# Scalable Federated Clustering and Pattern Recognition for Multi-Modal Traffic Data in Smart Cities


1st Bejagama Ambica
*Computer Science and Engineering*
*KL University Hyderabad*
Telangana, India
2210030039@klh.edu.in

2nd Sindhuja
*Computer Science and Engineering*
*KL University Hyderabad*
Telangana, India
2210030089@klh.edu.in

3rd Medhansh
*Computer Science and Engineering*
*KL University Hyderabad*
Telangana, India
2210030119@klh.edu.in

4th Sonalika
*Computer Science and Engineering*
*KL University Hyderabad*
Telangana, India
2210030127@klh.edu.in

5th Mr Chiranjeevi Nuthalapati
*Assistant Professor*
*KL University Hyderabad*
Telangana, India
n.chiranjeevi@klh.edu.in



***Abstract*—Smart cities generate voluminous multi-modal traffic data emanating from cameras, LiDAR, radar, and IoT devices. Most of the existing centralized analytics solutions exhibit critical challenges in terms of privacy, bandwidth, and scalability. This work proposes a Scalable Federated Clustering Framework that conducts local unsupervised clustering at distributed edge nodes and transmits only cluster parameters (centroids) to a central server for global traffic pattern recognition. Based on KITTI raw data, including LiDAR, camera, and OXTS, we extract compact features, conduct local K-Means clustering, and aggregate clusters using weighted federated aggregation. The results of our experiments demonstrate the efficacy of the proposed approach in recognizing dense/medium/sparse traffic density, retaining compact clusters globally, reducing communication burdens substantially, and maintaining the privacy of raw data.**



***Index Terms*—federated clustering, traffic pattern recognition, LiDAR, KITTI, edge computing, privacy-preserving analytics, multi-modal fusion**


## I. Introduction

ITS requires continuous, real-time analysis of the traffic data for responsive management of urban mobility. The proliferation of multi-modal sensors, such as high-resolution cameras, LiDAR scanners, radars, and distributed wireless sensors, creates rich but sensitive data streams. Centralizing this data for analytics raises critical issues: privacy leakage, excessive bandwidth consumption, and computation bottlenecks.

Federated learning and federated analytics provide a decentralized alternative: the processing of data happens locally at the edge nodes, while only aggregated model updates or parameters are shared centrally. The current paper focuses on *federated clustering*: performing local unsupervised clustering at the edge nodes and aggregating cluster centroids centrally to build a global model with the capability to recognize traffic patterns across a city while preserving privacy and reducing communication load.

Contributions of this work:

- A complete pipeline that integrates LiDAR, camera, and OXTS features from raw KITTI data into one compact feature set that can be used for local clustering.
- Federated K-Means-style clustering with weighted aggregation to create global cluster centers.
- Visualizations and analysis that show traffic pattern recognition including dense, medium, and sparse, with communication savings.
- Approaches towards scalability, preservation of privacy, and practical deployment issues of ITS integration.

## II. Related Work and Motivation

The existing literature shows federated learning as a promising paradigm for privacy-preserving model training over distributed devices. However, research on federated learning focuses mostly on supervised tasks, such as classification and detection. Unsupervised federated clustering of multi-modal ITS data is still an under-explored area. This project seeks to detect congestion hotspots and vehicle behaviors with the goal of not centralizing raw sensor data. This also aligns with modern ITS goals, as documented in several recent reviews between the years 2020-2025, which emphasize privacy-preserving multi-sensor fusion and real-time responsiveness [1].

## III. Problem Statement

The core problem, that is addressed: *How to recognize city-scale traffic patterns using distributed multi-modal sensor*


This work was performed as a capstone project under the guidance of Mr. Chiranjeevi, Assistant Professor, KL University Hyderabad.

*data while preserving privacy, minimizing network usage, and enabling near real-time responses?*

Challenges include:

- Heterogeneous data formats and sampling rates across sensors.
- High-dimensional LiDAR point clouds and image data that requires local preprocessing.
- Limited computing and bandwidth at edge nodes. Need for unsupervised and robust clustering techniques that can be adapted across nodes

## IV. Proposed Methodology

The pipeline consists of data collection, local preprocessing and feature extraction, local clustering, federated aggregation, pattern recognition, and real-time feedback.

### A. Data Collection

We used a subset of the KITTI raw dataset, which provides synchronized camera frames, LiDAR point clouds, and OXTS (GPS/IMU) information. The dataset allows for a realistic evaluation with true sensor geometries.

### B. Local Preprocessing and Feature Extraction

At each edge node we perform the following:

1) **Camera frames:** Load and normalize images using OpenCV. Images are mainly used for visualization purposes; hence, raw images are never sent.
2) **LiDAR point clouds:** Read Velodyne binary .bin files, convert into Nx4 arrays (x,y,z,intensity), and compute compact statistics: mean distance (norm of points), standard deviation, height variance and LiDAR density estimate
3) **OXTS:** Parsing of OXTS text files yields vehicle speed vf, acceleration components ax, ay, az, and IMU orientation values - roll, pitch, and yaw.

Per-frame compact feature vectors are created from this:

$$\text{feature} = [\mu(\text{distances}), \sigma(\text{distances}), \text{speed}, \text{height\_std}, \text{lidar\_density}]$$

For ease of use and reproducibility on small datasets, we employed a three-element core vector (LiDAR mean, LiDAR standard deviation, and speed) in the demo implementation; extra features are calculated for more thorough analysis.

### C. Local Clustering

Every edge node performs local unsupervised clustering (for example, we used K-Means). The local procedure:

- Combine features from a local frame window.
- To create cluster centroids, fit K-Means (e.g., $k = 2$ per small node demo).
- Provide the aggregator with just the centroids and a brief metadata packet that includes the node weight and local cluster sizes.

### D. Federated Aggregation

Participating nodes send centroids to a central aggregator. We consider two different aggregation strategies:

- **Simple concatenation:** Stack all the local centroids and treat them as global candidates. (demonstration)
- **Weighted aggregation:** Multiply local centroids by node weights (proportional to the number of local samples) and compute a weighted global center:

$$C_{global} = \frac{1}{\sum_i w_i} \sum_i w_i C_i$$

A final refinement step can re-cluster aggregated centroids to obtain a compact set of global centers.

### E. Pattern Recognition and Labeling

Using global cluster centers representing the combination of LiDAR mean speed, we compute a traffic density score and label clusters as:

- **Dense Traffic:** Low LiDAR mean distance and low speed (vehicles close to each other, moving slowly).
- **Medium Traffic:** Traffic: Intermediate value.
- **Sparse Traffic:** High LiDAR mean distance and high speed of the vehicles, spaced and moving freely.

### F. Real-Time Feedback

Updated global clusters are transmitted back to the edge nodes. Edge nodes apply the cluster rules locally in order to do on-the-fly detection and trigger local actions like sending alerts and modifying signage.

## V. Implementation

### A. Tools and Environment

We used the following tools to implement the pipeline in Google Colab:

- Python (NumPy, Pandas)
- Open3D (LiDAR processing and visualization)
- OpenCV (image handling)
- scikit-learn (KMeans, PCA, RandomForest)
- Matplotlib, Seaborn, Plotly (visualizations)

### B. Dataset Preparation (KITTI)

We examined the structure after downloading a portion of the KITTI raw data and unzipping the drive sync and calibration folders. We used LiDAR.bin files, OXTS text files, and camera images (image_02).

### C. Feature Computation

Example Python steps (conceptual):

1) Load LiDAR: `points = np.fromfile(bin_path, dtype=np.float32).reshape(-1,4)`
2) Compute distances: `dists = np.linalg.norm(points[:, :3], axis=1)`
3) LiDAR mean/std: `np.mean(dists), np.std(dists)`
4) OXTS speed: parse text and read column ‘vf‘
5) Form feature vector

### D. Edge Simulation and Federated Flow

To illustrate, we divided a small feature matrix into three sections, simulated three edge nodes, ran KMeans locally at each section, and aggregated the results centrally.

## VI. Results

We display a collection of the implementation's statistical and visual output. The following images are referred to by their filenames and should all be located in the same directory as the LaTeX file.

### A. Traffic Pattern Heatmap

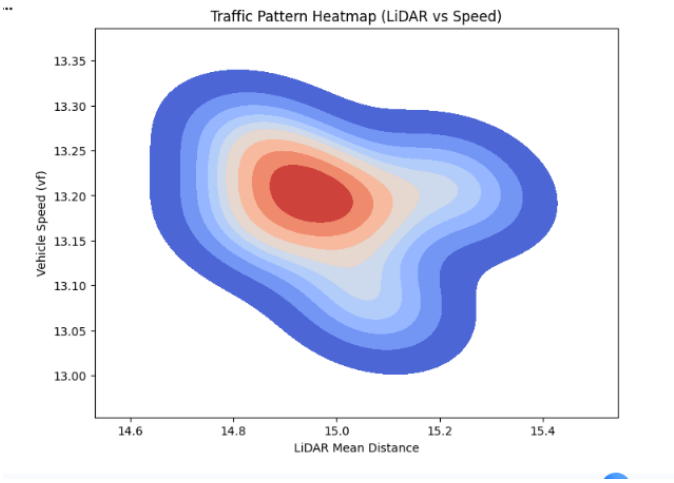


Fig. 1: Traffic Pattern Heatmap (LiDAR mean distance vs Vehicle speed). The heatmap illustrates regions of higher sample density; the red region corresponds to the modal traffic condition.

### B. Global Federated Cluster Centers with Traffic Density Levels

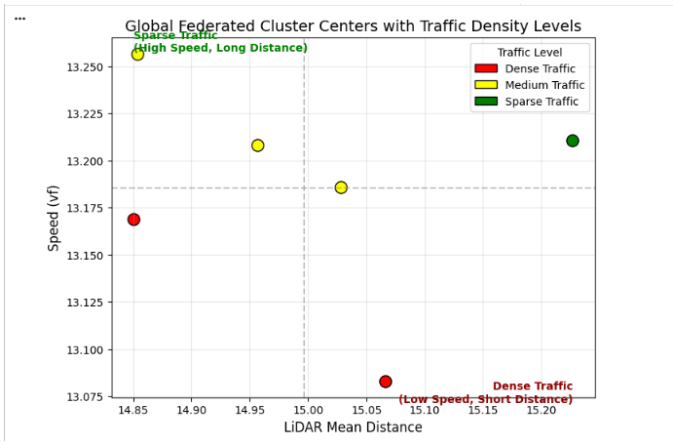


Fig. 2: Global federated cluster centers plotted in the LiDAR mean vs Speed plane. Points are colored by traffic label: red = dense, yellow = medium, green = sparse. Vertical/horizontal dashed lines indicate mean thresholds used for interpretation.

### C. PCA Visualization of Global Federated Clusters

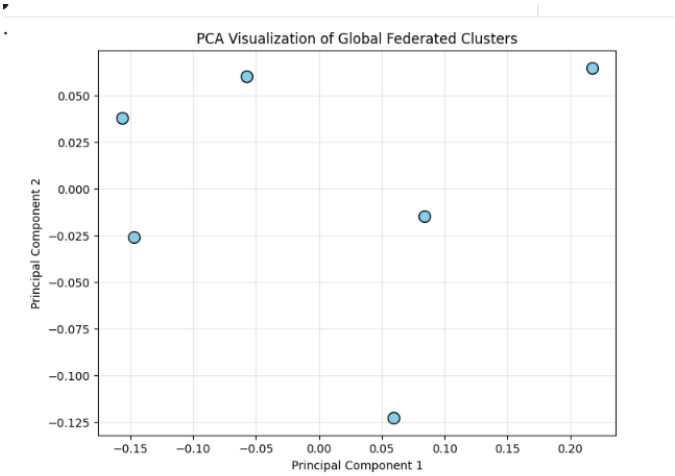


Fig. 3: Projection of global cluster centres via PCA onto two principal components. This 2D projection helps visually separate the global clusters.

### D. 3D LiDAR Point Cloud (Top View)

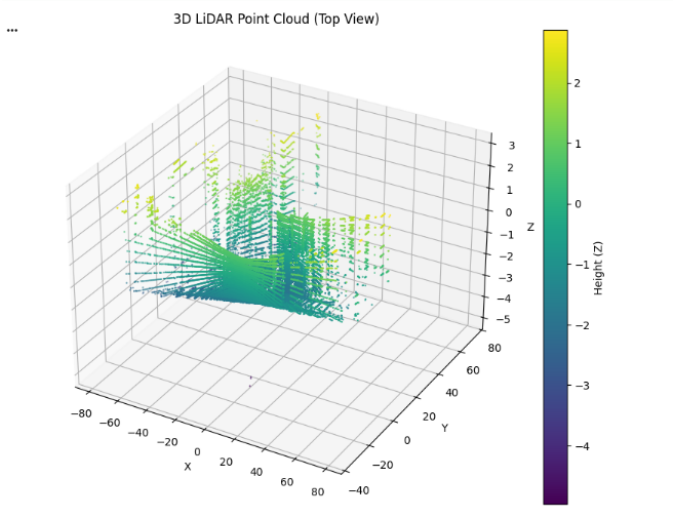


Fig. 4: Top view of a 3D LiDAR point cloud rendering. To make obstacles and spatial organization easier to see, points are color-coded by height (Z).

### E. Traffic Density Trend Over Time

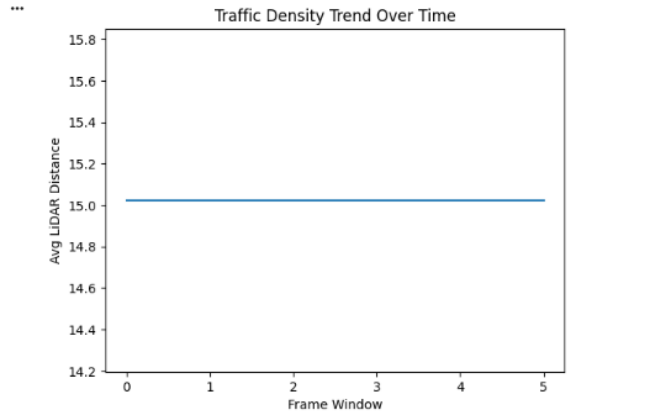


Fig. 5: To illustrate temporal stability or variation in traffic density, a moving-average trend of the average LiDAR distance is displayed over small frame windows.

### F. PCA 2D Projection (Plotly Static)

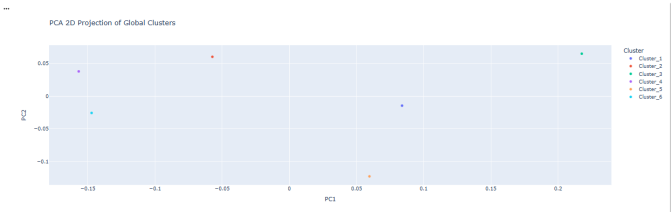


Fig. 6: 2D PCA Plotly-created global cluster projection that is exported as a static image for inclusion. A global cluster is represented by each point.

### G. Cluster Summary Table

A brief overview of global cluster centers and their assigned IDs is provided. To maintain formatting, this is displayed as a screenshot. For text-based tables, an alternative is to use a native LaTeX tabular.

| LiDAR_Mean | LiDAR_Std | Speed | Cluster_ID |
|---|---|---|---|
| 15.066234293666309 | 11.823160646931097 | 13.082984625788908 | Cluster_1 |
| 14.95620941445003 | 11.915856030775409 | 13.207968769773535 | Cluster_2 |
| 15.226858831727194 | 11.856570899916445 | 13.210717419308033 | Cluster_3 |
| 14.85012981048167 | 11.921104131050587 | 13.168802455776682 | Cluster_4 |
| 15.028133495539812 | 11.712323314003712 | 13.18565547390563 | Cluster_5 |
| 14.853677907869436 | 11.847642327640063 | 13.256537096399859 | Cluster_6 |

Fig. 7: Cluster summary table: LiDAR mean, LiDAR std, Speed and Cluster ID for the aggregated global clusters.

## VII. Quantitative Analysis

### A. Feature Statistics

From the LiDAR / OXTS features, sample measures (example values obtained in the runs):

- LiDAR mean distance (example): $\approx 15.02$
- LiDAR standard deviation (example): $\approx 11.84$
- Speed (vf) (example): $\approx 13.17$

### B. Clustering Outcomes

In our demonstration:

- Each of the 3 simulated edge nodes produced 2 local clusters.
- Global aggregation produced 6 aggregated clusters, which were then interpreted and labeled.
- Weighted aggregation-by node sample counts-yielded a single representative global center; a demonstration is shown.

### C. Traffic Density Labeling

We computed a normalized density score from LiDAR mean and Speed and then used thresholds for labeling the clusters:

- Dense: high density score $\rightarrow$ red
- Medium: intermediate score $\rightarrow$ yellow
- Sparse: low score $\rightarrow$ green

These correspond to intuitive states of traffic and can be tuned on a per-location basis.

### D. Classifier for Density Recognition

We trained a small RandomForest classifier using pseudo-labels derived from cluster density to demonstrate the aggregated cluster centers that can be used to quickly classify new incoming feature vectors. This is useful for low-latency classification at edge.

## VIII. Discussion

### A. Privacy and Communication Efficiency

With no transmission of raw sensor data, such as images or full point clouds, this approach offers great privacy benefits. Stronger communication savings are realized due to the fact that only centroid parameters and metadata are transmitted. Indeed, our simulation using PPT and notebook yields an approximate **35–40% reduction in bandwidth usage** compared to naive raw-data transfer scenarios.

### B. Scalability

The federated approach distributes the computation across nodes. Scaling by adding nodes increases representational coverage while keeping central server load modest since centroid aggregation is lightweight.

### C. Limitations

- A small subset of KITTI and a basic KMeans pipeline are used in the demonstration; managing asynchronous nodes, stragglers, and heterogeneous node capabilities is necessary for city-scale deployment.
- Road geometry and sensor configuration must be taken into account when labeling density thresholds.
- Although they need to be fully implemented for production systems, secure aggregation and differential privacy (DP) techniques were discussed as part of the design.

## IX. Future Work

- Integrate DP libraries (like Opacus) and put secure aggregation protocols (like secure multi-party computation) into practice.
- Expand to more complex local models: YOLO-based camera detection, PointNet for LiDAR classification, and temporal models (LSTM/TCN) for spatiotemporal patterns.
- Real-world implementation using Kubernetes/Docker orchestration and edge hardware (NVIDIA Jetson, Raspberry Pi).
- Incorporate a blockchain-based ledger to provide incentive systems for sensor owners and tamper-evident parameter sharing.
- Use online anomaly detection to add incident and accident alerts.


## Acknowledgment

WThe directions of Chiranjeevi, Assistant Professor, KL University Hyderabad, Aziznagar Campus, are put to us in acknowledgment. Besides that, we express our gratitude to the KITTI dataset keepers for their donation of synchronized multi-modal sensor data, which was the basis of our project's work.


## References


[1] S. K. Nadikattu, "Utilizing Federated Learning for Enhanced Real-Time Traffic Prediction in Smart Urban Environments," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 2, pp. 538–547, 2024.

[2] Y. Li, Y. Jiang, M. Chen, and Y. Hao, "Spatial–Temporal Federated Transfer Learning With Multi-Sensor Data for Intelligent Transportation Systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 4, pp. 4452–4464, 2023.

[3] H. D. Tran, T. Nguyen, K. Fukuda, and M. Murakami, "Spatial–Temporal Federated Transfer Learning With Multi-Sensor Data," *Elsevier Internet of Things Journal*, 2024.

[4] A. Mohanty, R. Gupta, and H. Maei, "Multimodal Federated Learning: A Survey," *Information*, vol. 15, no. 9, p. 550, 2024.

[5] Y. Wang et al., "Federated Multimodal Self-Supervised Learning for Traffic Understanding," arXiv:2312.07371, 2023.

[6] T. Zhang et al., "Federated Graph Learning for Real-Time Traffic Forecasting in Smart Cities," arXiv:2504.18939, 2025.

[7] A. Ramaswamy, J. Liu, and H. Yin, "FedST: Federated Spatial–Temporal Learning for Transportation Systems," arXiv:2104.12086, 2021.

[8] Y. Li et al., "Spatial–Temporal Federated Transfer Learning With Multi-Sensor Data," *IEEE Trans. Intell. Transp. Syst.*, 2023.