Proceedings of CONF-CDS 2026 Symposuim: Data-Centric AI Security: Securing Models, Learning Agents, and Autonomous Systems
DOI: 10.54254/2755-2721/2026.CH34985

Onboard Pathfinding for Urban Delivery Robots on Known Road Networks: An Embedded Systems Review

Chuan Jiang

Faculty of Science and Engineering, University of Nottingham Ningbo China, Ningbo, China
ssycj1@nottingham.edu.cn

Abstract. This review examines single-robot onboard path planning for urban delivery robots operating on known road or sidewalk networks under embedded resource constraints. The analysis emphasizes practical deployment scenarios, where CPU capacity, memory size, power budget, and planning time are limited. Based on representative journal and conference studies in mobile robot navigation, delivery robotics, and embedded robotic systems, the paper reviews how known urban networks can be represented and how planning methods can be selected for resource-constrained platforms. The discussion covers graph-based map representations, global route planning methods, local obstacle handling, incremental replanning, and implementation-oriented strategies like compact data structures, offline preprocessing, bounded online search, and fallback mechanisms. The results demonstrate that dense map representations are generally unsuitable for low-cost onboard deployment, while sparse topological or hybrid graphs provide a better balance among storage requirements, search efficiency, and implementation simplicity in known-network scenarios. Moreover, classical graph search algorithms, like Dijkstra and A*, remain effective for small to medium networks. In contrast, preprocessing-based or incremental approaches are more advantageous when handling repeated queries, local cost updates, or strict timing constraints. It further indicate that lightweight graph-based path planning with bounded runtime and local recovery is more suitable for urban delivery robots than heavy, compute-intensive planning pipelines.

Keywords: Path planning, urban delivery robots, embedded systems, known road networks, incremental replanning

1. Introduction

Urban delivery robots have gained growing interest in last-mile logistics due to their ability to handle short-distance, repetitive delivery operations in campuses, business parks, residential communities, and other semi-structured settings [1, 2]. In many scenarios, the robot operates on a known road or sidewalk network rather than in a fully unknown space. Even under this assumption, it must plan routes that work within limited CPU, memory, power, and time. These constraints make its path planning different from purely algorithmic shortest-path problems, because effective solutions need to account for route quality, speed reliability, cost, and fault recovery [3, 4]. This paper explores how a single urban delivery robot can efficiently plan routes on a known road

DOI: 10.54254/2755-2721/2026.CH34985

network under limited onboard resources. To explore this, representative studies on mobile robot navigation, delivery robotics, graph-based path planning, and embedded systems are reviewed. It focuses on road-network representation, global route planning, local obstacle handling, incremental replanning, and implementation strategies, including compact data structures, offline preprocessing, bounded online search, and fallback mechanisms. In this context, the problem is treated as a deployment-oriented systems challenge, where navigation trades off performance, cost, and complexity. This study help guide future research toward methods that are both algorithmically efficient and practical to deploy, maintain, and scale in real urban delivery systems.

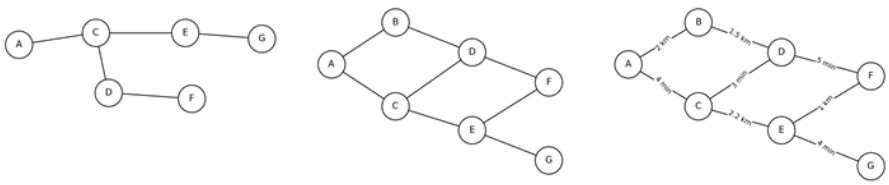
2. Fundamentals of path planning for urban delivery robots

2.1. Vehicle path planning problem

The vehicle path planning problem involves generating onboard routes for a single urban delivery robot moving from an origin node to a destination node within a known road or sidewalk network. In this context, the environment is represented as a weighted graph $G=(V,E)$, where nodes denote intersections, entrances, crossings, or delivery points, and edges correspond to traversable road or sidewalk segments. On this graph, the route computation, or pathfinding task, aims to efficiently identify a feasible route. In practical delivery tasks, route cost may include distance, travel time, energy use, and turn penalties. Within this framework, onboard path planning can be formulated as a graph-search problem, where route generation and route updating constitute its two fundamental functions [3, 4]. When local edge costs change during execution, incremental pathfinding methods such as D* Lite are relevant because they can update the route without restarting the full search from scratch [5].

2.2. Modeling road networks for onboard navigation

Known road networks can be represented at different levels of abstraction, as illustrated in Figure 1. In particular, sparse graphs enhance efficiency by reducing memory demands and computational overhead, retaining only the most necessary connections. In contrast, topological graphs capture the connectivity of road segments, which is particularly useful for onboard path planning in structured networks. Hybrid topological-metric graphs additionally attach distance or travel-time information to edges, making them more suitable for practical onboard path planning tasks. For embedded urban delivery robots, hybrid graph representations are generally preferred over dense spatial maps because they preserve critical routing information while keeping onboard computation manageable. In known-network scenarios, this representation provides an effective compromise between compact storage, search efficiency, and cost modeling [6-8].



(a) Sparse Graph (b) Topological Graph (c) Hybrid Topological-Metric Graph

Figure 1. Types of road network representations (a. minimal nodes/edges; b. connectivitypreserved; c. topology preserved with distances/travel times attached)

DOI: 10.54254/2755-2721/2026.CH34985

2.3. Major strategies for global path planning

For known road networks, several classical graph-search approaches can be applied to onboard path planning. From an embedded systems perspective, the primary considerations include path optimality, computational efficiency, memory usage, preprocessing cost, and implementation complexity. In this context, Dijkstra is simple and robust, but its computational efficiency decreases as the graph size increases. In contrast, A* is often more suitable for onboard use as its lightweight heuristic minimizes redundant node expansions while also maintaining low memory requirements. In addition, Contraction Hierarchies (CH) enable very fast path queries on large and stable networks, although they require offline preprocessing and additional storage. Similarly, ARA* proves advantageous when planning time is constrained, as it can rapidly produce a feasible solution and iteratively refine it if additional computation time is available. For small to medium known networks, A* generally provides the best balance between efficiency and implementation simplicity, whereas CH and ARA* offer greater benefits for repeated queries or stricter timing constraints, as shown in Table 2 [3, 8, 9].

Table 1. Local obstacle avoidance and incremental replanning

Method	Online time	Memory	Preprocessing	Implementation complexity
Dijkstra	Moderate	Low	None	Low
A^+	Low-moderate	Low	None	Low
CH	Very low query time	Moderate	Required	Higher
ARA^+	Budget-dependent	Moderate	None	Moderate

3. The design of embedded systems and local control strategies

3.1. Local obstacle avoidance and incremental replanning

In urban delivery scenarios, disturbances occur frequently, so execution cannot rely solely on the initial global path. Thus, navigation is typically organized into two layers, with the global layer computing a route on the known road network and the local layer adjusting motion over a short horizon. And this structure is widely used in mobile robot navigation, because it keeps the global routing relatively stable and at the same time allows fast local response during movement [3, 4].

For local motion control, the Dynamic Window Approach (DWA) tests feasible velocity commands under the dynamic limits of the robot, and selects a short-horizon motion according to progress, safety, and smoothness. This method is suitable for embedded delivery robots, because the computation is local and can be repeatedly executed in a fixed control cycle [10]. Nevertheless, local avoidance alone may be insufficient. For example, when a previous edge becomes blocked because of cost changes or temporary obstacles, the robot must update the path. In this case, incremental replanning methods such as D* Lite are preferable, as the search updates based on local cost changes rather than recomputing the entire route. This speeds up response and reduces unnecessary computation when only a small area is affected [5].

For known urban road networks, the on-board task is therefore to first compute a global path on the sparse graph, then track it by the local controller, detect blocked or risky parts during operation, update the corresponding edge costs, and trigger incremental replanning in the affected part of the network. In this way, route continuity, local safety, and moderate resource usage can be achieved for compute-limited delivery robots working in changing urban environments [4, 5].

DOI: 10.54254/2755-2721/2026.CH34985

3.2. Embedded-oriented system design

Figure 2 illustrates an onboard navigation architecture that is designed to emphasize memory efficiency, maintain a bounded online workload, and ensure predictable runtime stability.

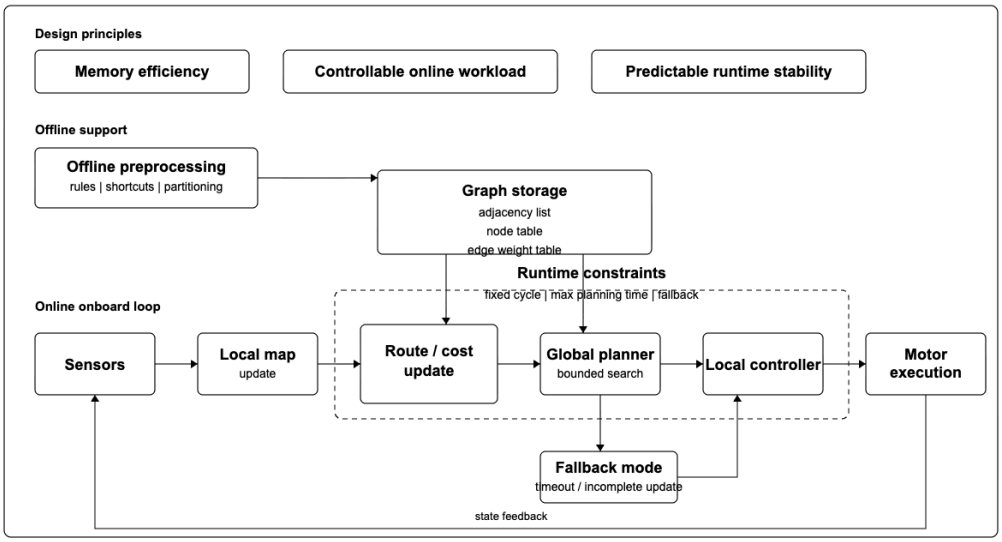


Figure 2. Onboard nvigation achitecture

For memory efficiency, the known road network is stored in a compact graph form rather than a dense spatial map. In this setup, graph storage uses an adjacency list, a node table, and an edge weight table, which lowers memory demand while retaining the information required for route planning on embedded platforms [7, 8]. Such a design is particularly suited for known-network navigation, where sparse graphs are more practical than dense map representations.

To control the online computation load, some tasks are shifted to offline preprocessing, including static access rules, shortcut construction, and graph partitioning, while the online loop manages local map updates, route validation, cost updates, and incremental replanning. This separation keeps onboard workload manageable and restricts search to relevant graph sections during operation, which is essential under constrained CPU and timing budgets [8, 11].

Besides, predictable runtime stability is ensured by organizing sensing, local updates, planning, and control into a repeated onboard cycle with explicit bounds, like a set update interval, maximum planning time, and fallback mode. Bounded execution maintains route continuity even if a full update cannot be completed in the current cycle. In these situations, the system can keep the current route, apply a partial update, or fall back to a simpler control strategy until the next cycle. This conservative design enhances timing consistency and system reliability under real resource constraints [4, 5, 9, 12].

3.3. Cost-effectiveness evaluation metrics

For embedded onboard path planning, evaluation should consider four groups of metrics: path quality, computational performance, operational performance, and deployment cost, as summarized in Table 3. However, these categories are not equally important for embedded urban delivery robots. In practice, computational performance and operational performance generally take the highest

DOI: 10.54254/2755-2721/2026.CH34985

priority because the planner must return a usable route within limited CPU, memory, and timing budgets while maintaining stable execution [4, 5].

Table 2. Path planning evaluation metrics

Metric Group	Specific Metrics
Path Quality	Path length, Travel time, Number of turns, Energy estimate
Computational Performance	Mean planning time, P95 / P99 planning time, Replanning latency, Memory occupancy
Operational Performance	Mission success rate, Recovery capability, Path stability, Execution continuity
Deployment Cost	Hardware requirement, Deployment complexity, Maintenance burden, System integration effort

Regarding path quality, metrics like path length, travel time, number of turns, and energy estimate reflect delivery efficiency. Nevertheless, for embedded deployment, a slightly less optimal route may be acceptable if it can be generated more reliably and within a predictable runtime budget. By contrast, computational performance metrics, including mean planning time, P95/P99 latency, replanning delay, and memory occupancy, are more directly related to whether the planning method can be sustained onboard [4].

Similarly, operational performance metrics matter. For delivery robots, mission success, recovery, path stability, and continuous operation are key, since the system must work under local blockages and temporary disruptions, not just handle single route queries. In addition, deployment cost represents a critical practical constraint, as hardware requirements, integration effort, and maintenance burden can significantly influence long-term adoption. These metric groups are best understood as a priority-based evaluation framework, in which bounded runtime and stable operation outweigh minor gains in path optimality [2, 12].

4. Implementation and challenges of urban delivery robots

4.1. Applications in urban delivery robots

Urban delivery robots operate in several typical scenarios, which differ in environmental stability, planning demand, and hardware requirements. Onboard path planning strategies are better compared by scenario type rather than described repeatedly in terms of known networks and local obstacle handling.

In campus environments, conditions are fairly stable and structured. Roads, crossings, and delivery points are often repeated, and route changes across the network are usually few. Therefore, planning can rely on a preconstructed road graph with lightweight onboard route queries and moderate local adjustments during execution. Hardware demand is also moderate because the service area is often bounded and the route network can be prepared in advance. This makes campus delivery one of the most suitable scenarios for low-cost onboard path planning [1, 2].

Industrial and business parks also favor onboard planning, but the service area is typically larger and subject to more regular access rules. Although these environments remain comparatively stable, route planning may need to support repeated queries over longer distances and across multiple subregions. In such cases, planning gains from compact graph storage, offline preprocessing, and region-based routing to cut query delay. Hardware needs stay manageable, but bigger maps and more frequent queries make preprocessing and memory use more important than in campus settings [8, 11].

DOI: 10.54254/2755-2721/2026.CH34985

By contrast, community and sidewalk delivery environments are less stable, since narrow passages, parked vehicles, shared pedestrian spaces, and temporary blockages occur more frequently than in campus or park scenarios. Therefore, planning must focus on local responsiveness and route recovery, not just on a global path. Consequently, the system needs both efficient global planning and local control with incremental replanning. Although hardware costs remain low, operational demand is higher due to frequent route updates [4, 5, 10]. The main difference across scenarios is network stability and the frequency of local changes. Lightweight graph-based routing with limited online updates works for campus and industrial parks, whereas community and sidewalk delivery requires local adjustment and incremental replanning. Thus, onboard path planning should match the deployment context, not use a single solution for all environments [7, 12].

4.2. Existing limitations and future research directions

Although graph-based pathfinding on known networks is well-studied, urban delivery robots still face practical limits. In particular, modeling costs under dynamic pedestrian flow remains unresolved. At the same time, current planners rely on static or slowly varying edge costs, whereas sidewalks and shared spaces change rapidly in crowd density, waiting time, and risk. Therefore, refining edge costs to reflect pedestrians, crossings, and temporary blockages would improve both global planning and local replanning for real delivery scenarios [2, 3].

In addition, another important direction is to jointly consider path quality and resource consumption. Many studies mainly focus on route length or travel time. For embedded deployment, memory usage, planning delay, and runtime stability are also important factors. Future planning approaches could treat route efficiency and computational load as dual objectives, enabling a better balance between transport performance and onboard hardware constraints [4, 12]. Moreover, the combination of learning-based methods and graph search is a promising approach. This framework permits estimation of edge costs, traffic states, or local traversability, while conventional graph search ensures transparent structure, reliable runtime, and simpler validation. It also benefits delivery robots by maintaining an interpretable, resource-aware planning pipeline, and enhancing adaptation to complex urban environments [3, 4].

Despite these developments, a persistent challenge is the absence of standardized benchmarks for embedded pathfinding systems. Existing studies often use different metrics, maps, and hardware assumptions, making direct comparison difficult. A standardized benchmark could include common road-network datasets, fixed onboard hardware configurations, and a shared set of evaluation metrics, like path quality, replanning delay, tail runtime, and memory usage [4, 12]. Another important direction is to incorporate real urban delivery data. Current research on autonomous delivery robots remains largely simulation-based or concept-driven, with insufficient field data on street flow, blockage rates, stop durations, and service demand. Future work could become more practical by combining known-network planning with real delivery logs and urban operating data, facilitating evaluation and deployment of onboard pathfinding systems [1, 2].

5. Conclusion

This paper has surveyed the onboard pathfinding problem for urban delivery robots operating on known road or sidewalk networks under embedded resource constraints. In such scenarios, pathfinding can be understood as efficient route generation and maintenance on a prepared traversable graph, where the robot must balance path quality, computational cost, and execution stability during real-time deployment. Moreover, map representation, graph search, local control,

DOI: 10.54254/2755-2721/2026.CH34985

and replanning are closely interconnected, collectively influencing runtime, memory usage, recovery capability, and field reliability. A practical framework discussed in this paper combines sparse graph representation, graph-based global search, local motion control, and incremental replanning. Within this framework, the urban network is stored as a compact weighted graph, global routes are computed using lightweight search methods such as Dijkstra or A*, the robot follows these routes via a local controller such as DWA, and route updates are performed by adjusting edge costs and triggering incremental replanning in response to local blockages or temporary disruptions, which supports repeated route queries while maintaining responsive behavior on robots with limited onboard computation. From a deployment perspective, low-cost onboard implementation is achievable through small data structures, offline preprocessing, local graph partitioning, bounded online search, and fallback strategies that maintain route continuity under short-term overloads or environmental changes, enabling practical operation with reasonable hardware requirements and relatively low software complexity. Nevertheless, further research is needed to improve dynamic cost modeling, integrate path planning more closely with real delivery data, and develop evaluation methods that explicitly link navigation performance to embedded resource consumption, which are essential to enhance the practicality, reliability, and efficiency of onboard pathfinding systems in real-world urban delivery scenarios.

References

[1] Srinivas, S., Ramachandiran, S., & Rajendran, S. (2022). Autonomous robot-driven deliveries: A review of recent developments and future directions. Transportation Research Part E: Logistics and Transportation Review, 165, 102834.
[2] Alverhed, E., Hellgren, S., Isaksson, H., Olsson, L., Palmqvist, H., & Woxenius, J. (2024). Autonomous last-mile delivery robots: A literature review. European Transport Research Review, 16(4).
[3] Liu, L., Wang, X., Yang, X., Liu, H., Li, J., & Wang, P. (2023). Path planning techniques for mobile robots: Review and prospect. Expert Systems with Applications, 227, 120254.
[4] Macenski, S., Moore, T., Lu, D. V., Merzlyakov, A., & Ferguson, M. (2023). From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the Robot Operating System 2. Robotics and Autonomous Systems, 168, 104493.
[5] Koenig, S., & Likhachev, M. (2002). D* lite. In Eighteenth national conference on Artificial intelligence, 476-483.
[6] Thrun, S., & Bücken, A. (1996). Integrating grid-based and topological maps for mobile robot navigation. Proceedings of AAAI-96.
[7] Tang, L., Wang, Y., Ding, X., Yin, H., Xiong, R., & Huang, S. (2019). Topological local-metric framework for mobile robots navigation: A long term perspective. Autonomous Robots, 43, 197-211.
[8] Geisberger, R., Sanders, P., Schultes, D., & Vetter, C. (2012). Exact routing in large road networks using contraction hierarchies. Transportation Science, 46(3), 388–404.
[9] Likhachev, M., Gordon, G. J., & Thrun, S. (2004). ARA*: Anytime A* with provable bounds on sub-optimality. Advances in Neural Information Processing Systems, 16.
[10] Fox, D., Burgard, W., & Thrun, S. (1997). The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1), 23-33.
[11] Muravyev, K., & Yakovlev, K. (2024). NavTopo: Leveraging topological maps for autonomous navigation of a mobile robot. In Interactive Collaborative Robotics (Lecture Notes in Computer Science, 14898, 144-157.
[12] Tahir, N., & Parasuraman, R. (2025). Edge computing and its application in robotics: A survey. Journal of Sensor and Actuator Networks, 14(4), 65.