GraphMap: Scalable Crowd-Sourced Global Vectorized HD Map Construction via Sparse Visual Graph Fusion

Ruiyang Zhu[1], Minkyoung Cho[1], Shuqing Zeng[2], Fan Bai[2], Z. Morley Mao[1]

***Abstract*— High-definition (HD) maps are vital for autonomous driving, offering fine-grained geometric and semantic information beyond onboard perceptions. While recent vision-based methods enable *online local* HD map detection, constructing accurate *global* vectorized maps at scale remains a fundamental challenge: (1) individual vehicles have limited sensing range, and (2) the mainstream ego-centric dense fusion approaches for collaborative perception suffer from poor scalability, high computation costs, and sensitivity to spatial-temporal misalignment. In this paper, we present `GraphMap`, a novel crowd-sourced vehicle-cloud framework for global vectorized HD map construction via sparse graph fusion. Unlike prior collaborative perception methods that rely on dense, fixed-size bird's-eye-view (BEV) features, our approach is not constrained by ego-centric views or fixed perception ranges. `GraphMap` encodes local vectorized maps as sparse geometric graphs and incrementally fuses them through a sparse-to-sparse graph fusion algorithm. To reduce network bandwidth and computation overhead, `GraphMap` incorporates a selective map sensing mechanism that prioritizes data uploads from more informative agents with higher sensing value. Experiments across multiple real-world and simulated driving scenes with up to 60 agents demonstrate that `GraphMap` constructs accurate global maps under sparse, spatially misaligned observations, outperforming state-of-the-art baselines by 37.2 mAP while reducing communication costs by more than 30%.**

I. Introduction

High-definition (HD) maps [1], [2], [3] play a critical role in autonomous driving by providing rich geometric and semantic information, such as lane dividers, boundaries, and pedestrian crossings. Among various formats [4], [5], [6], [7], [8], [9], [10], [11], vectorized HD maps [4], [5], which represent map elements as structured primitives such as polylines and polygons, are becoming the new technical trend of mapping solutions [12], [13], [14] for connected and autonomous vehicles (CAVs). Conventional HD map construction pipelines [15], [16] often rely on collecting dense LiDAR point clouds and require significant human annotation, limiting scalability and update frequency. Recent advances in online HD mapping [14], [13], [17] use deep learning models to generate local HD maps around the ego-vehicle in real time from onboard sensors, significantly reducing the need for offline human intervention. However, individual vehicles suffer from limited sensing range [17], occlusions [6], and incomplete coverage [18], making it difficult to build reliable global maps independently. This motivates a shift toward ***crowdsourced global vectorized*** HD map construction, where a fleet of vehicles (agents) contributes local observations [2] to collaboratively build a unified global map [19], [20], [21], [22], [23].

Previous efforts in online HD mapping and collaborative perception have primarily relied on **dense feature fusion**, where sensor data (*e.g.,* Cameras and LiDARs) are encoded and projected into a shared bird's-eye-view (BEV) representation [6], [24], [25], [26]. These methods aim to align perspective differences among agents and sensors by aggregating features on fixed-size BEV tensors. However, dense fusion methods suffer from several key limitations: First, they are inherently ego-centric, bounded by local fields of view and fixed perception ranges (*e.g.,* 60 $\times$ 30 m), making them poorly suited for constructing global maps; Second, they are sensitive to spatial and temporal misalignment [26], [25], struggling to robustly fuse spatially misaligned and partially overlapped observations from different agents; Third, increasing the size of dense BEV tensor can incur significant computational [27] and communication overhead [28] due to the high-dimensional, redundant nature of dense feature maps; These limitations highlight the need for a more *scalable*, *efficient*, and *robust* approach to global vectorized HD map construction, particularly in large-scale and dynamically changing road environments. To meet these goals, a new approach must: (1) incorporate multi-agent observations and handle spatial-temporal misalignment over extended space and time in a sparse manner; and (2) operate efficiently under limited network and computation resources.

To address the key challenges of scalable global HD map construction, we propose `GraphMap`, a novel framework that shifts from dense, ego-centric fusion to a graph-based sparse fusion for collaborative, vectorized global HD map construction. `GraphMap` operates in a two-tier interactive architecture consisting of vehicle-side sensing and cloud-side aggregation and scheduling, with the following key features:

Scalable multi-agent sparse fusion across space and time. To enable scalable fusion across spatially misaligned, partially overlapping observations, `GraphMap` adopts a graph-based representation of local HD maps. Each agent creates a compact map graph from onboard sensor data using online HD map generators. These graphs consist of semantic and geometric attributes (e.g., type, shape, orientation) of map elements and are lightweight for transmission. On the cloud, we introduce the MapGlue algorithm, which incrementally aligns and merges local maps through Type-Aware Structure-Preserving Graph Matching. By modeling intra-region graph structure and solving the NP-hard maximum subgraph matching problem [29] with an efficient greedy

[1]University of Michigan, Ann Arbor, USA {ryanzhu, minkycho, zmao}@umich.edu
[2]General Motors LLC, Warren, MI, USA {shuqing.zeng, fan.bai}@gm.com

approximation, MapGlue robustly handles spatial variation and partial overlaps while preserving topological consistency in map alignment. For map growth beyond local regions, MapGlue performs inter-region graph registration, expanding coverage with new elements and merging consistent observations across space and time.

Communication and computation-efficient map contribution. Rather than uploading every local observation, agents and cloud in GraphMap jointly make data contribution decisions via a selective map sensing policy. This policy evaluates the sensing value (SV) of individual local maps by considering three key factors: (1) the trade-off between exploring new areas and reinforcing known regions of the global map, (2) the redundancy of spatially or temporally proximate observations from the same agent, and (3) the differential contribution of heterogeneous agents such as fixed infrastructure and moving vehicles. Solved with an ϵ-greedy Multi-Armed Bandit (MAB) algorithm, this vehicle-cloud feedback loop optimizes that only informative and non-redundant data are transmitted, reducing network and computation overhead while maintaining global map quality.

Combining the merits above, the resulting system is an end-to-end global map construction framework to build large vectorized maps through crowd-sourcing with multiple agents while maintaining precision and resource efficiency.

In summary, the main contributions of this work are:

- We propose GraphMap, a scalable, graph-based framework for crowdsourced global HD map construction. It achieves robust global stitching of map data from heterogeneous agents by demonstrating tolerance to spatial and temporal misalignments. Our collaborative map construction method, based on sparse graph fusion and selective sensing, enhances accuracy, optimizes wireless network bandwidth usage and computation overhead, improves robustness in dynamic environments, ultimately enabling a more scalable solution for real-world deployment.
- We introduce the MapGlue algorithm, which features a novel Type-Aware Structure-Preserving Graph Matching for efficiently fusing sparse map graphs. On the cloud side, we introduce a Sensing Value (SV)-Guided Selective Map Sensing mechanism to address real-world bandwidth and compute constraints for global map construction. This strategy evaluates the sensing gain of individual local maps based on their potential to improve the global map while maintaining map consistency.
- Our extensive evaluations demonstrate the scalability, effectiveness, and resilience of GraphMap across real-world multi-agent datasets [30], [31] and large-scale simulation town-level scenarios [28] with 30-60 agents. Compared to state-of-the-art methods, GraphMap achieves 37.2 mAP improvement in mapping accuracy while reducing computational overhead by $>3\times$ and bandwidth usage by 32%.

II. Background and Related Work

Online local HD map construction methods. Existing local HD map models, such as HDMapNet [8], VectorMapNet [9], and MapTR [14], are limited by small perception ranges (e.g., 60x30m), with even extensions like StreamMapNet [17] (100x50m) or longer-range efforts like ScalableMap [32] remaining focused on the ego vehicle's immediate vicinity rather than true global coverage. Despite these advancements in local HD map construction, these methods still focus on a relatively small area compared to the requirements for constructing a global HD map. Building an end-to-end global HD map construction framework faces two main challenges: (1) Single-frame or short-sequence data from one vehicle is insufficient for building large regions due to the limited view and occlusions; (2) Scaling transformer-based BEV to global regions is computationally infeasible due to the quadratic complexity. Consequently, global HD map construction necessitates architectural shifts and multi-agent collaboration. GraphMap addresses this by enabling collaborative, large-scale HD map creation through sparse graph fusion across diverse road agents and time, thereby overcoming the limitations of ego-centric local detectors.

Limitation of existing global HD map construction methods. Prior work [33], [34], [19], [35] on global HD map construction primarily employs two fusion strategies: (1) *element-fusion:* Methods like PolyMerge [33] and VMA [36] merge vectorized map instances (*e.g.,* polylines) from a single agent with multiple local frames using rule-based heuristics like proximity thresholds. Rule-based coarse-grained merging, however, overlooks finer features across agents or time, often degrading map quality. (2) *temporal intermediate-fusion:* Recent methods [34], [17], [35] incorporate temporal priors to aggregate historical map information from a single vehicle. While accumulating sequential data over time enables global map construction, this approach is restricted to single-agent settings, failing to leverage crowdsourced multi-vehicle data due to significant spatial misalignments [25] from differing viewpoints. In contrast, GraphMap employs fine-grained, type-aware fusion across both time and agents, facilitating robust global mapping despite real-world sensing and communication noise.

III. Design of GraphMap

A. System Architecture

GraphMap (Fig. 1) operates across two planes. The **data plane** manages the generation, transmission, and fusion of vectorized HD map data. Each road agent (*e.g.,* vehicle i, j or infrastructure unit k) processes onboard camera or LiDAR streams through an online local HD map model (*e.g.,* MapTR [14]) to produce vectorized map segments, which are selectively uploaded to the cloud with GPS/IMU pose. The cloud-side **MapGlue** module (§III-B) transforms observations into global coordinates, encodes them as sparse HD graphs, and fuses them in two stages: *Intra-region Refinement* consolidates overlapping maps of the same region, and *Inter-region Fusion* stitches distinct geographic segments into a progressively expanding global map $\mathcal{G}$. The **control plane** manages upload scheduling. The **Selective Map Sensing** module (§III-C) determines whether to upload each local observation using an sensing value (SV) policy computed jointly from cloud-side global map feedback and agent-local

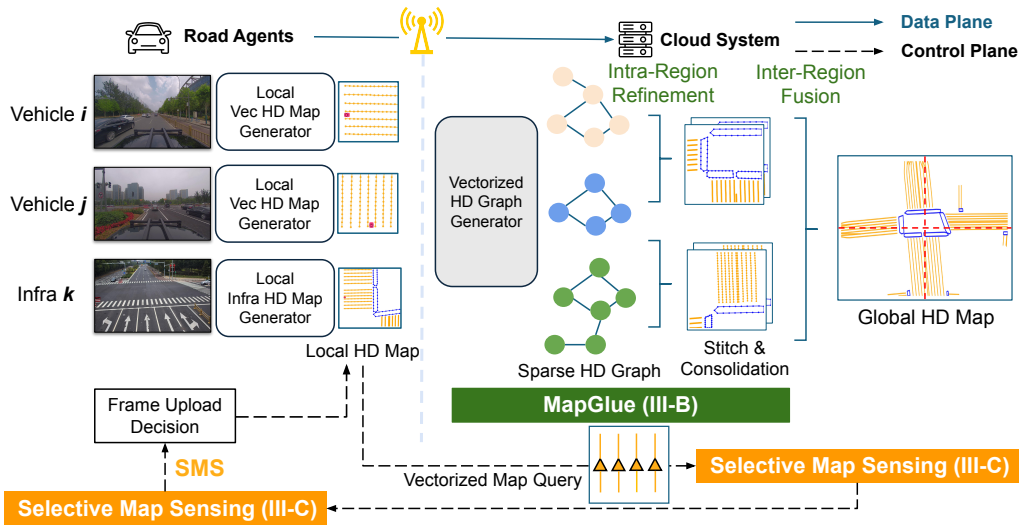


Fig. 1: System Architecture of GraphMap. Local HD map graphs generated by road agents are selectively uploaded to the cloud based on their sensing value. The cloud incrementally fuses these graphs with MapGlue into a global map.

characteristics, prioritizing high-value uploads to optimize bandwidth and onboard computation.

B. MapGlue: Type-Aware Structure-Preserving Graph Matching

CATEGORY ❶: *Shape-Preserving with Spatial Offsets*

Many map elements, when partially observed by different agents at different times, tend to preserve their intrinsic geometric structure (e.g., **lane dividers**, and **road boundaries**). However, these elements can exhibit large spatial offsets in their reported absolute positions due to partial occlusions, limited sensing range, or localization drifts.

CATEGORY ❷: *Location-Preserving with Shape Variations*

Conversely, other types of map elements typically maintain a relatively consistent location or centroid across multiple observations. However, their perceived geometric structure—such as shape, extent, or completeness—can differ substantially. These variations often arise from differing sensor view angles, the granularity of the detection algorithm, or partial views of the element. **Pedestrian crossings** and **junctions** exemplify this behavior.

At the heart of GraphMap's cloud-side fusion pipeline is MapGlue, a Type-Aware Structure-Preserving Graph Matching algorithm designed for robust and scalable vectorized HD map fusion. MapGlue consolidates sparse vectorized map observations from heterogeneous agents, incrementally merging overlapping segments across time and space—even when observations are noisy, spatially misaligned, or temporally disordered.

Prior collaborative perception methods [26], [37], [38] apply graph-based alignment under a *locality assumption*: co-visible objects appear at nearby positions with similar sizes across views, making proximity-based matching effective. This assumption does not hold for global HD map elements, which exhibit the two categorically distinct behaviors above (Fig. 2). Consequently, traditional graph matching approaches that rely on uniform proximity thresholds or fixed structural constraints for all element types are insufficient for reliable multi-agent HD map fusion. MapGlue addresses this by adapting its matching and fusion strategies to the behavioral category of each element: CATEGORY ❶ elements are matched with permitted location shifts but preserved structure; CATEGORY ❷ elements are matched with stricter locality constraints but their polygon geometry is allowed to grow via union to accommodate shape variations.

Graph encoding. MapGlue encodes each local HD map observation as a sparse semantic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v \in \mathcal{V}$ represents a map element (e.g., a segment of a lane divider) and stores attributes including its 2D centroid (x, y), heading θ, length ℓ, polyline/polygon geometry, and semantic label. Edges $e \in \mathcal{E}$ capture geometric relationships between nodes, such as spatial proximity or structural adjacency, commonly quantified using the Chamfer distance [8].

Type-aware affinity. Upon receiving a new observation, the cloud constructs two graphs: the local graph $\mathcal{G}^L = (\mathcal{V}^L, \mathcal{E}^L)$ from the agent and the intra-region reference graph $\mathcal{G}^G = (\mathcal{V}^G, \mathcal{E}^G)$ extracted from the current global map near the observation, with $n = |\mathcal{V}^L|$ and $m = |\mathcal{V}^G|$. MapGlue computes a type-aware affinity matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ composed of two components: node affinity and edge affinity.

The **node affinity** is computed only for semantically compatible pairs ($t_i^L = t_j^G$; otherwise set to $-\infty$ to prevent cross-type matching):

$$\mathbf{M}_{\text{node}}[i, j] = -(d_{ij} + \Delta\theta_{ij} + \Delta\ell_{ij}) + c$$

where $d_{ij} = |(x_i, y_i) - (x_j, y_j)|_2$ is the Euclidean centroid distance, $\Delta\theta_{ij} = |\arctan 2(\sin(\theta_i - \theta_j), \cos(\theta_i - \theta_j))|$ is the heading difference, $\Delta\ell_{ij} = |\ell_i - \ell_j|$ is the length difference, and c is a bias constant. The **edge affinity** captures structural consistency by comparing average intra-class neighbor distances between nodes:

$$\mathbf{M}_{\text{edge}}[i, j] = -\lambda \cdot |\bar{d}_i^L - \bar{d}_j^G|$$

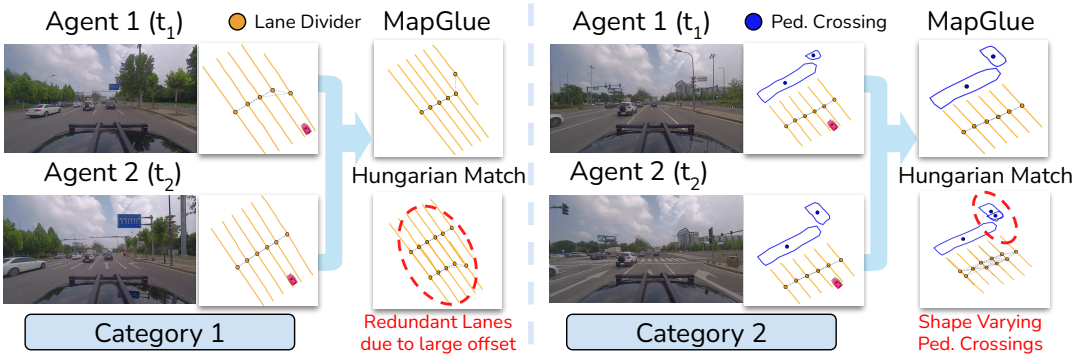


Fig. 2: Importance of Type-Aware Structure-Preserving Graph Matching: Category 1 (lane dividers) often exhibit larger positional offsets, whereas Category 2 (pedestrian crossings) appear in consistent locations but vary in shape. Dots represent visual graph node encodings. Hungarian matching [39], a standard distance-based algorithm without type or structure awareness, fails in both cases.

where $\bar{d}_i^L$ and $\bar{d}_j^G$ are the average distances of node i (resp. j) to its close neighbor nodes within threshold τ, and λ is a weighting hyperparameter. The full type-aware affinity is $\mathbf{M}[i,j] = \mathbf{M}_{\text{node}}[i,j] \oplus \mathbf{M}_{\text{edge}}[i,j]$ (element-wise sum), ensuring that MapGlue matches elements only when they share both semantic labels and compatible local geometry.

Greedy type-aware matching. Identifying the largest consistent subgraph between $\mathcal{G}^L$ and $\mathcal{G}^G$ is a Maximum Common Subgraph problem, which is NP-hard [29]. To balance accuracy and efficiency, we design a greedy algorithm that exploits the type-specific constraints from CATEGORY ❶ and ❷, iteratively selecting the highest-affinity node pair while enforcing per-type displacement and geometry constraints:

1) Find the highest-scoring unmatched node pair (i,j) in $\mathbf{M}$.
2) Check if (i,j) satisfies displacement and heading constraints for its semantic label t.
3) If valid, record the match, update the displacement buffer, and remove row i and column j from $\mathbf{M}$ to enforce one-to-one matching.
4) Repeat until no valid matches remain; add unmatched nodes to the merged graph $\mathcal{G}^{G'}$.

For CATEGORY ❶ (*e.g.,* lane dividers), matching permits global displacement subject to a displacement variance constraint and a heading angle constraint. For CATEGORY ❷ (*e.g.,* pedestrian crossings), locality constraints are stricter, but matched polygons are merged via union rather than rigid alignment to accommodate shape variations; nearby crossing nodes can additionally be merged. This greedy approach handles real-world misalignments and view inconsistencies efficiently while preserving semantic structure, enabling incremental fusion of asynchronous multi-agent observations.

Inter-region alignment. After intra-region merging produces $\mathcal{G}^{G'}$, MapGlue performs an inter-region alignment step to integrate it into the global map $\mathcal{G}$. We apply rigid Coherent Point Drift (CPD) [40] to align $\mathcal{G}^{G'}$ with the global map, leveraging its robustness in aligning structured 2D primitives, including polylines and polygons, under the residual small misalignments that remain after greedy matching has resolved larger displacements [41]. Unlike prior approaches [33] that rely on proximity-based instance-level heuristics, combining type-aware graph matching with point-level rigid registration [42], [43] enables `GraphMap` to resolve complex viewpoint discrepancies with higher accuracy and semantic fidelity, yielding a more consistent global map.

Algorithm 1: Selective Map Sensing with Feedback-Delayed Double-Ended Bandit

Input : Epsilon ϵ, feedback delay d, total frames T, Hyperparamater DataType

Initialize: $Q_{\text{upload}}, Q_{\text{skip}} \leftarrow 0$ `// SV rewards`

$N_{\text{upload}}, N_{\text{skip}} \leftarrow 0$ `// Action counts`

Global raster mask $\mathcal{M}_G \leftarrow \emptyset$

for $t \leftarrow 1$ **to** T **do**

  Extract local map graph $\mathcal{G}_t^L$

  `/* Agent-side: upload decision */`

  if *Random()* $< \epsilon$ **then**

    $a_t \leftarrow$ random choice from {upload, skip} `// Explore`

  else

    $a_t \leftarrow \arg\max(Q_{\text{upload}}, Q_{\text{skip}})$ `// Exploit`

  end

  if $a_t =$ *upload* **then**

    `/* Cloud-side: compute SV reward */`

    Rasterize $\mathcal{G}_t^L$ to get mask $\mathcal{M}_t$

    `/* Calculate fraction of` $\mathcal{M}_t$ `not in/overlaps with` $\mathcal{M}_G$ `*/`

    NewArea, Consistency $\leftarrow \frac{|\mathcal{M}_t \setminus \mathcal{M}_G|}{|\mathcal{M}_t|}, \frac{|\mathcal{M}_t \cap \mathcal{M}_G|}{|\mathcal{M}_t|}$

    $SV_t \leftarrow \alpha$NewArea $+ \beta$Consistency; $\mathcal{M}_G \leftarrow \mathcal{M}_G \cup \mathcal{M}_t$

    Schedule SV_t as feedback r_t

  end

  `/* Agent-side: receive feedback */`

  if *feedback* r_t *is available* **then**

    $a_{t-1} \leftarrow$ action taken at time $t-1$; $N_{a_{t-1}} \leftarrow N_{a_{t-1}} + 1$

    Freshness $\leftarrow \exp(\Delta t) + \exp(\Delta d)$ where Δt is the time since the agent's last upload, Δd is the spatial displacement from the last upload location

    $Q_{a_{t-1}} \leftarrow Q_{a_{t-1}} + \frac{1}{N_{a_{t-1}}}(r_{t-1} + \gamma \cdot \text{Freshness} + \delta \cdot \text{DataType} - Q_{a_{t-1}})$

  end

end

C. Selective Map Sensing

To efficiently manage network bandwidth and cloud computation in large-scale collaborative HD map construction, `GraphMap` incorporates a Sensing Value (SV)-guided *Selective Map Sensing* (SMS) mechanism. This strategy determines whether an agent should upload its local observation to the cloud, optimizing data transmission so that only information with significant utility for global map construction is processed, thereby reducing wireless network bandwidth requirements and computation load. The design of our SV function is motivated by three key observations: (1) Global map construction must balance exploration (new

area coverage) and exploitation (consistency improvement); (2) Redundant uploads—especially from stationary or slow-moving agents—should be penalized; (3) Agents differ in their mapping value. Static infrastructure sensors initially provide high coverage and consistency, but their marginal utility for mapping unchanged areas diminishes over time. Vehicles, on the other hand, continuously contribute dynamic updates and capture new information.

Based on these insights, we define the Sensing Value of an observation $\mathcal{G}^L$ as:

$$SV(\mathcal{G}^L) = \alpha \cdot NewArea(\mathcal{G}^L) + \beta \cdot Consistency(\mathcal{G}^L) + \gamma \cdot Freshness(\mathcal{G}^L) + \delta \cdot DataType(\mathcal{G}^L) \quad (1)$$

with scalar weights $\alpha+\beta+\gamma+\delta = 1$ tunable based on system priorities. Each term captures a distinct utility dimension:

- **NewArea**$(\mathcal{G}^L)$ quantifies the fraction of $\mathcal{G}^L$ covering previously unmapped regions, encouraging spatial exploration of new areas.
- **Consistency**$(\mathcal{G}^L)$ measures how well $\mathcal{G}^L$ aligns with the current global map—rewarding observations that reinforce map coherence while penalizing those that are perfectly consistent and thus unlikely to contribute new information.
- **Freshness**$(\mathcal{G}^L)$ penalizes redundant uploads from slow-moving or static agents by incorporating time elapsed and spatial displacement since the agent's last upload, computed locally via GPS/IMU, favoring non-repetitive updates.
- **DataType**$(\mathcal{G}^L)$ assigns source-dependent weights as hyper-parameters (e.g., fixed infrastructure vs. dynamic vehicles), capturing the heterogeneity and time-varying mapping value of different agent types.

While Consistency and Freshness both relate to temporal characteristics, they operate at different layers: Consistency is evaluated cloud-side against the current global map, whereas Freshness is computed agent-side from local movement and timestamps. This separation forms the core of our *double-ended* SV design. Crucially, cloud-side feedback can override a freshness penalty as an agent that recently uploaded may still observe a new area or reveal previously occluded geometry, where the combined cloud-side signal outweighs the freshness penalty and justifies the upload.

To schedule uploads across time and agents, `GraphMap` formulates the SMS policy using a *feedback-delayed, double-ended* ϵ-greedy multi-armed bandit (MAB) algorithm [44], [45], [46]. Each agent (or potential data stream) is treated as an arm, and its reward corresponds to the SV of its observation. With probability $1-\epsilon$, each agent exploits known high-SV behavior from past feedback; with probability ϵ, it explores by making random decisions to account for uncertainty or unobserved changes, preventing the system from getting stuck in local optima.

The *double-ended* nature ensures SV computation is distributed: agents locally compute DataType (a static hyperparameter based on their role) and Freshness (from GPS/IMU movement and time since last upload), while the cloud computes NewArea (by comparing the observation against the global coverage mask) and Consistency (by rasterizing and aligning with existing fused segments). The *delayed feedback* setting reflects the practical reality that NewArea and Consistency can only be assessed after MapGlue processes the upload, making the MAB formulation particularly suitable for asynchronous, multi-agent scenarios. Full details are in Algorithm 1. In our experiments, we empirically set α=0.4, β=0.4, γ=0.1, δ=0.1, ϵ=0.15, and feedback delay d=2 frames, finding this strikes a good balance between exploration and exploitation.

IV. Implementation

We implement `GraphMap` using both Python and C++. We implement an emulator of crowd-sourced multi-agent global map construction. The emulator accepts pre-recorded sensor traces of different modalities, including camera images, LiDAR point clouds and 3D poses of vehicles over consecutive frames. Emulated vehicles run MapTRv2 [13] as their online local HD map detector, and upload the vectorized detections via SMS policy. We run all our experiments on an Ubuntu 20.04 machine with a 64-core CPU and 128 GB of memory. To reflect real vehicle-cloud communication, we use real-world cellular and V2X [47], [48] network traces collected during driving and emulate changing bandwidth while simultaneously replaying the sensor data streams. The bandwidth statistics of the traces are 13.5 $\pm$ 10.4 Mbps, with 16.6 $\pm$ 5.4 ms of latency and 4.5 $\pm$ 12.7% of packet loss. In our evaluation, we use UDP for uploading the vectorized local maps, as it is the primary underlying transport layer protocol for vehicular applications [49], [48].

Training details. We train a state-of-the-art online local HD model, MapTRv2 [13] with 4 NVIDIA A100 GPUs with a batch size of 8. We adopt ResNet50 [50] as the image backbone and PointPillar [51] as the LiDAR backbone. Other training details are consistent with [14].

V. Evaluation

A. Experiment Setup

Motivation for using multi-agent datasets. Unlike prior work that evaluates local HD map models on single-vehicle datasets such as nuScenes [52] or Argoverse [53], our goal is to evaluate the end-to-end performance of collaborative, vectorized global HD map construction across asynchronous and spatially distributed data. Multi-vehicle datasets are essential to capture key properties such as agent heterogeneity, spatial overlap, and asynchronous sensing. Therefore, we use both real-world and synthetic multi-agent datasets to faithfully emulate the crowdsourced global map construction scenario.

Datasets and evaluation metrics. We evaluate `GraphMap` on two real-world multi-agent datasets: DAIR-V2X [30] and V2XPnP [31], and one synthetic dataset: OPV2V [28]. DAIR-V2X includes one vehicle and one infrastructure agent at six intersections, each with a front-view RGB camera, with effective map perception ranges of $[0, 30]$m (vehicle) and $[0, 75]$m (infrastructure) forward. V2XPnP involves four agents (two vehicles with four cameras each, two infrastructure nodes), with range $[-30, 30]$m. OPV2V provides photo-realistic CARLA [54] town-level scenes with 30–60 agents per scene equipped with cameras and LiDAR, with range $[-30, 30]$m on both axes. We use mean Average Precision

TABLE I: Performance on DAIR-V2X. GraphMap (C): camera only; (C+L): camera + LiDAR.

Map	AP_{ped}	AP_{div}	mAP	Method	Map	AP_{ped}	AP_{div}	mAP
06 300×300m	14.8	8.5	11.7	PolyMerge	10 300×300m	0.1	18.3	9.2
	20.2	2.7	11.4	FDC		31.0	0.3	15.7
	16.7	9.4	13.0	HBM		0.0	11.9	6.0
	87.5	30.3	58.9	GraphMap (C)		85.7	41.6	63.6
	87.5	**32.0**	**59.8**	GraphMap (C+L)		**85.7**	**45.7**	**64.7**
08 250×250m	8.0	13.7	7.4	PolyMerge	13 250×250m	61.8	8.1	39.9
	33.0	0.2	16.6	FDC		51.6	1.8	26.4
	0.0	22.9	11.5	HBM		57.5	6.3	31.9
	100.0	**50.2**	**75.1**	GraphMap (C)		**100.0**	8.8	54.4
	100.0	48.1	74.1	GraphMap (C+L)		**100.0**	**16.3**	**58.1**
09 300×300m	76.9	5.8	41.3	PolyMerge	16 250×250m	49.4	1.7	25.6
	59.4	4.6	32.1	FDC		45.0	1.2	23.1
	7.0	5.0	6.0	HBM		12.5	9.8	11.1
	92.3	7.7	50.0	GraphMap (C)		83.3	9.9	46.6
	92.3	**16.6**	**54.5**	GraphMap (C+L)		**83.3**	**14.0**	**52.8**

TABLE II: Performance on V2XPnP. UAR: Urban Arterial Road; UE: Urban Expressway.

Map	AP_{bound}	AP_{div}	mAP	Method	Map	AP_{bound}	AP_{div}	mAP
UAR 300×450m	24.0	13.1	18.5	PolyMerge	UE 250×600m	39.5	12.8	26.2
	13.9	14.6	14.3	FDC		39.0	12.1	25.6
	14.5	0.8	7.6	HBM		13.0	2.8	7.9
	31.2	**46.7**	**38.9**	GraphMap (C)		**52.8**	**37.6**	**45.2**
	28.9	45.7	37.3	GraphMap (C+L)		47.1	35.7	41.4

(mAP) [14] over Chamfer distance thresholds (1.0, 1.5, 2.0m). We evaluate pedestrian crossings and lane dividers on DAIR-V2X and OPV2V, and lane dividers and road boundaries on V2XPnP based on dataset annotations.

Baselines. We compare GraphMap against several baselines, which pair the vehicle-side MapTRv2 [13] model with different state-of-the-art (SOTA) cloud-side fusion strategies.

• **PolyMerge** [33]: A SOTA rule-based method merging vector polylines via spatial proximity and geometric heuristics.

• **Fréchet Distance Clustering (FDC)** [55]: A strong representative of trajectory similarity-based clustering approaches for grouping and consolidating similar map elements based on their intrinsic shapes, often considered a more sophisticated alternative to simple proximity. These metrics are frequently benchmarked in trajectory analysis [56], [57].

• **Heatmap Buffer Merge (HBM)**: Utilizes an intermediate raster representation, converting vectors to heatmaps, buffering for merging [35], then re-vectorizing [58]. Raster-based fusion followed by vectorization is a technique explored in various map works [35], [19].

We train distinct MapTRv2 detectors for vehicle agents and infrastructure agents to handle heterogeneity. To evaluate the impact of sensor modality, we create two variants of GraphMap: GraphMap (C) using camera-only local HD map detectors, and GraphMap (C+L) using both camera and LiDAR sensors for richer environmental perception. All models across baselines and GraphMap use identical training supervision and configurations for a fair comparison.

On GlobalMapNet [19]. GlobalMapNet targets a different setting from ours: it accumulates observations from a single ego vehicle over time and fuses a clipped global map prior into the local BEV detector. Its single-agent, on-device design is not directly applicable to spatially distributed multi-agent mapping. We therefore adopt PolyMerge, FDC, and HBM as baselines due to their better applicability to multi-agent fusion, while extending GlobalMapNet's map merging logic to multi-agent settings is left for future work.

TABLE III: Performance on OPV2V town-level global HD map construction.

Map	AP_{ped}	AP_{div}	mAP	Method	Map	AP_{ped}	AP_{div}	mAP
04 35 agents	73.1	24.3	48.7	PolyMerge	06 53 agents	66.4	40.6	53.5
	32.8	16.7	24.8	FDC		53.2	19.6	36.4
	30.3	7.8	19.1	HBM		75.0	24.8	49.9
	92.8	35.7	64.2	GraphMap (C)		78.5	45.5	62.0
	92.8	**38.0**	**65.4**	GraphMap (C+L)		**85.7**	**45.6**	**65.7**
05 60 agents	48.3	25.5	36.9	PolyMerge	10 30 agents	58.0	51.7	54.8
	22.6	17.7	20.2	FDC		35.2	16.9	26.1
	42.5	16.1	29.3	HBM		65.8	40.2	53.0
	86.8	33.0	59.9	GraphMap (C)		87.5	46.8	67.2
	86.8	**33.2**	**60.1**	GraphMap (C+L)		**93.7**	**52.9**	**73.3**

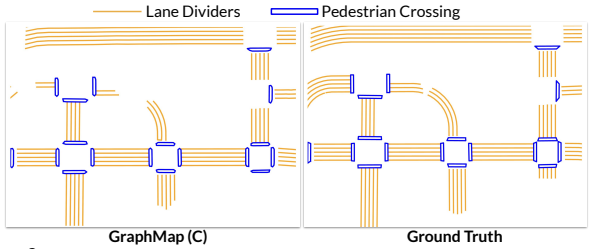


Fig. 3: A multi-intersection map built by GraphMap in Town05.

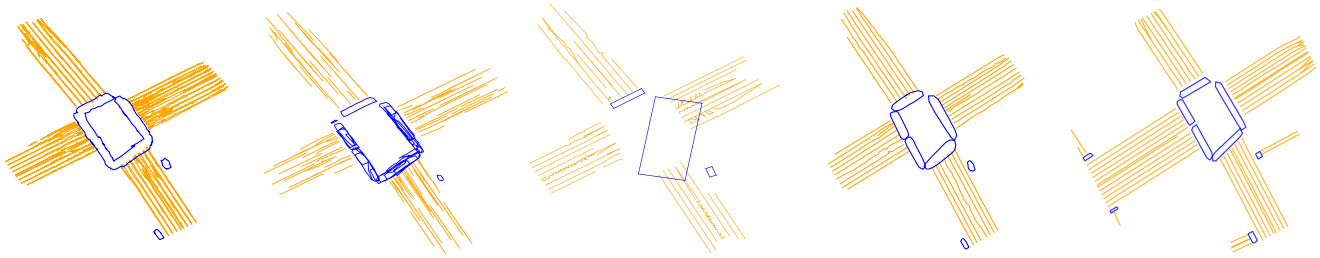
B. End-to-end Global Vectorized Map Results

We first systematically evaluate the end-to-end performance of GraphMap on global vectorized HD map construction across multiple real-world intersections, road segments, and simulated town-level maps.

Global vectorized HD map accuracy. Table I and Table II present the results across multiple global intersections and road segments from the DAIR-V2X and V2XPnP datasets. GraphMap significantly outperforms all baselines across all maps. On DAIR-V2X, GraphMap (C+L) achieves 25–50% higher mAP over the best baseline; Map-08 and Map-10 reach mAPs of 74.1 and 64.7 vs. 16.6 and 15.7 for FDC. Map-13 achieves perfect pedestrian crossing detection. Adding LiDAR improves lane divider alignment in DAIR-V2X (single front camera), but shows minimal gain on V2XPnP where multi-camera vehicles already capture sufficient local information.

Town-level map results on OPV2V. Table III shows GraphMap consistently outperforms all baselines across all four town-level maps in CARLA [54]. With 35 agents in Town04, GraphMap (C+L) achieves 65.4 mAP, improve by 16.7 mAP compared to the best baseline; with 60 agents in Town05, GraphMap leads by over 60%. In Town06 and Town10, the gains exceed 22%. These results confirm that GraphMap scales effectively with agent count and map complexity. Fig. 3 illustrates structural continuity across a large multi-intersection map.

Visualization of vectorized global map construction. Fig. 4 shows an example of global HD map construction results using different fusion methods. All three baselines fail to produce accurate global maps: PolyMerge produces smooth lane dividers but is distorted by outlier pedestrian crossing observations, while FDC and HBM suffer from broader fusion errors. GraphMap produces fine-grained, accurate maps by resolving viewpoint misalignments via type-aware graph matching and point-set registration.



(a) HBM (b) FDC (c) PolyMerge (d) GraphMap (C) (e) Ground Truth

Fig. 4: Comparison of HBM, FDC, PolyMerge, GraphMap, and Ground Truth on DAIR-V2X Map 06. Orange lines represent lane dividers, and blue polygons represent pedestrian crossings.

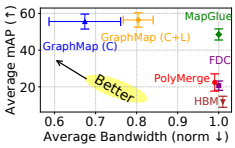


Fig. 5: Map accuracy *vs.* Bandwidth saving.

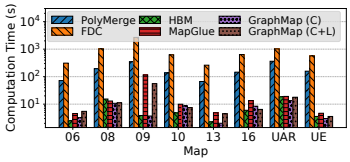


Fig. 6: Computation cost of global map construction.

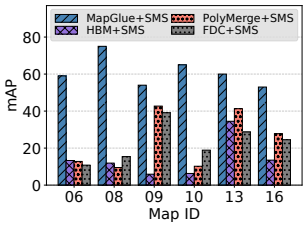


Fig. 7: Impact of MapGlue fusion.

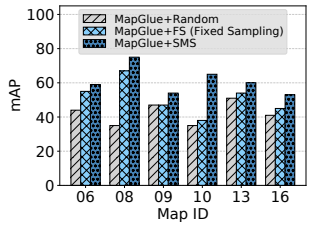


Fig. 8: Impact of SMS strategy.

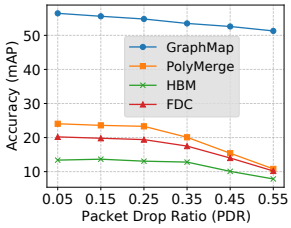


Fig. 9: Impact of PDR.

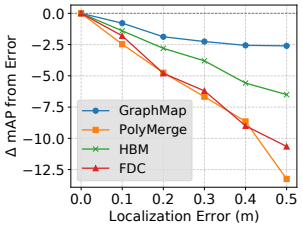


Fig. 10: Impact of pose error.

Global map accuracy vs. bandwidth efficiency. Fig. 5 compares different global map construction approaches in terms of average mAP and normalized communication bandwidth usage. We observe that both GraphMap (C) and GraphMap (C+L) substantially outperform all baselines in the accuracy-bandwidth tradeoff. GraphMap reduces communication load by 25–33% and still achieves 2–3× higher average mAP. This is attributed to our SMS mechanism, which filters redundant and low-quality uploads. Notably, combined with the SMS, GraphMap even achieves a 5% more accurate improvement compared to MapGlue (our ablated variant with full upload but graph-based fusion) while reducing the bandwidth requirement by over 25%. This interesting observation highlights the effectiveness of our selective data transmission in that it *not only saves network bandwidth* but also *avoids redundant and low-quality data from being uploaded* to reduce global map construction accuracy.

Computation efficiency. Fig. 6 shows the total computation time required to construct the global map. GraphMap achieves significantly lower compute time, due to its lightweight sparse graph representation and efficient matching algorithm. GraphMap (C) is 3× faster than PolyMerge and over 10× faster than FDC, which performs clustering over full pairwise trajectory distances.

C. Ablation Study

Next, we conduct an ablation study to dissect the contributions of key components of GraphMap, and to show how our design enables robustness against environmental noise and achieves efficient, lightweight runtime performance. We run all ablation experiments on the DAIR-V2X dataset.

Impact of GraphMap's key design component. To study the effectiveness of GraphMap's core map fusion algorithm, MapGlue, we replace the map fusion algorithm with other baselines while keeping the SMS for bandwidth saving. MapGlue improves 37.2 absolute mAP over all baselines (Fig. 7), reinforcing the need for a type-aware, structure-preserving fusion algorithm. To study the effectiveness of GraphMap's SMS module (§III-C), we keep the MapGlue algorithm for map fusion but replace the SMS with two baseline strategies that match the bandwidth savings of SMS: (1) random frame upload sampling (Random); (2) fixed frame-rate sampling (FS). Fig. 8 shows that SMS indeed selects informative data to upload for building the global map accurately, improving over the baselines by an average of 12.0 mAP in accuracy with the same bandwidth usage.

Robustness to environmental noises. We evaluate GraphMap under two critical and realistic types of environmental noises: *network packet loss* and agent *localization error*. Fig. 9 and 10 reveal that GraphMap is more robust under higher noises compared to baselines, as the accuracy drop of GraphMap is more graceful (<3% for 55% of packet drop rate and 0.5m of localization error). GraphMap's resilience to network packet loss comes from the SMS strategy because when a frame gets scheduled to be uploaded and gets lost, the agent will redo the upload for the next observation. GraphMap is also robust to localization noises because MapGlue (§III-B) takes drifts of map elements into account in the first place. Combined with CPD registration, spatially misaligned components still get fused accurately into the global map, thus enhancing GraphMap's performance under high pose errors.

VI. Concluding Remarks

In this work, we present GraphMap, a scalable and resource-efficient framework for global HD map construction

using crowdsourced observations from heterogeneous agents. Departing from dense, ego-centric fusion, `GraphMap` adopts a graph-based sparse fusion paradigm that enables robust global stitching. `GraphMap` adopts a selective sensing mechanism to determine valuable observations to upload, balancing map coverage, consistency, and network resource usage. Extensive experiments demonstrate that `GraphMap` significantly improves the accuracy of global HD map construction while reducing communication and computation overhead. These results validate our core insight: *sparse, selective, and semantically structured* collaboration enables scalable global HD mapping in large, dynamic environments.

REFERENCES

[1] "The waymo driver handbook: How our highly-detailed maps help unlock new locations for autonomous driving," https://waymo.com/blog/2020/09/the-waymo-driver-handbook-mapping, 2021.

[2] M. Developers, "Mobileye: Driver assist and autonomous driving technologies," 2025. [Online]. Available: https://www.mobileye.com/

[3] "Putting our robots on the map: The calibration, localization, and mapping process that helps zoox robotaxis find their way," https://tinyurl.com/55va3smm, 2025.

[4] "Opendrive format specification," https://www.asam.net/standards/detail/opendrive/, 2025.

[5] "Navigation data standard (nds)," https://nds-association.org/, 2025.

[6] R. Xu *et al.*, "CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers," *arXiv:2207.02202*, 2022.

[7] C. Chen *et al.*, "Trans4Map: Revisiting holistic bird's-eye-view mapping from egocentric images to allocentric semantics with vision transformers," in *Proc. WACV*, 2023, pp. 4013–4022.

[8] Q. Li *et al.*, "HDMapNet: An online hd map construction and evaluation framework," in *Proc. ICRA*, 2022, pp. 4628–4634.

[9] Y. Liu *et al.*, "VectorMapNet: End-to-end vectorized hd map learning," in *Proc. ICML*, 2023, pp. 22 352–22 369.

[10] Z. Jiang *et al.*, "P-mapnet: Far-seeing map generator enhanced by both sdmap and hdmap priors," *IEEE RA-L*, 2024.

[11] B. Yu *et al.*, "Metis: Crowdsourced lane line map construction," in *ACM BuildSys*, 2025.

[12] H. Lyu *et al.*, "Online high-definition map construction for autonomous vehicles: A comprehensive survey," *J. Sensor Actuator Netw.*, vol. 14, no. 1, p. 15, 2025.

[13] B. Liao *et al.*, "MapTRv2: An end-to-end framework for online vectorized hd map construction," *Int. J. Comput. Vis.*, pp. 1–23, 2024.

[14] B. Liao, S. Chen *et al.*, "MapTR: Structured modeling and learning for online vectorized hd map construction," in *Proc. ICLR*, 2023.

[15] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IROS*, 2018, pp. 4758–4765.

[16] T. Shan *et al.*, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IROS*, 2020, pp. 5135–5142.

[17] T. Yuan *et al.*, "StreamMapNet: Streaming mapping network for vectorized online hd map construction," in *Proc. WACV*, 2024.

[18] Y. Hu *et al.*, "Where2comm: Communication-efficient collaborative perception via spatial confidence maps," *NeurIPS*, 2022.

[19] A. Shi *et al.*, "GlobalMapNet: An online framework for vectorized global hd map construction," *arXiv:2409.10063*, 2024.

[20] Q. Liu, Y. Zhang, and H. Wang, "EdgeMap: Crowdsourcing high definition map in automotive edge computing," in *Proc. IEEE ICC*, 2022, pp. 4300–4305.

[21] A. Stoven-Dubois *et al.*, "A collaborative framework for high-definition mapping," *Proc. ITSC*, 2019.

[22] F. Immel *et al.*, "HD map generation from noisy multi-route vehicle fleet data on highways with expectation maximization," in *Proc. IEEE IV*, 2023.

[23] P. Chen *et al.*, "MapCVV: On-cloud map construction using crowdsourcing visual vectorized elements towards autonomous driving," *IEEE Robot. Autom. Lett.*, 2024.

[24] Q. Chen *et al.*, "F-Cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds," in *Proc. ACM/IEEE Symp. Edge Comput.*, 2019, pp. 88–100.

[25] Q. Zhang *et al.*, "Robust real-time multi-vehicle collaboration on asynchronous sensors," in *Proc. MobiCom*, 2023, pp. 1–15.

[26] Y. Lu *et al.*, "Robust collaborative 3D object detection in presence of pose errors," in *Proc. ICRA*, 2023, pp. 4812–4818.

[27] Z. Li *et al.*, "BEVFormer: Learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers," *TPAMI*, 2024.

[28] R. Xu *et al.*, "OPV2V: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication," in *Proc. ICRA*, 2022, pp. 2583–2589.

[29] H.-C. Ehrlich and M. Rarey, "Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review," *WIREs Comput. Mol. Sci.*, vol. 1, no. 1, pp. 68–79, 2011.

[30] H. Yu *et al.*, "DAIR-V2X: A large-scale dataset for vehicle-infrastructure cooperative 3D object detection," in *Proc. CVPR*, 2022.

[31] Z. Zhou *et al.*, "V2XPnP: Vehicle-to-everything spatio-temporal fusion for multi-agent perception and prediction," *arXiv:2412.01812*, 2024.

[32] J. Yu *et al.*, "ScalableMap: Scalable map learning for online long-range vectorized hd map construction," *arXiv:2310.13378*, 2023.

[33] M. Sayed *et al.*, "PolyMerge: A novel technique aimed at dynamic hd map updates leveraging polylines," in *Proc. ICAR*, 2023, pp. 94–99.

[34] J. Yang *et al.*, "HisTrackMap: Global vectorized high-definition map construction via history map tracking," *arXiv:2503.07168*, 2025.

[35] J. Chen *et al.*, "MapTracker: Tracking with strided memory fusion for consistent vector hd mapping," in *Proc. ECCV*, 2024, pp. 90–107.

[36] S. Chen *et al.*, "VMA: Divide-and-conquer vectorized map annotation system for large-scale driving scene," *arXiv:2304.09807*, 2023.

[37] S. Shi *et al.*, "VIPS: Real-time perception fusion for infrastructure-assisted autonomous driving," in *Proc. MobiCom*, 2022, pp. 133–146.

[38] Z. Ni *et al.*, "Self-localized collaborative perception," *arXiv:2406.12712*, 2024.

[39] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Q.*, vol. 2, no. 1-2, pp. 83–97, 1955.

[40] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE TPAMI*, vol. 32, no. 12, pp. 2262–2275, 2010.

[41] M. Cho *et al.*, "ADoPT: Lidar spoofing attack detection based on point-level temporal consistency," *arXiv:2310.14504*, 2023.

[42] X. Huang *et al.*, "A comprehensive survey on point cloud registration," *arXiv:2103.02690*, 2021.

[43] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Trans. Robot.*, vol. 37, no. 2, 2020.

[44] R. Sutton *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[45] D. E. Koulouriotis and A. Xanthopoulos, "Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems," *Appl. Math. Comput.*, vol. 196, no. 2, pp. 913–922, 2008.

[46] F. Azizi *et al.*, "MIX-MAB: Reinforcement learning-based resource allocation algorithm for LoRaWAN," in *Proc. IEEE VTC*, 2022.

[47] A. Narayanan *et al.*, "A variegated look at 5G in the wild: performance, power, and QoE implications," in *Proc. ACM SIGCOMM*, 2021.

[48] R. Mo *et al.*, "SEE-V2X: C-V2X direct communication dataset: An application-centric approach," in *Proc. SenSys*, 2025, pp. 305–311.

[49] F. Bai *et al.*, "Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers," in *Proc. MobiCom*, 2010, pp. 329–340.

[50] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[51] A. H. Lang *et al.*, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. CVPR*, 2019, pp. 12 697–12 705.

[52] H. Caesar *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. CVPR*, 2020, pp. 11 621–11 631.

[53] B. Wilson *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," *arXiv:2301.00493*, 2023.

[54] "CARLA: Open-source simulator for autonomous driving research," https://carla.org/, 2021.

[55] P. C. Besse *et al.*, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3306–3317, 2016.

[56] J. Gudmundsson and N. Valladares, "A GPU approach to subtrajectory clustering using the Fréchet distance," in *Proc. ACM SIGSPATIAL*, 2012, pp. 259–268.

[57] S.-W. Cheng and H. Huang, "Curve simplification and clustering under Fréchet distance," in *Proc. ACM-SIAM SODA*, 2023, pp. 1414–1432.

[58] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica*, vol. 10, no. 2, pp. 112–122, 1973.