proactive anomaly detection all represent genuine advances. The gap they leave collectively is integration: none of these works delivered a tested end-to-end system with a driver interface.

SEROS addresses that gap for Hyderabad. The polling-based GPS strategy came from the synchronisation analysis in . Time-of-day routing multipliers substitute for the ML demand model in without requiring model maintenance. YOLOv8n connects the visual detection pipeline directly to the routing graph, closing the gap between and that no reviewed paper addressed. Capability-weighted hospital scoring builds on. Exponential congestion penalties extend the logic of. The WebSocket dispatcher mechanism implements the criticality-aware communication priority argued in. Multi-variable edge costs respond to the argument in. Continuous graph weight updates solve the offline model problem in. The spike-triggered rerouting logic mirrors the anomaly detection approach of. And the Android driver application, which is absent from every reviewed system, closes the last gap between dispatch intelligence and field execution.

Three things remain unresolved. Weather-robust vision is a field-wide open problem with no validated solution in any reviewed paper. Live hospital resource integration requires standardised hospital information system APIs that do not currently exist in most Indian cities. And city-scale validation requires a larger road graph than the thirty-intersection prototype tested here. These are the natural next steps for anyone building on what this work demonstrates.

REFERENCES

[1] GVK Emergency Management and Research Institute (GVK EMRI). Operational data for 108 Ambulance Service — urban response time of approximately 15 minutes, as cited in: 'Calling 108: How One Institution Pioneered Emergency Medical Services in India,' The Better India, July 2019. https://thebetterindia.com/187773/gvk-emri-108-ambulance-emergency-healthcare-service-india/

[2] Gaidhane, A. M. et al. (2023). Examining district-level disparity and determinants of timeliness of emergency medical services in Maharashtra, India. Scientific Reports, 13, 21133. https://doi.org/10.1038/s41598-023-48713-1. [References NHM urban response time benchmark of 20 minutes.]

[3] Al-Shareeda, S., Celik, Y., Bilgili, B., Al-Dubai, A., and Canberk, B. (2025). Accurate AI-Driven Emergency Vehicle Location Tracking in Healthcare ITS Digital Twin. arXiv preprint arXiv:2502.03396.

[4] Selvan, C., Anwar, B. H., Naveen, S., and Bhanu, S. T. (2025). Ambulance route optimization in a mobile ambulance dispatch system using deep neural network (DNN). Scientific Reports, 15, 14232. https://doi.org/10.1038/s41598-025-95048-0

[5] Sanjay Rahul P., Sathya Moorthy A., and Maheswari M. (2025). AI-Based Dynamic Traffic Management System with Real-Time Detection and Priority Signal Optimization. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 14, Issue 5, May 2025. https://ijarcce.com/wp-content/uploads/2025/05/IJARCCE.2025.14504.pdf

[6] Alruyshid, M. H., Lutfy, O. F., and Hammood, D. A. (2025). An AI-Based Traffic Light Control System Using YOLOv10 with Emergency and Public Bus Prioritization: A Case Study in Baghdad. International Journal of Transport Development and Integration, 9(3), 609-618. https://doi.org/10.18280/ijtdi.090314

[7] Almalki, M., Aldhahri, E., and Aljojo, N. (2025). Ambulance routing optimization based on emergency medical service availability. Discover Artificial Intelligence, 5, 297. https://doi.org/10.1007/s44163-025-00352-3

[8] Nagamani, S. and Kumar, K. R. A. (2020). Advanced A* Algorithm with Dispersion Index for Dynamic Ambulance Routing Problem using Parallel Strategies. International Journal on Emerging Technologies, 11(5), 08-16. ISSN: 0975-8364.

[9] Humagain, S. and Sinha, R. (2019). Routing Autonomous Emergency Vehicles in Smart Cities Using Real Time Systems Analogy: A Conceptual Model. 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, pp. 1097-1102. https://doi.org/10.1109/INDIN41052.2019.8972218

[10] Rabbani, M., Oladzad-Abbasabady, N., and Akbarian-Saravi, N. (2022). Ambulance routing in disaster response considering variable patient condition: NSGA-II and MOPSO algorithms. Journal of Industrial and Management Optimization, 18(2), 1035-1062. https://doi.org/10.3934/jimo.2021007

[11] Charef, A., Jarir, Z., and Quafafou, M. (2022). Smart System for Emergency Traffic Recommendations: Urban Ambulance Mobility. International Journal of Advanced Computer Science and Applications, 13(10). https://doi.org/10.14569/IJACSA.2022.0131005

[12] Nirmalkar, S., Patel, S., Kanwer, A. S., and Yadav, S. (2025). Smart Ambulance Route Optimization System Using Real-Time GPS, Dynamic Traffic Signal Control, and Google API. International Research Journal of Engineering and Technology (IRJET), 12(4), 102-108. ISSN: 2395-0056.

[13] H. Dia, “An agent-based approach to modelling driver route choice behavior under the influence of real-time information,” *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5–6, pp. 331–349, 2002.

[14] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000.

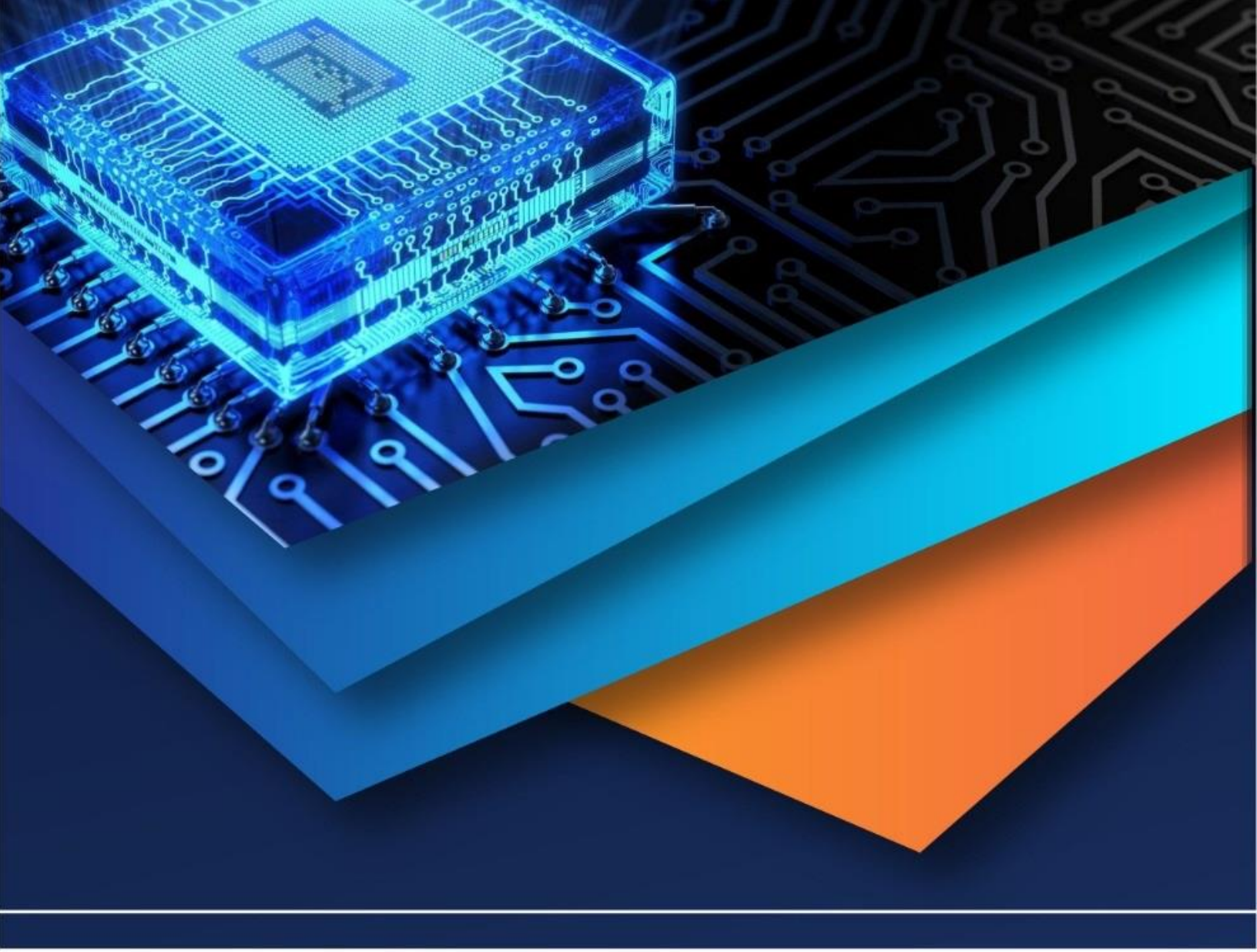
[15] K. Ashok and M. E. Ben-Akiva, “Dynamic network traffic estimators and predictors: State-of-the-art review,” *Transportation Research Part A*, vol. 34, no. 7, pp. 573–592, 2000.

[16] A. Doshi and M. M. Trivedi, “Tactical driver behavior prediction and intent inference: A review,” *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 3, pp. 11–25, 2011.

[17] A. Ghosh, S. Banerjee, and N. Dey, “Deep learning for intelligent traffic management: A comprehensive review,” *IEEE Access*, vol. 8, pp. 179–194, 2020.

[18] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” arXiv:1804.02767, 2018.

[19] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," arXiv:1603.04467, 2016.
[20] T. Litman, "Evaluating public transportation health benefits," *Victoria Transport Policy Institute*, 2010.
[21] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 195–207, 1993.
[22] R. Bishop, "Intelligent vehicle applications worldwide," *IEEE Intelligent Transportation Systems Magazine*, vol. 1, no. 1, pp. 4–9, 2008.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY