POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Vehicle-to-Vehicle Cooperative LiDAR Perception Based on Graph Neural Networks

Tesi di Laurea Magistrale in
Computer Science Engineering - Ingegneria Informatica

Vlad Marian Cimpeanu, 10606922

Advisor:
Prof. Matteo Matteucci

Co-advisors:
Lorenzo Cazzella

Academic year:
2023-2024

Abstract: The imminent rise of autonomous driving, slated for realization by 2030, promises a transformative era marked by enhanced safety, comfort, and operational efficiency. The journey from driver assistance to fully autonomous systems presents challenges, with LiDAR technology playing a crucial role. In this paper, we explore cooperative perception strategies for LiDAR point clouds, addressing challenges like occlusion and limited data sharing. Current cooperative perception works propose a framework in which each vehicle shares the position of the detected objects. These methods experience significant estimation errors and noise due to insufficient local observation. Other studies develop neural networks to compress and reconstruct entire point clouds, minimizing the reconstruction error. Even though these studies prove to be effective, the correct functioning of the algorithm requires its deployment on all connected vehicles. The purpose of this paper is to investigate an alternative approach where raw data can be transmitted without relying on the latter requirement. We design and test a point selection algorithm based on a graph neural network that aims to identify which points belonging to a point cloud acquired by a vehicle are worth to be transmitted to another vehicle. Our experiments are conducted in a simulated vehicular urban scenario relying on realistic LiDAR and Vehicle-to-vehicle communications simulators. Experimental results show that our algorithm can detect important areas that cannot be perceived by the receiver vehicle with mean 81% validation accuracy over different communication bandwidths, reducing the redundant transmitted data. Challenges in training convergence speed and hyperparameters search are acknowledged, suggesting avenues for further developments.

Key-words: cooperative perception, LiDAR, autonomous vehicles, point cloud segmentation, graph neural networks

1. Introduction

The imminent advent of autonomous driving, a technological milestone slated for realization by 2030, promises to usher in a transformative era characterized by substantial societal benefits, notably regarding safety, comfort, and operational efficiency. The Society of Automotive Engineers (SAE) [1] systematically categorizes this paradigm shift in transportation, distinguishing autonomous driving into six distinct levels. In the initial three levels (0-2), while maintaining complete control of the vehicle, human drivers find themselves assisted by

elementary functionalities such as lane-changing detection, blind spot warnings, and emergency braking. As we progress through levels 3 to 5, a visionary landscape emerges, wherein a fully driverless system eliminates the requirement for human intervention even in over-driving scenarios.

The current landscape of the self-driving revolution is marked by the development of Autonomous Vehicles (AVs) tailored for specific, controlled environments, including industrial fleets dedicated to logistics, farming, and operations within airport precincts. This nascent stage, however, is merely the precursor to a forthcoming decade that holds the promise of a monumental paradigm shift. A profound transformation awaits consumers as they are poised to reimagine their travel time, utilizing it for work-related activities, moments of relaxation, or accessing various forms of entertainment.

Within the intricate tapestry of sensing technologies pivotal for realizing autonomous driving, LiDAR emerges as a standout, earning recognition as the "eyes" of driverless vehicles [2]. This technology distinguishes itself through its markedly superior spatial resolution. However, the journey toward achieving seamless autonomy is accompanied by inherent challenges. These challenges include occlusion, a confined perception horizon from a limited field of view, and the lower density of data points in distant regions, as shown in Fig. 1. Failing to overcome these limitations can lead to dangerous situations: in the Hyundai competition, one of the AVs crashed because of bad weather conditions [3], Tesla's Autopilot failed to detect a white truck and crashed with it, killing the driver [4], and Google's AV collided with a bus during a lane change due to its inability to estimate the bus's speed [5].

To tackle these impediments head-on, a compelling solution emerges through vehicle-to-vehicle (V2V) communications to exchange sensory information between vehicles [7]. This exchange of information can enhance the perception of the environment for each vehicle: such process takes the name of *cooperative perception* [8, 9]. However, the pursuit of cooperative perception introduces its own set of challenges. For instance, a standard commercial LiDAR system with 64 laser diodes produces a staggering 2.8 million data points per second, covering a 360° horizontal field of view, a 26.8° vertical field of view, and a range extending beyond 70m in all directions. While this precise three-dimensional mapping capability is commendable, sharing even a fraction of this voluminous data imposes significant challenges due to the impracticality of transmitting raw data with existing communication technologies.

In this paper, we explore the cooperative perception of point clouds, aiming to uncover novel strategies to address the limited perception offered by LiDARs and communication constraints. We explore the potential application of a graph neural network, specifically opting for a Grid-GCN [10] in the context of point cloud segmentation. This aims to determine the points within a point cloud that merit transmission to another vehicle.

The main contributions of this paper are the following:

- We present a *novel deep learning-based cooperative perception method* that, differently from the existing cooperative perception approaches, proposes to use a graph neural network to identify which points belonging to a point cloud acquired by a vehicle are worth to be transmitted to another vehicle. This approach allows the transmission of valuable raw data without overloading the network.
- We develop a *simulation framework* for testing our cooperative perception algorithm. Differently from works, e.g., [11, 12], that have already provided a simulator based on SUMO [13] and CARLA [14], we integrate these simulators with the GEMV^2 [15] to simulate the V2V communication channel accounting for the geometry of the urban scenario.

The remainder of this paper is organized as follows. In Section 2, we give an overview of the state of the art for cooperative perception methodologies. Furthermore, we list and describe the advantages and disadvantages of the current techniques available for object detection and semantic segmentation on point cloud based data. Section 3 defines the system model taken into consideration. The problem formulation is provided in Section 4, while in Section 5, we propose our deep learning-based cooperative perception method. In Section 6, we specify the simulation framework and provide experimental results. Finally, the conclusions are drawn in Section 7.

2. Related works

In this section, we provide an overview of the related works. In Section 2.1, we list and discuss different approaches used in the literature to address cooperative perception. In Section 2.2, we investigate which are the most suitable methods and architectures that can take as input a point cloud. Finally, in Section 2.3, we describe in detail the properties of the model we recognize as most suitable for the cooperative perception scenario.

2.1. Cooperative perception using point clouds

In cooperative perception, vehicles acquire raw data through their sensors and share relevant information to increase environmental awareness. The pipeline of cooperative perception can have different implementations

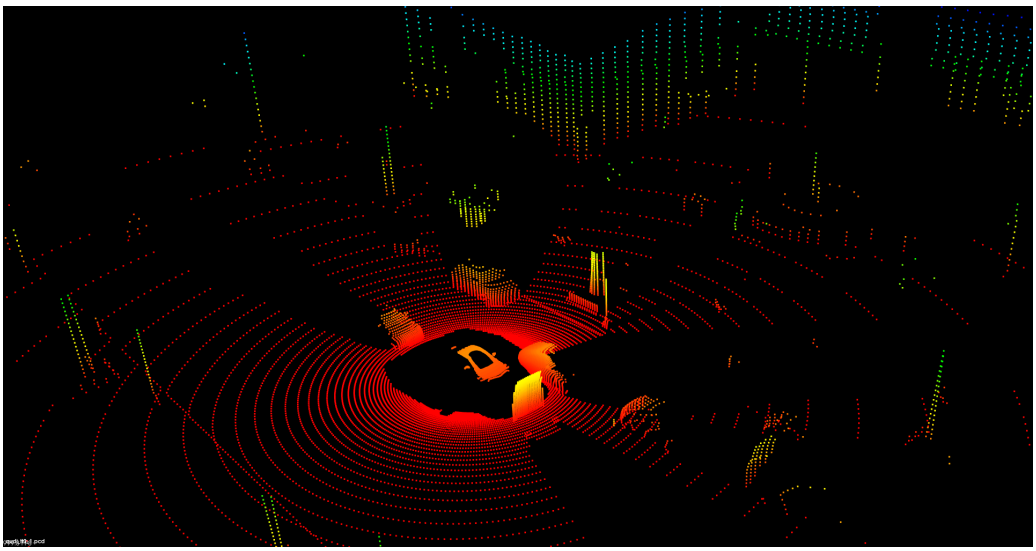


Figure 1: Example of point cloud from simulated LiDAR within a random vehicular scenario. Two sensing limitations of LiDAR can be seen here, i.e., the point cloud density decreases with the distance from the sensor, and dynamic obstacles (e.g., vehicles) can occlude regions of the environment (black regions). Different colors represent different point heights.

and differ in what information is shared. Based on the bandwidth constraints, the principal cooperative methodologies are late collaboration, early collaboration, and mixed collaboration. In this section, we discuss the most relevant works for each of them and analyze their advantages and disadvantages.

Late collaboration Late collaboration conducts cooperation in the output space, which encourages the fusion of the perception result output by each agent. As an example, each agent performs object detection tasks on their own perceived data and sends the detected objects' positions to the other agents. Aoki *et al.* [11] are the pioneers in this research area. The authors propose to model the cooperative perception task as a control problem and to solve it using a Deep Q Network (DQN) [17]. The agent's state is modeled as a grid, and a label is associated to each grid's cell to indicate if the agent has identified an object in that position. Furthermore, the receiver's spatial information and the communication channel information are added to the sender's state. For each cell, the agent can choose whether to send that cell to the paired agent. The agent receiving the new data returns a reward that indicates the level of satisfaction based on the quality and novelty of the received data. Even though this approach shows an overall perception improvement, scaling with the resolution is limited. Indeed, given that each grid's cell corresponds to an action, the number of possible actions is proportional to the level of grid resolution. Unfortunately, in a DQN, the number of discrete actions that must be explicitly represented grows exponentially with increasing action dimensionality.

Abdel *et al.* [18] propose to overcome this issue by replacing the DQN with a Branching Dueling Q Network (BDQ) [19]. In this architecture, the represented actions linearly increase with the action dimensionality. Moreover, they use the quadtree data structure to compress grid information cleverly, reducing bandwidth overload.

Even though late collaboration saves bandwidth, it is susceptible to agent positioning mistakes and experiences significant estimation errors and noise due to insufficient local observation.

Early collaboration Early collaboration occurs in the input space, where infrastructure and vehicles communicate unprocessed sensory data. The early collaboration approach compiles raw measurements from every vehicle and piece of infrastructure to support an integrated viewpoint. As a result, every car may carry out the subsequent processing and complete perception from a holistic viewpoint, which can effectively resolve the long-range and occlusion problems that arise in single-agent perception. With the aid of rich information, Arnold *et al.* [20] demonstrate how early fusion can be highly beneficial with respect to late fusion models in cooperative perception scenarios. The results demonstrate that early cooperative perception fusion can recall more than 95% of the objects, contrasting with the 30% for single-point perception in the most challenging scenarios. Nevertheless, exchanging raw sensory data necessitates extensive connection and can quickly overload the communication network with large amounts of data, which makes it impractical for most applications.

Mixed collaboration To benefit from both the early collaboration and late collaboration approaches, mixed collaboration methods have been proposed. Arnold *et al.* [20] suggested exchanging low-level information (early collaboration) in situations where visibility is inadequate and high-level information (late collaboration) in situations where the sensor has high visibility. The authors notice that objects in close proximity to a sensor exhibit a high point density, thereby increasing the likelihood of detection through observations from a single sensor.

An alternative and promising work is V2VNet [21]. Each vehicle in the V2VNet system transforms its acquired point cloud data into a bird's-eye-view (BEV) representation [22]. This BEV data is then processed by a 2D convolutional neural network (CNN) to learn new features and compress the representation to meet bandwidth constraints. The compressed data is then broadcast to other vehicles. Upon receiving compressed messages from neighboring vehicles, each vehicle decompresses these messages using a 2D CNN. The aggregated information from multiple vehicles is fused into a unified BEV representation of the entire scene using a Graph neural network (GNN). This approach enables efficient data sharing and fusion among vehicles, enhancing overall perception and prediction capabilities in complex driving scenarios. However, it is important to note that this method assumes that all vehicles use the same compression and decompression algorithms. This could pose challenges, particularly in establishing communication between diverse vehicular equipment, as the latter may embed different cooperation models.

For this reason, we are interested in learning how to identify and transmit only the pertinent points from the acquired point cloud to the receiver.

2.2. Deep learning for point cloud analysis

Following [23], we define a point cloud as an irregular data structure that represents a subset of points from an Euclidean space, and it has the following main properties:

- Unordered. In contrast to voxel arrays in volumetric grids and pixel arrays in images, a point cloud is a collection of points in any sequence. Put differently, a network that feeds data in order and consumes N $3D$ points must be invariant to the $N!$ permutations of the input set.
- Spatial correlation among points. The points come from a distance-metric space, therefore, nearby points constitute a significant subset, and points cannot be considered isolated. As a result, the model must be able to represent local structures from adjacent points as well as the combinatorial interactions between local structures.
- Invariance under transformations. The learned representation of the point set should be invariant to specific transformations since it is a geometric object. For instance, neither the segmentation of the points nor the global point cloud category should be altered by rotating and translating the points altogether.

Given the just mentioned properties, it is clear that the point cloud data structure shares some features with image data, for instance, the invariance under transformations and the spatial correlation between close regions. Nevertheless, unlike images, point clouds are non-structured data, raising new research challenges. Different approaches have been proposed to deal with such data. In the following, we list and briefly describe the most widespread ones.

Voxel-based methods Voxel-based methods try to solve the challenges related to the disposition of unstructured points by transferring the point cloud to a new well-structured data structure. Thus, they usually bring point clouds to spatially quantized voxel grids. As these methods take inspiration from traditional 2D CNNs, once the point cloud is voxelized, a 3D CNN is applied to the volumetric point cloud representation to leverage the spatial correlation among close regions of the point cloud.

Grid data structures have proven effective. However, to maintain the granularity of the data placement, high voxel resolution is necessary. Processing massive point clouds is expensive because of the cubic growth in computing and memory requirements with voxel resolution. Furthermore, as most point clouds have approximately 90% empty voxels [24], processing no information may waste a large amount of computing resources. The work by Riegler *et al.* [25] introduces the use of octrees to represent non-empty voxels, effectively reducing the processing of noninformative space. This approach offers efficient data structuring but necessitates reducing the voxel resolution to achieve computational speed. This limitation hinders its scalability when dealing with dense point clouds. Since scalability is a paramount concern in cooperative sensing, voxel-based methods appear less suitable.

Pointwise MLP-based methods Standard deep learning methods for 2D images cannot be directly applied to 3D point clouds due to the unstructured form of the latter. PointNet [23] is the first model that

directly takes point clouds as its input without any structural preprocessing. Not relying on an ordered data structure requires that the model is invariant to the input order.

Given an unordered point set $\{x_1, x_2, \ldots, x_n\}$, the core idea behind PointNet is to achieve such invariance by defining a set function $f : \chi \rightarrow \mathbf{R}^k$ that maps a set of points to a k-dimensional vector:

$$f(x_1, x_2, \ldots, x_n) = \lambda \left(\mathcal{A}_{i=1,\ldots,n} \{ \mathcal{M}(x_i) \} \right) \tag{1}$$

where λ and $\mathcal{M}$ are usually multi-layer perceptrons (MLP), and $\mathcal{A}$ is a symmetric function, e.g., MaxPooling, AveragePooling, or sum functions. With this approach, $\mathcal{M}$ is used to extract pointwise features independently, whereas $\mathcal{A}$, since it is a symmetric function, is used to aggregate the features achieving order invariance.

Since PointNet learns features independently for each point, the local structural information between points cannot be captured, leading to poor results on segmentation tasks [26, 27]. Therefore, Qi *et al.* [26] proposed to extend PointNet with a hierarchical network PointNet++ to learn fine geometric structures from the neighborhood of each point. The set abstraction level, which forms the foundation of the PointNet++ hierarchy, is divided into three layers: the sampling layer, the grouping layer, and the PointNet-based learning layer. The sampling layer uses the farthest point scaling (FPS) algorithm to downsample the point cloud. Then, the grouping layer leverages the K nearest neighbor (kNN) search algorithm to partition the point cloud subset into subsets of close points, each representing a local region. Finally, the PointNet layer is applied to each local region to extract the encoding of each neighborhood. PointNet++ learns features from a local geometric structure by stacking several set abstraction levels and abstracts the local features layer by layer. Many networks have been constructed based on PointNet because of its robust representation capability and simplicity. The computation cost in point-based methods grows linearly with the number of input points, making it appealing for cooperative perception purposes. However, Liu *et al.* [28] show that the algorithms used to downsample and perform range queries in three of the most popular point-based models [29–31] take up to 88% of the overall computational time, making them difficult to scale to large point clouds.

Graph-based methods Among graph-based deep learning methods, the use of Graph Convolutional Neural Networks (GCNN) has been proposed in the literature to better exploit the geometric information brought by each point. Simonovsky *et al.* [33] are the pioneers of this approach as they claim point clouds can be seen as a graph $G(V, E)$ with vertices V and edges E. Each point of the point cloud is a vertex of the graph G, and its features usually are its geometric coordinates, laser intensities, or colors associated with the points. Furthermore, a directed edge connects each point to all its neighbors in the geometric space.

Different is the Dynamic Graph CNN (DGCNN) proposed by Wang *et al.* [31]. As the name suggests, in contrast to standard GCNNs, the graph's topological structure is not fixed but rather is dynamically updated after each layer of the network. The main reason is that the neighborhood of each point is computed in the feature space, which changes after each layer, instead of the geometric space, which is static. Even though learning new graph topologies can lead to better feature extraction, since kNN suffers the curse of dimensionality, increasing the feature dimensionality can lead to meaningless neighborhoods. Moreover, the kNN algorithm is computationally expensive, and running this algorithm for each layer can result in an inefficient model.

Graph-based methods prove to be effective in segmentation tasks as they can leverage the spatial information intrinsic to the graph data structure. Moreover, these methods have a low memory footprint with respect to voxel-based models. Nevertheless, as pointwise-based MLP methods, data structuring poses a computational bottleneck for these algorithms.

To tackle the issue of current data structures and algorithms employed in graph-based methods (e.g., neighbor points querying and FPS) taking a long time, Xu *et al.* [10] propose Grid-GCN (GGCN), which combines the memory footprint efficiency of graph-based methods and the data structuring of volumetric-based methods to increase computing efficiency. Tests conducted on the ModelNet classification task [34] reveal that the suggested GGCN network has a computational efficiency that is five times quicker on average than other models. Whereas, on ScanNet segmentation task [35], GGCN achieves a 10× speed-up on average over other models. Moreover, GGCN shows a competitive accuracy, with 93.1% overall voxel labeling accuracy (OA) on ModelNet40 and 85.4% on Scannet.

In this paper, we adopt this architecture, given its structural advantages and its remarkable performance on the point cloud segmentation task. Therefore, we dedicate the following section to a thorough specification of the GGCN model.

2.3. Grid-GCN

Since we employ the GGCN model in our proposed method and experiments, we dedicate this section to a detailed description of the GGCN's architecture. The GGCN model is based on the composition of multiple

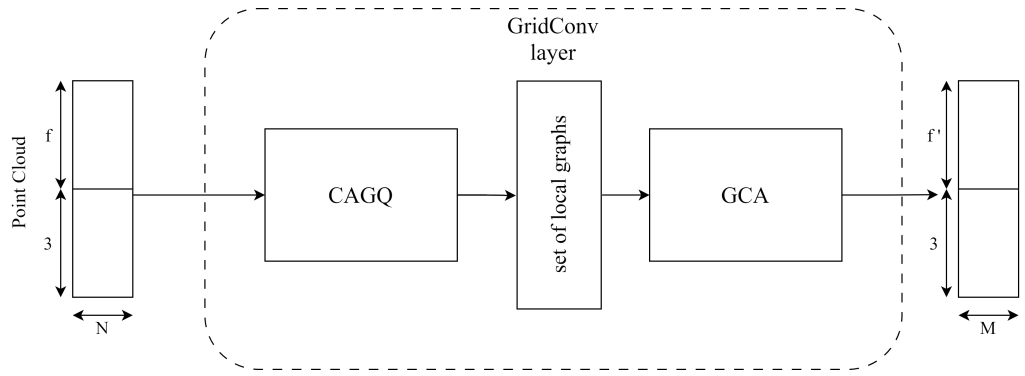


Figure 2: Architecture of GridConv layer. The GridConv layer takes as input a set of N points: each point has $f + 3$ features, where f is the number of semantic features. The layer's output is a set of M points having f' semantic features.

GridConv layers. Each GridConv processes information carried out by N input points and maps them to M points. There are two types of GridConv layers: downsampling GridConv and upsampling GridConv layers. The downsampling GridConv layers can extract important point cloud features by reducing the number of points ($N > M$). By stacking multiple downsampling GridConv layers, it is possible to achieve a compressed representation of the point cloud also called latent representation that represents in a low dimensional space the input point cloud properties useful for the downstream task of interest.

To address the segmentation task, once we have achieved the point cloud's latent representation, we can apply a sequence of upsampling GridConv layers, which recovers the original dimension space of the point cloud ($N < M$). Each GridConv layer is composed of two main submodules: Coverage Aware Grid Query and Grid Context Aggregation. An example of the GridConv layer is illustrated in Fig. 2.

Coverage-Aware Grid Query module Given a point cloud, the Coverage-Aware Grid Query (CAGQ) module's goal is to structure the point cloud effectively and ease the process of querying the neighboring points.

Now, we describe the algorithm the CAGQ module relies on. First, the input space is voxelized by defining a voxel size (v_x, v_y, v_z) and the maximum amount of points n_v a voxel can contain. Then, each point p_i is mapped to the coordinates (x_i, y_i, z_i) to a voxel having voxel index:

$$V_{\text{id}}(x_i, y_i, z_i) = \left(\left\lfloor \frac{x_i}{v_x} \right\rfloor, \left\lfloor \frac{y_i}{v_y} \right\rfloor, \left\lfloor \frac{z_i}{v_z} \right\rfloor \right). \tag{2}$$

Points having the same V_{id} belong to the same voxel, and $\mathbf{V}$ is defined as the set of all the voxels containing at least one input point p_i. Each voxel v_i, with voxel index $V_i = (x_i, y_i, z_i)$, is characterized by its voxel neighborhood $\pi(v_i)$ which can be formally defined as the following set:

$$\pi(v_i) = \{v_j \in \mathbf{V} \mid \|V_i - V_j\| \leq \sqrt{3}\}. \tag{3}$$

Here, the points belonging to $\pi(v_i)$ define the context points of voxel v_i.

Once the point cloud has been voxelized, M non-empty voxels, also called center voxels, are sampled from $\mathbf{V}$ such that they can cover the most occupied space. Finally, for each center voxel v_i, K points n_j, also called node points, are sampled from the context points $\pi(v_i)$ and the group center $g(v_i)$ coordinates of the center voxel are computed as:

$$g(v_i) = \frac{1}{K} \sum_{j=1}^{K} n_j. \tag{4}$$

Grid Context Aggregation module The Grid Context Aggregation (GCA) module's goal is to extract meaningful features from the structured space provided by the CAGQ module.

For each group center $g(v_i)$, the GCA module builds a local graph $G(V, E)$, where V consists of the group center $g(v_i)$ and its K node points provided by CAGQ, and E is the set of edges that connect each node point to the group center. For a given group center, GCA computes its features $\tilde{f}_c$ from its node points p_i as:

$$\tilde{f}_{c,i} = e(\chi_i, f_i) * \mathcal{M}(f_i) \tag{5a}$$

$$\tilde{f}_c = \mathcal{A}(\{\tilde{f}_{c,i}\}, i \in 1 \dots K) \tag{5b}$$

where f_i and χ_i are respectively the features and (x, y, z) coordinates location of the node point p_i, while $\tilde{f}_{c,i}$ is the contribution of node point p_i to its center group. $\mathcal{M}$ is a multi-layer perceptron, e is an edge attention function, and $\mathcal{A}$ is an aggregation function. The edge attention function designed by the GGCN's authors considers both geometrical and semantic relations between the group center and the node point p_i. Furthermore, the authors believe that the underlying contribution of each node point from the previous layers should also be taken into account. As a matter of fact, they claim that points carrying more information from previous layers should be more critical and, thus, they should be given more attention. As a result, they introduce the notion of *coverage weight*, which is defined as the count of points aggregated to a node in previous layers.

As mentioned before, the attention function is modeled to consider the semantic relation between the center group and the node point, which requires both node point features f_i and group center features f_c. Nevertheless, since the group center is just a virtual point that does not necessarily belong to the set of N input points, no semantic features are available. To solve this problem, let $g(v_i)$ be the group center of the voxel v_i, the semantic feature f_c can be replaced by the context semantic feature f_{cxt}:

$$f_{cxt} = \mathcal{A}_{cxt}(\{f_{p_j}\}, p_j \in v_k \mid v_k \in \pi(v_i)), \tag{6}$$

where $\mathcal{A}_{cxt}$ is any aggregation function. In other words, the context feature of $g(v_i)$ is computed as the aggregation of the voxel v_i's context points features.
Finally, the edge attention function can be summarized with the following:

$$e = \mathcal{M}_{att}(\mathcal{M}_{geo}(\chi_c, \chi_i, w_i), \mathcal{M}_{sem}(f_{cxt}, \mathcal{M}(f_i))), \tag{7}$$

where χ_c is the geometrical location of the group center computed with the Equation 4, w_i is the coverage weight of point p_i and f_{cxt} is the contextual feature of the group center.

Sampling strategies Among the main contributions provided by the GGCN's authors are the algorithms used to sample the M center voxels from $\mathbf{V}$ and the k nodes from the context points in $\pi(v_i)$. The first algorithm includes two complementary methods:

1. Random Voxel Sampling (RVS): each occupied voxel has the same probability of being drawn. The authors demonstrate that the center voxels are more uniformly distributed than the centers sampled from the input points by the Random Point Sampling (RPS) method; as a matter of fact, the RVS is more resilient to point density imbalance.
2. Coverage-Aware Sampling (CAS): each drawn center voxel can cover up to λ non-empty voxel neighbors. The goal of CAS is to find a set of M center voxels $\mathbf{V}_c \in \mathbf{V}$ such that their neighborhood can maximize the covered space. To optimally solve this problem, an exhaustive enumeration of all the possible combinations is required, which is computationally impractical. Thus, a greedy algorithm is used to approximate the optimal solution. First, RVS is employed to select M voxels from $\mathbf{V}$, also called *incumbent* voxels. Then, CAS iteratively selects one point at a time, also called the *challenger*, from the unpicked ones to challenge a random incumbent. If replacing the incumbent with the challenger improves the coverage, the incumbent is discarder, and the challenger becomes a new incumbent. For a challenger v_c and an incumbent v_i, the coverage improvement is computed with the following heuristics:

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0. \\ 0 & \text{otherwise.} \end{cases} \tag{8a}$$

$$H_{\text{add}} = \sum_{v \in \pi(v_c)} \delta(C_v) - \beta \cdot \frac{C_v}{\lambda_{v_c}} \tag{8b}$$

$$H_{\text{rmv}} = \sum_{v \in \pi(v_i)} \delta(C_v - 1) \tag{8c}$$

where λ_v is the number of neighbors of voxel v and C_v is the number of incumbents covering voxel v. As a result, H_{add} represents the coverage improvement by adding v_c to the incumbents, whereas H_{rmv} is the coverage aggravation by removing v_i. If $H_{add} > H_{rmv}$, the challenger v_c replaces the incumbent v_i.

The authors claim CAS is more effective and efficient than RPS and FPS, which are the main sampling algorithms used by other models. According to the experiments, on average, CAS covers 75.2% of the occupied space, whereas RPS and FPS respectively occupy 45.6% and 65%. FPS is shown to be competitive in terms of coverage; nevertheless, its time complexity is $O(N \log(N))$, in contrast, CAS is linear in time.

For what concerns the node points querying, GGCN uses a modified version of kNN. GGCN relies on the voxelized structure to find the k nearest neighbors. Assuming the vanilla k-NN relies on a KD-tree data structure, its time complexity to find all the k neighbors for a given point is $O(k \log(N)$. In contrast, the proposed kNN restricts the search of the neighbors only to the voxel neighborhood $\pi(v_i)$, dramatically reducing the search space, and achieving a computational time of $O(k \log(V))$, where V is the number of points of the neighborhood.

3. System Model

We consider a vehicular urban setting covered and served by a single Road Side Unit (RSU), i.e., a communication node deployed along a road or on the roadside providing connectivity with the infrastructure to the vehicles crossing the scenario.

Let $\mathcal{V}_t = \{1, \cdots, V_t\}$ be the set of vehicles served by the RSU at time step t. Each vehicle $v \in \mathcal{V}_t$ is assumed to be equipped with a communication transceiver to enable vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. In Section 3.1, we provide a description of the communication system model, while in Section 3.2, we describe the LiDAR sensor model.

3.1. Communication system model

In our setting, V2I communication between the vehicles crossing the urban scenario and the RSU is used to provide two kinds of information from the vehicle infrastructure:

1. inter-vehicle association, i.e., which vehicles are matched for communication among the available ones at a given time step;
2. communication resources selection, i.e., which transmission power and bandwidth resources have to be used by the vehicles during communication.

For the considered V2I setting, communication between a vehicle and the RSU is performed with frequency $1/\tau_{RSU}$. Therefore, association and resources selection information is received by each vehicle every τ_{RSU}.

We assume that the RSU has sufficient computational and communication resources to achieve the centralized tasks required for vehicles association and resource selection, while in the following we focus on V2V sensory data exchange, where the contributions proposed in this work are centered.

For inter-vehicle sensory data exchange, communication is performed with frequency $1/\tau_v$, with $\tau_v = \tau_{RSU}/L$, for a positive integer L. Therefore, the vehicle association and resource selection procedures are here assumed to be performed every L inter-vehicle communication slots, over which the same vehicles' pairs and the same communication resources are selected—which is a common assumption in the literature, e.g., [18]. We consider a time-slotted communication with temporal slots of duration $\Delta t < \tau_v$ to allow sensory data processing at the receiving vehicle.

Vehicle association At a given time step $t = i \cdot \tau_{RSU}$, $i \in \mathbb{N}$, the association between vehicle $v \in \mathcal{N}_t$ and vehicle $v' \in \mathcal{V}_t$ for communication is represented by the Boolean variable $a_{vv'}$, which is 1 the communication link between v and v' is used during time slot t, and 0 otherwise. We assume that, at each time slot, a vehicle can communicate only with up to a single vehicle, i.e., for all $v \in \mathcal{V}_t$,

$$\sum_{\substack{v' \in \mathcal{V}_t \\ v' \neq v}} a_{vv'} \leq 1. \tag{9}$$

Resources selection Each vehicle v is assumed to have a maximum bandwidth $B_{v,\max}$ and a transmit power P_v for communication, which are assumed to be the same for all vehicles. The communication bandwidth is partitioned in resource blocks (RBs) of size ω. The set of RBs is denoted as $\mathcal{K}$, and its cardinality is $B_{\max}/\omega$, where $B_{\max}$ is considered to be a multiple of the RB bandwidth ω. The selection of the bandwidth resources to be used by a vehicle v is managed through the Boolean variable $\nu^k_{vv'}$, which is 1 if RB $k \in \mathcal{K}$ is used for communication between vehicle v and vehicle v', and 0 otherwise.

Data points exchange Following the modeling in [18], the number of points that can be effectively sent in the communication between the vehicles v and v' at time slot t is given by:

$$R_{vv'}(t) = \frac{\tau_v}{D} \sum_{k \in \mathcal{K}} \nu^k_{vv'}\, \omega\, \log_2(1 + \gamma^k_{vv'}), \tag{10}$$

where $\gamma^k_{v'}$ is the Signal to Interference plus Noise Ratio (SINR) at the v' equipment, and D is the total number of bits needed to represent the data to be sent. In the following, we consider the transmission of a set of N

data points, associated to F q-bit features each. Therefore, the total number of data points that can be sent over the communication link is given by:

$$N = \left\lfloor \frac{D}{F\,q} \right\rfloor. \tag{11}$$

The communication SINR $\gamma^k_{vv'}$ is defined as:

$$\gamma^k_{vv'} = \frac{P^k_{R,vv'}}{N_0\omega + I^k_{v'}(t)}, \tag{12}$$

with $P^k_{R,vv'}$ the received power at vehicle v' from vehicle v, and $I^k_{vv'}(t)$ the communication interference at vehicle v'. The received power $P^k_{R,vv'}$ is provided by:

$$P^k_{R,vv'} = \frac{P_v}{|\mathcal{K}_v|}|h^k_{vv'}|^2, \tag{13}$$

where $h^k_{vv'}$ is the complex-valued communication channel gain that models mixed Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS) electromagnetic propagation between the transmitter vehicle v and the receiver v'.

We assume that inter-vehicle communication happens at millimeter wave (mmWave) frequencies, where both the transmitter and the receiver can employ directive communication beams, perfectly pointed to each other. In this case, we can safely neglect the interference term $I^k_{v'}(t)$ in (12) for the computation of the SINR at the receiver.

3.2. LiDAR sensor model

In addition to the communication equipment, each vehicle is endowed with an ideal rotating LiDAR sensor to perceive the environment and make informed decisions. This sensor is characterized by a horizontal field of view (FOV) θ_h, a vertical FOV θ_v, a rotation frequency f_{lidar}, and a resolution of ψ planes.

During each rotation period $\delta = 1/f$, the LiDAR mounted on vehicle v captures a point cloud defined as a set of points $\mathcal{P}_v = \{p_1, p2, \ldots, p_n\}$.

Each point $p_j \in \mathcal{P}_v$ is associated with a tuple:

$$p_j = (x, y, z, \iota), \tag{14}$$

where x, y, and z are the x, y, and z Cartesian coordinates of the detected point with respect to the coordinate system O_v, i.e., the Cartesian coordinate system centered at the LiDAR position, integral with to the vehicle v, so that the positive X axis aligns with the vehicle's heading, the positive Y axis is 90 degrees anticlockwise relative to the positive X axis, and the positive Z axis extends out of the XY plane composing a right-handed triplet.

On the other hand, ι defines the laser intensity for the hit point and is computed as follows:

$$\iota = e^{-a\cdot d}, \tag{15}$$

where a is a constant attenuation coefficient depending on atmospheric conditions, and d is the distance of the hit point from the LiDAR. Consequently, the intensity serves as a proxy for distance and does not consider the material properties of the point. Therefore, the ι value does not provide additional information with respect to the point position and is not considered as a feature in our experiments.

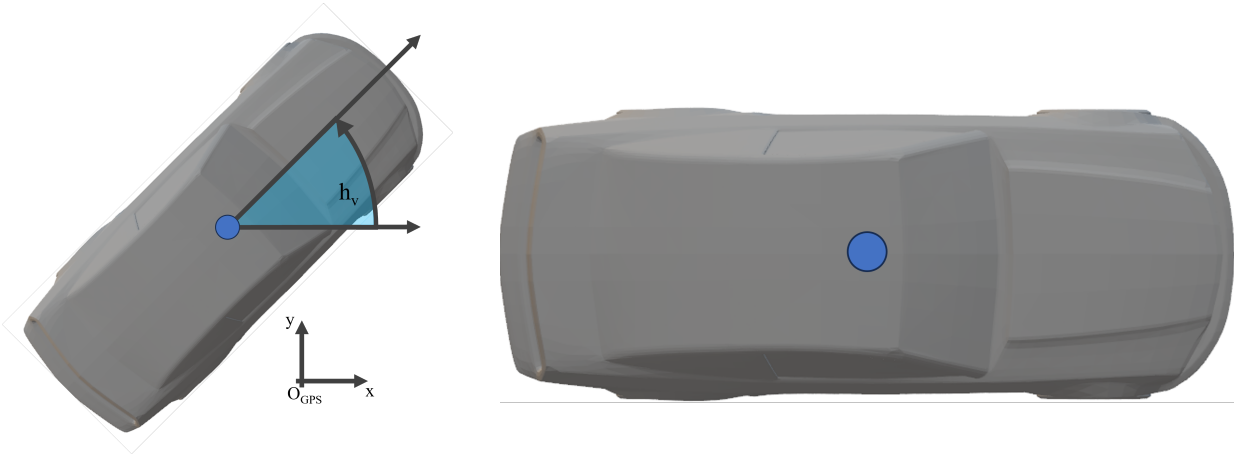
4. The cooperative point selection problem formulation

In this section, we formulate the cooperative point selection problem, which is the focus of this paper and for which we propose our solution in Section 5.

Let $\mathcal{V}_t$ be the set of vehicles v served by the RSU at time step t with the properties defined in Section 3. Assuming that the ground is plane, at time t, each vehicle $v \in \mathcal{V}_t$ is positioned at:

$$l_v(t) = (x_v, y_v), \tag{16}$$

where x_v and y_v represent the longitude and latitude of the vehicle's barycenter. Given the reduced spatial extent of the urban scenario, x_v and y_v are approximated to the X and Y coordinates on a Euclidean plane with a chosen local Cartesian coordinate system O_{gps} where the x-axis points towards East and the y-axis towards North.



(a) Vehicle orientation. The vehicle orientation is defined by the angle h_v it forms with the O_{GPS} x-axis.

(b) LiDAR placement on a sedan. The sensor is installed on the rooftop, in a position with high visibility in the front area of the vehicle. In this figure, the LiDAR position is highlighted with a blue circle.

Figure 3: Vehicle orientation and LiDAR placement on a sedan.

Each vehicle v moves at a speed of s_v in a direction defined by angle $h_v \in [0, 2\pi]$ as in Fig. 3a.

We assume that each vehicle $v \in \mathcal{V}_t$ is equipped with a LiDAR sensor. All LiDARs are assumed to be of the same type and to have the same hardware equipment. As illustrated in Fig. 3b, the sensor is installed on top of the roof at height z_v with respect to the street level. Moreover, each vehicle is equipped with a communication transceiver to support inter-vehicle communication. As for the sensors, we assume all vehicles share the same technical specifications for the installed antennas. A thorough description of this communication system and of the LiDAR sensor model is provided in Section 3.

4.1. Problem formulation

In the considered vehicular cooperative perception scenario, each vehicle v_{rx} is interested in pairing with another vehicle v_s to extend its environment perception by receiving information from v_s. The easiest approach would be to make v_s send its own point cloud $\mathcal{P}_s$ to v_{rx}. However, the maximum amount of points that can be sent is constrained by the available communication resources summarized by eq. (10).

Assuming that vehicle v_{rx} at timestep t has a position $l(t)$, speed s and heading h, we aim to determine if, given this information, v_s can learn which are the points $\mathcal{P}_{max} = \{p_1, p_2, \ldots, p_m\} \subset \mathcal{P}_s$ that maximize the satisfaction of v_{rx}.

Given a point $p \subset \mathcal{P}_{max}$, we formulate the *satisfaction* of v_{rx} as:

$$\mathcal{S}(p) = \mathcal{R}_{v_{rx}}(p) \cdot \alpha(p) \cdot \eta(p), \tag{17}$$

where $\mathcal{R}_{v_{rx}} : \mathbb{R}^5 \to [0, 1]$ is the function measuring the spatial interest on point p manifested by vehicle v_{rx} moving with speed s and heading h, while $\alpha : \mathbb{R} \to [0, 1]$ is a decreasing function of elapsed time from the acquisition of point p, representing the temporal interest on p. Finally, $\eta(p) : \mathbb{R}^3 \to [0, 1]$ is the novelty score associated with p, measuring how much information p adds to the v_{rx}'s point cloud $\mathcal{P}_{rx}$.

Assuming that the v_{rx} position, heading, and speed are known to vehicle v_s and that all vehicles use the same functions $\mathcal{R}$ and α, v_s can easily compute both $\mathcal{R}_{v_{rx}}(p)$ and $\alpha(p)$. Hence, the problem reduces to learning the function $\eta(p)$. The design of $\mathcal{R}_{v_{rx}}$, α and η functions is defined in Section 5.1.

5. Proposed method

Cooperative perception introduces several challenges. When a vehicle v_s transmits information to vehicle v_{rx}, it is crucial to determine the most relevant information for the receiver. This ensures that bandwidth is not wasted on transmitting non-relevant or redundant data. In this section, we describe the proposed method for cooperative LiDAR sensing in V2V communication scenarios.

We define which points we consider important in Section 5.1. Furthermore, when vehicles communicate for consecutive steps, it is crucial to avoid transmitting the same data sent in the previous steps. We address this issue in Section 5.2.

In Section 5.3, we extend the Grid-GCN architecture [10] to address the cooperative LiDAR points selection problem introduced in Section 4.

Having a functioning algorithm that recognizes the most important points for transmission is futile if we cannot associate vehicles that share points of interest or are obstructed from communication due to occlusions. To address this, we discuss how to effectively associate communicating vehicles in Section 5.4.

5.1. Point importance metrics

In this section, we formalize how to quantify a vehicle's interest in a point p belonging to a point cloud $\mathcal{P}$. This interest can be divided into three distinct types: spatial, temporal, and novelty. The following paragraphs provide a detailed description of each.

Spatial point interest In the domain of autonomous driving, the Region of Interest (RoI) refers to a specific area within the field of view of sensors where the system concentrates its attention on critical decision-making tasks like object detection, lane tracking, and obstacle avoidance. Hence, given a point cloud $\mathcal{P}$, the vehicle shows varying degrees of interest for each point $p \in \mathcal{P}$ based on its position relative to the vehicle.

Given a point p and its coordinates (x, y, z), we model the RoI score function $\mathcal{R}_v(p)$ that measures the interest of a vehicle v for point p as:

$$\mathcal{R}_v(p) = r_\omega(p) \cdot r_d(p) \cdot r_z(p) \cdot r_n(p), \tag{18}$$

where $r_\omega(p)$ measures the interest along a given direction represented by an angle $\omega \in [0, 2\pi]$ on the XY plane, and $r_d(p)$ measures the interest with respect to the distance between the vehicle and the point. Complementarily, $r_z(p)$ computes the interest of a point with respect to its height, i.e., along the Z axis. Regarding $r_d(p)$, we assume that the vehicle has its main interest along its heading direction, and the interest slowly decays as the direction departs from the vehicle's heading. For this reason, we model $r_\omega(p)$ with the scaled Von Mises probability density function:

$$r_\omega(p) = \frac{1}{2\pi I_0(\kappa) R_\omega} \exp\left(\kappa \cos(\omega)\right), \tag{19}$$

where ω is the angle of the point p with respect to the X axis in the XY plane, $I_0(\kappa)$ is the modified Bessel function of the first kind of order 0, and R_θ is the maximum value of r_ω, so that the function can distribute values in the range $[0, 1]$.

Regarding $r_d(p)$, we assume that the vehicle is more interested in points close to it. Nevertheless, it is realistic to assume that the RoI's range increases with the vehicle's speed s. For this purpose we can model $r_d(p)$ with a smooth exponential function:

$$r_d(p) = \lambda_d e^{-\lambda_d d} \tag{20a}$$

$$d = \sqrt{x^2 + y^2 + z^2} \tag{20b}$$

$$\lambda_d = \frac{\gamma}{s}, \tag{20c}$$

where $\gamma > 0$ is a hyperparameter that controls the dependency of $r_d(p)$ on s, and $\lambda_d > 0$ is a hyperpeparameter that regulates the function smoothness.

Finally, we model $r_z(p)$ so that the vehicle focuses more on the street level. Thus, the interest decays as the height of the points increases:

$$r_z(p) = \lambda_z e^{-\lambda_z (z + z_0)}, \tag{21}$$

where $\lambda_z > 0$ is a hyperparameter and z_0 is the height of the lidar. A 2D representation of an example of RoI function is shown in Fig. 4.

Temporal point interest In an urban environment, the presence of static or low-dynamic objects, such as pedestrians, can be occluded by highly dynamic objects like vehicles. In such scenarios, the LiDAR's high acquisition frequency might result in the detection of a low-dynamic object in one time step, only to have it occluded by a highly dynamic object in the next time step, causing the vehicle to overlook the initial object. This issue can be mitigated by preserving all recent points instead of discarding them and tracking the time elapsed from the acquisition of each point.

Given a point p acquired at time t_0, we define its Age of Information (AoI) at time t as follows:

$$\alpha(p) = e^{-\lambda_{AoI}(t - t_0)}. \tag{22}$$

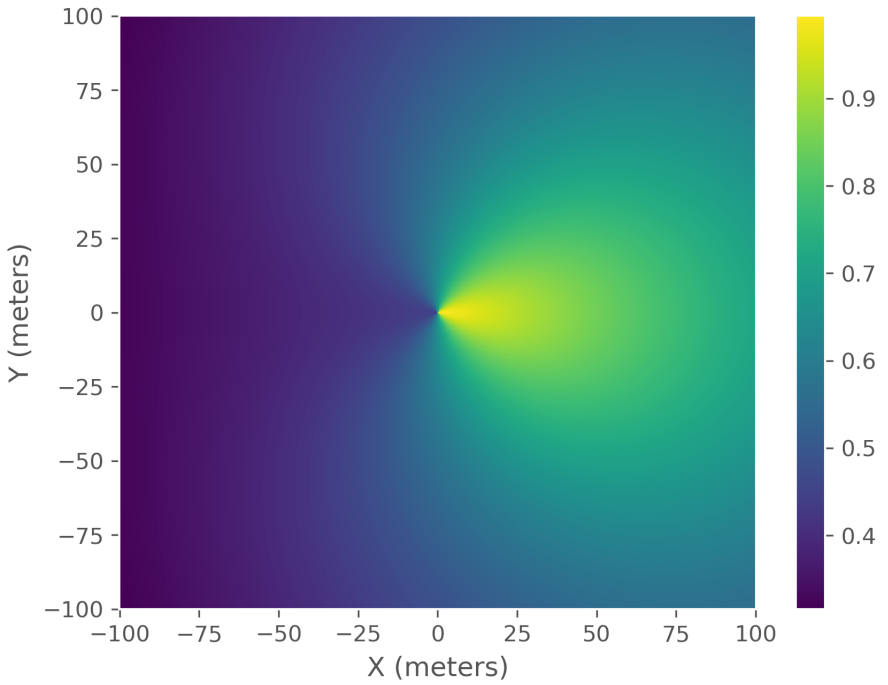


Figure 4: Heatmap of $\mathcal{R}_{v_i}$ for a vehicle cruising with a speed of $3.6m/s$, computed on plane $z = 0$.

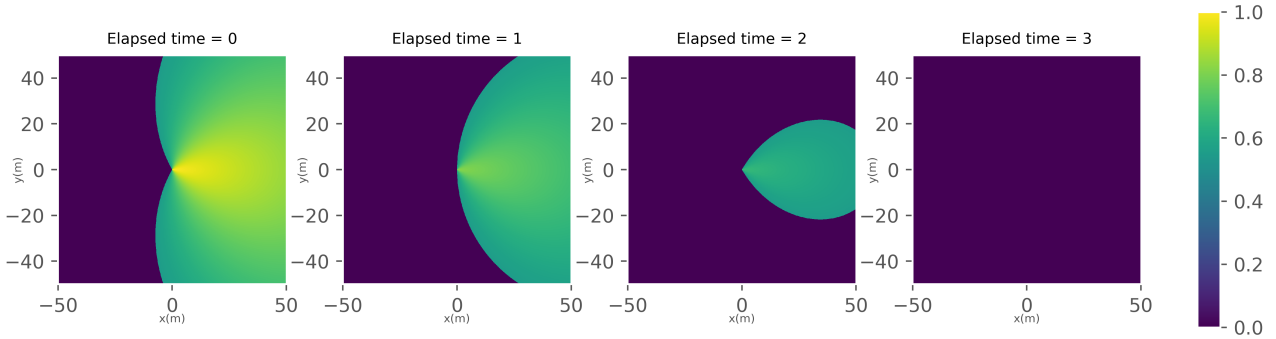


Figure 5: Heatmap of $\mathcal{R}_{v_i} \cdot \alpha$ for a vehicle cruising at a speed of $3.6\frac{m}{s}$ on the plane $z = 0$. The highlighted region of interest has a score $\mathcal{R}_{v_i} \cdot \alpha > 0.55$.

Here, $\lambda_{AoI} > 0$ is a decay factor and serves as a hyperparameter to control the decay of the AoI. As a result, the higher the AoI, the more recent—and so, also more reliable—is the acquired point. Instead of discarding points of past acquisitions, we can use the AoI to define the temporal interest in old data. As the Age AoI for a point p decreases, the vehicle's interest in that point also reduces. Fig. 5 illustrates this concept.

Novelty point interest We assume that the points in the set $\mathcal{P}_{rx}$ are realizations of an unknown probability distribution $\mathcal{U}$, and that dense regions in $\mathcal{P}_{rx}$ contains noteworthy information. In contrast, low-density regions do not provide enough information to the vehicle v_{rx}. We can model the amount of information a new point p adds to the knowledge of v_{rx} as:

$$\eta(p) = \begin{cases} 1 - f_{\mathcal{U}}(p) \cdot \frac{1}{c} & \text{if } 0 \leq f_{\mathcal{U}}(p) \leq 1. \\ 1 & \text{if } f_{\mathcal{U}}(p) \geq 1. \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

where $f_{\mathcal{U}}$ is the probability distribution function for the distribution $\mathcal{U}$ and c is a normalization constant to distribute the $\eta(p)$ values in the interval $[0, 1]$. Since c is an empirical value, there are no guarantees that $f_{\mathcal{U}}(p) < c$. Thus, we need to explicitly limit $\eta(p)$ to the interval $[0, 1]$. Since the distribution $\mathcal{U}$ is unknown—and so is the probability distribution function—to approximate $f_{\mathcal{U}}$, we rely on a non-parametric estimator [36] based on the k-nearest neighbors (kNN) algorithm defined as follows:

Parameter	Value	Parameter	Value
κ	0.4	λ_z	0.03
γ	0.01	c	0.0006607
k	8	β	0.5
λ_{AoI}	0.2	λ_{AoT}	0.2
$\bar{c}$	4.189		

Table 1: Hyperparameters for the RoI, AoI, AoT and η.

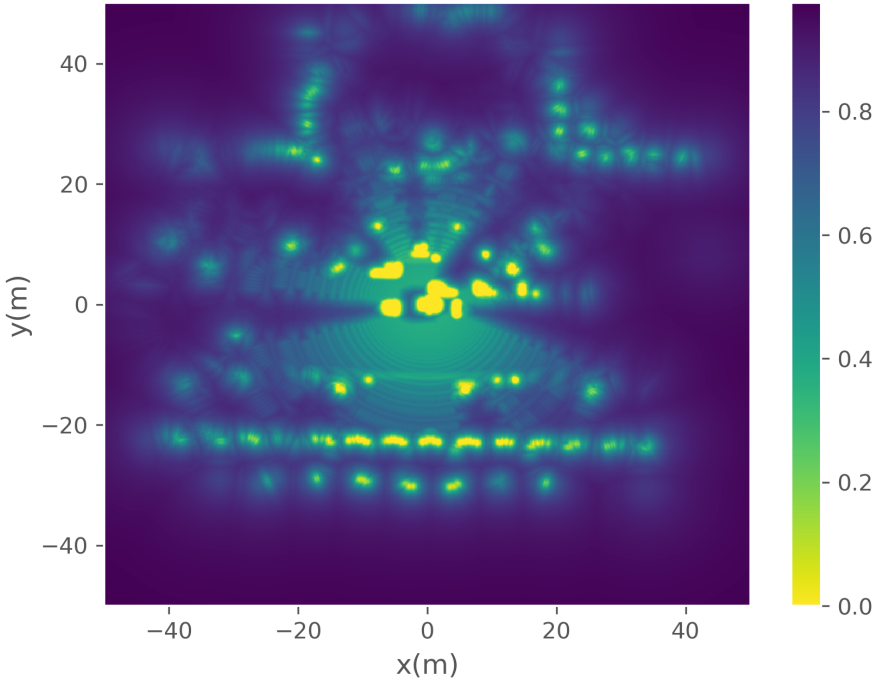


Figure 6: Heatmap for the $\eta(p)$ function computed on plane with z=0.5m. Here, z=0.5m means 0.5 meters above the level of the street. Brighter colors of the heatmap indicate a high density of points and, consequently, a low value for η. In contrast, darker colors indicate a lack of information, hence, a high value of η.

$$\hat{f}_{\mathcal{U}}(p) = \frac{k}{m\bar{c}\rho_k(p)}, \tag{24}$$

where k is the kNN's hyperparameter, m is the number of points in $\mathcal{P}_{rx}$, $\bar{c}$ is the volume of a d-dimensional unit ball, and $\rho_k(p)$ is the distance between the point p and its k-nearest neighbor in $\mathcal{P}_{rx}$. The efficacy of the estimator $\hat{f}_{\mathcal{U}}(p)$ used to compute $\eta(p)$ is illustrated in Fig. 6.

5.2. Age of transmission

As mentioned in Section 3, we assume that two associated vehicles can exchange sensory data, with a frequency $1/\tau_v$, where the duration τ_v can be very short (≈ 0.05 s). Hence, the environment does not change abruptly between two consecutive communication slots. In this situation, we want to avoid that the sender vehicle v_s sends to v_{rx} the same data sent in the previous time steps. A naive approach is to take track of the sent data. By doing so, the model no longer considers the already sent data as input. Nevertheless, we are not considering the spatial correlation between sent and not sent points. Considering the example provided in Fig. 7, the vehicle v_s might send at time step t_0 several points $\mathcal{P}_r$ belonging to a specific object O, represented by the red points. At the next timestep t_1, the vehicle v_s can still have other points $\mathcal{P}_y$ relative to object O, which were not sent the time step before illustrated as yellow points. If the points $\mathcal{P}_r$ at time step t_0 are dense enough for the vehicle

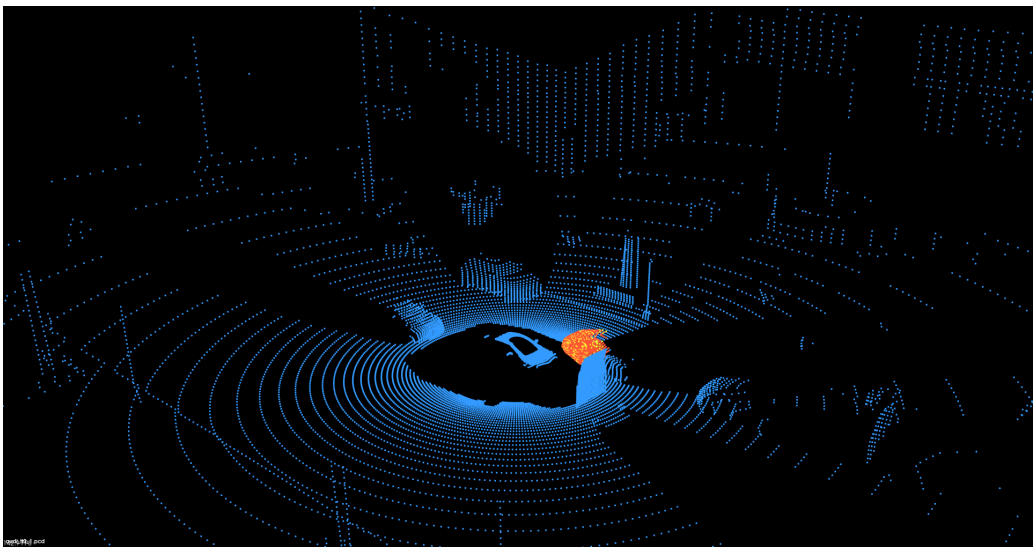


Figure 7: Example supporting the need for AoT. Red points represent sent points semantically belonging to object O, while yellow points semantically belong to O but have not been sent yet. Blue points are non sent points that do not semantically belong to object O.

v_{rx}, points $\mathcal{P}_y$ are redundant and should not be sent. Instead, we believe the vehicle v_s should be incentivized to send points that do not semantically belong to the object O (blue points), increasing the overall perception of the environment. This is also supported by the fact that we expect the vehicle v_{rx} to return a low novelty score $\eta(p)$ for the points belonging to $\mathcal{P}_y$.

For this purpose, we introduce the *Age of Transmission* (AoT) feature, which is assigned to each point of the vehicle's point cloud. The AoT of a point p is a proxy for the time elapsed from the last time the point p was sent to v_{rx}. A vehicle assigns the AoT to the point p acquired or received at time t_0 with the following:

$$\alpha_{AoT}(p,t) = \begin{cases} 0 & \text{if } t = t_0. \\ 1 & \text{if } p \text{ is sent to } v_{rx} \text{ at time } t. \\ e^{-\lambda_{AoT}(t-t_0)} & \text{otherwise.} \end{cases} \tag{25}$$

5.3. Cooperative points selection via Grid-GCN

In this section, we propose to use a graph neural network (GNN) to address the cooperative point selection problem formulated in Section 4. GNNs are architectures that can capture spatial relations between points belonging to a set, keeping the permutation invariance of the input. Among the different GNN architectures, we propose to use the Grid-GCN (GCNN) architecture due to its computational efficiency, which is an essential aspect when dealing with situations in which high throughput is needed. For further details about the GGCN architecture, the reader can refer to Section 2.3.

From Section 4 emerges the need to use contextual information to extract meaningful insight from the input. In GNNs, it is common to incorporate such information in a global node [38]. Nevertheless, GGCN does not provide such functionality. For this reason, we adapt the GGCN implementation to our needs.

In a standard GNN framework, a directed graph is defined as a tuple $G = (\mathbf{u}, V, E)$. The $\mathbf{u}$ is a vector representing global attributes, which can embed contextual information about the graph. The $V = \{\mathbf{v}_i\}_{i=1:|V|}$ is the set of nodes, where each $\mathbf{v}_i$ is a vector representing the node's features. Finally, $E = \{\mathbf{e}_{i,j}\}$ is the set of edges, where each $\mathbf{e}_{i,j}$ is a vector representing the attributes of the edge connecting the sender node $\mathbf{v}_i$ to the receiver node $\mathbf{v}_j$.

Provided a graph G as input, a traditional GNN layer executes the following steps:

1. It updates the value of each edge $\mathbf{e}_{i,j}$ to:

$$\mathbf{e}'_{i,j} = \phi^e(\mathbf{e}_{i,j}, \mathbf{u}), \tag{26}$$

   where ϕ^e is a nonlinear function. The set of updated edges going towards a node $\mathbf{v}_i$ is denoted as E'_i, whereas the set of all updated edges is E'.
2. It applies to each set E'_i the symmetric aggregation function $\rho^{e\to v}$ to extract the aggregated value $\bar{\mathbf{e}}_i$ of the edge updates, which is used for the node update.

3. It computes for each node $\mathbf{v}_i$ the updated node value as:

$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}), \tag{27}$$

   where ϕ^v is a nonlinear function. The set of updated nodes is denoted as V'.
4. It extracts the aggregated value $\bar{\mathbf{v}}'$ of the nodes in V' with the symmetric aggregation function $\rho^{v\to u}$.
5. It extracts the aggregated value $\bar{\mathbf{e}}'$ of the edges in E' with the symmetric aggregation function $\rho^{e\to u}$.
6. Finally, the global node $\mathbf{u}$ is updated to:

$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}), \tag{28}$$

   where ϕ^u is non linear function.

We adapt the GGCN architecture to this framework by introducing the global node. In a GridConv layer, the output graph is composed of the barycenters of the node points belonging to the input graph. This means the nodes in the output do not belong to the input graph, and the input nodes do not belong to the output graph. For this reason, there is no update rule applied to the input nodes but only to the barycenters, which are virtual points and do not have real features. From eq. (5b), we can notice that the node update rule coincides with the edge aggregation function $\rho^{e\to v}$ defined at point 2. As a result, we can eliminate the ϕ^v function without loss of expression. To adapt the edge update attention function to the global node framework, we can modify the eq. (7) into:

$$e = \mathcal{M}(\mathcal{M}_{geo}(\chi_c, \chi_i, w_i), \mathcal{M}_{sem}(f_{cxt}, f_i), \mathbf{u}). \tag{29}$$

Furthermore, given that:

$$\mathbf{v}'_i = \rho^{e\to v}(E'_i) = \mathbf{e}'_i, \tag{30}$$

we can rewrite:

$$\begin{aligned}
\bar{\mathbf{v}}' &= \rho^{v\to u}(V') && (31)\\
&= \mathcal{A}(\{\mathbf{v}' \mid \mathbf{v}' \in V'\}) && (32)\\
&= \mathcal{A}(\{\bar{\mathbf{e}}'_i \mid \bar{\mathbf{e}}'_i \,\forall i = 1 \cdots \mid V' \mid\}) && (33)\\
&= \rho^{e\to u}(E') && (34)\\
&= \bar{\mathbf{e}}'. && (35)
\end{aligned}$$

Thus, assuming to use the same aggregation function, $\bar{\mathbf{v}}'$ and $\bar{\mathbf{e}}'$ encode the same information in the GGCN architecture. As a final result, we can rewrite the global node with the following update function:

$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \mathbf{u}). \tag{36}$$

We model the cooperative point selection problem as a binary segmentation problem. We aim to learn a function $\mathcal{M} : \mathbb{R}^{m\times h} \cup \mathbb{R}^4 \to \{0,1\}^m$, where m is the number of input points and h is the number of features per point. The function $\mathcal{M}$ should approximate the function u representing which points are improvements for the receiving vehicle v_{rx}:

$$\forall p \in \mathcal{P}_s, \quad u(p, \mathcal{G}) = \begin{cases} 1 & \text{if } \eta(p) > \beta, \\ 0 & \text{otherwise,} \end{cases} \tag{37}$$

where $\mathcal{G}$ is a vector containing contextual information on the position and heading of vehicle v_{rx}, while β is a tunable threshold. To learn such a function, we aim to minimize the binary cross-entropy loss:

$$L = -\frac{1}{\mid \mathcal{P}_s \mid} \sum_{p\in\mathcal{P}_s} u(p,\mathcal{G}) \cdot \log(\mathcal{M}(p,\mathcal{G})) + (1 - u(p,\mathcal{G})) \cdot \log(1 - \mathcal{M}(p,\mathcal{G})). \tag{38}$$

To capture the relationship between the point cloud and the geometric information of the receiver and the sender, we incorporate the sender's position in the input. This is implicitly achieved since the coordinates of the points are relative to the reference system with the origin in the sender's LiDAR position, as discussed in Section 3.2. Still, the position l^g_{rx} and the heading h^g_{rx} of the receiver are with respect to the global reference system. Therefore, we bring such information to the sender's local reference system. Let l^g_s and h^g_s be, respectively, the sender's position and heading with respect to O_{gps}, then:

$$l^l_{rx} = l^g_{rx} - l^g_s, \tag{39a}$$
$$h^l_{rx} = h^g_{rx} - l^g_s. \tag{39b}$$

Moreover, once the points $\mathcal{P}_{max}$ have been selected, we must rototranslate them to get the points $\mathcal{P}^{rx}_{max}$ in the receiver reference system $O_{v_{rx}}$. Then:

$$R_{rx} = \begin{bmatrix} \cos(h^g_{rx}) & -\sin(h^g_{rx}) & 0 \\ \sin(h^g_{rx}) & \cos(h^g_{rx}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{40a}$$
$$R_s = \begin{bmatrix} \cos(h^g_s) & -\sin(h^g_s) & 0 \\ \sin(h^g_s) & \cos(h^g_s) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{40b}$$
$$\mathcal{P}^g_{max} = \mathcal{P}_{max} \times R^T_s + l_s \tag{40c}$$
$$\mathcal{P}^{rx}_{max} = (\mathcal{P}^g_{max} - l_{rx}) \times R_{rx}. \tag{40d}$$

Assuming that the learned model $\mathcal{M}$ captures the spatial relation between the input points, we also aim the function $\mathcal{M}$ to learn that points having low AoT close to points with high AoT should not be sent because the receiver already has enough information about that specific region.

5.4. Centralized vehicle association

For retrieving sensory data, we assume that each vehicle can form an association with a maximum of one other vehicle over a period τ_{RSU}, as defined in Section 3.1. Furthermore, we assume that communications are unidirectional. Consequently, in the scenario where vehicles v and v' are paired for communication, only one ot the two is transmitting information, while the other is solely receiving information. Since we focus on determining what the vehicles we opt for a heuristic method to pair the vehicles.

We assume that the RSU, responsible for vehicle pairing, possesses comprehensive knowledge of the urban scenario, i.e., the RSU can ascertain the position and heading of each vehicle, identify all available communication pairs, and determine the associated estimated received power for the communication link.

We model vehicle association as a matching problem on a directed, weighted non-bipartite graph $G(V, E, W)$. The nodes $V = \{1 \cdots \mid \mathcal{V} \mid\}$ represent vehicles served by the RSU. The edges $E = \{(i, j) \mid i, j \in V\}$ denote communication links between vehicles, and the weights $W = \{w_{i,j} \mid \exists (i, j) \in E\}$ signify the estimated received power for each communication link. The RSU's objective is to find a match that maximizes the overall estimated received power:

$$\text{Maximize} \sum_{(i,j)\in E} w_{ij} x_{ij} \tag{41}$$

Subject to:

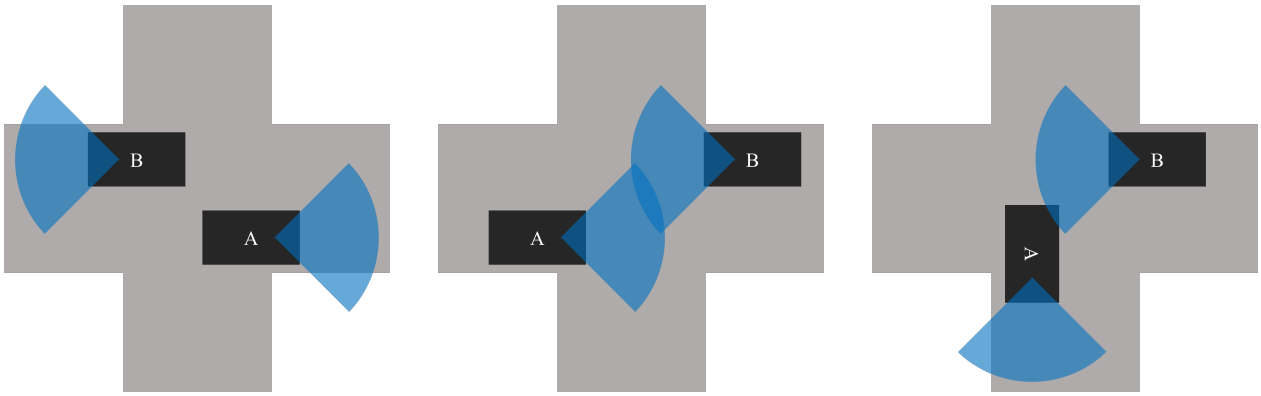
$$x_{ij} + \sum_{k\in V} x_{jk} \leq 1 \quad \forall i \in V \tag{42}$$
$$\sum_{j\in V} x_{ij} \leq 1 \quad \forall i \in V \tag{43}$$
$$\sum_{i\in V} x_{ij} \leq 1 \quad \forall j \in V \tag{44}$$
$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \tag{45}$$

where $x_{i,j}$ are binary decision variables indicating whether the vehicles i and j are matched or not.

We employ the Edmonds-Blossom algorithm [39] to solve this problem. Once the RSU establishes communication links, it must decide the direction of each communication.

Referring to Fig. 8, multiple situations can arise, and determining the proper roles for communicating vehicles significantly impacts the quality of transmitted information. We discuss the considered heuristics:

1. In Fig. 8a, vehicles travel in opposite directions with non-intersecting Regions of Interest (RoIs). Here, the likelihood that vehicles possess relevant information for each other is low. Consequently, the RSU excludes these communications when solving the matching problem.
2. In Fig. 8b, vehicles travel in opposite directions, and their RoIs intersect. In this case, both vehicles are expected to have meaningful information to send. Thus, the RSU randomly assigns the sender and receiver roles.



(a) (b) (c)

Figure 8: Various case scenarios of potential paired vehicles. The blue areas represent the vehicles' Regions of Interest (RoI).

3. In Fig. 8c, vehicles travel in the same direction. Here, it is anticipated that the leading vehicle (A) possesses more relevant information for the trailing vehicle (B). Consequently, the sender is designated as the leading vehicle (A).

6. Simulation results

In this section, we provide simulation results showing the effectiveness of the proposed GGCN-based point selection method. In Section 6.1, we provide implementation details of the developed simulation framework. Section 6.2 specifies the adopted training procedure. The experimental results on the generated simulation datasets are finally discussed in Section 6.3.

6.1. Simulation setup

To the best of our knowledge, the literature lacks a sizable open-source dataset for V2V cooperative perception. By treating the ego vehicle at various timestamps as many AVs, some work [40, 41] adapted the Kitti [42] dataset to mimic V2V settings. Due to the unpredictable appearance of dynamic agents at various locations, which causes temporal and spatial inconsistencies, such a synthetic approach is inappropriate and impractical for the setting addressed in this paper. Wang *et al.* [21] create a sizable V2V dataset by using a high-fidelity LiDAR simulator [43]. However, the dataset and the LiDAR simulator are inaccessible to the general public. Chen *et al.* [40, 41] propose the T&J dataset, a dataset based on real data by equipping two golf carts with 16-channel LiDAR to collect data. Nevertheless, the released version only includes 100 frames that lack ground truth labels and cover a limited range of road types.

Xu *et al.* [12] bring a significant contribution to the V2V open datasets by providing OPV2V, a V2V multi-modal dataset that contains different scenes involving multiple cooperative vehicles taken from the CARLA [14] open source simulator. To the best of our knowledge, this is the most complete V2V multimodal dataset available in the literature. However, it lacks scene extensibility and provides no information about vehicle communication links.

For the above reasons, we implement a new simulator providing sensor and communication data. Our simulator relies on the SUMO [13] simulator for the microscopic traffic simulation. We employ the CARLA [14] simulator to simulate the LiDAR acquisitions. Finally, we exploit GEMV^2 [15] to compute comprehensive information regarding the communication channels between vehicles. All the mentioned simulators are open source.

The diagram in Fig. 9 illustrates the simulator comprises multiple components. Here we briefly describe the simulation process:

1. We start from an OpenDRIVE file (xodr file extension) [44], a standardized format used to describe the logic of a street network.
2. Starting from the OpenDRIVE file, the `SumoSimulationGenerator` module generates a `sumocfg file` (SUMO configuration file), which describes the micro traffic logic of the simulation. SUMO then uses such a file to simulate the vehicular traffic.

3. We develop the `sumo_synchronization` module, which is an extension of the `Sumo_simulation` module provided by the CARLA's developers. The `Sumo_simulation` synchronizes the CARLA simulator and the SUMO simulator, using their APIs—respectively, the CARLA API and the `Traci` module.
4. The `sumo_synchronization` module manages the synchronization of the sensors and the actual simulation, ensuring consistency between the sensor acquisitions and the simulation—since is not guaranteed by the `Sumo_simulation` module.
5. The `Traci` module produces a `fcd output` file, which contains all the spatial information about each vehicle for each simulation timestep. Among the others, it provides each vehicle's position, heading, and speed.
6. Both opendrive and `fcd output` files are given as input to the GEMV^2 simulator. By utilizing this information, we can determine the potential communication partners for each vehicle at each timestep. Furthermore, for each potential communication, GEMV^2 generates information regarding the communication channel (`V2V communication dataset`), essential to determine the maximum data rate available.
7. The `V2V communication dataset`, the `fcd output` and the sensor acquisitions are then processed by the `Environment` module. Such module is responsible for the simulation of the system formalized in Section 3.

We build a synchronized LiDAR-V2V communications dataset by retrieving data from a simulation of 180 seconds and a step size of 0.05 seconds.

Selected evaluation scenario The simulation occurs around CARLA's map TOWN10, involving 60 vehicles of 4 different types. To recreate a realistic vehicle's type distribution, we include two sedans (Audi TT and Tesla Model 3), an SUV (Audi e-tron), and a truck (Mercedes Sprinter) from the vehicle types available in CARLA.

Vehicular traffic simulation The TOWN10 map is based on a close network. Therefore, it is much easier to generate a traffic jam, which might introduce a high redundancy in the data, inducing a bias. We identify the junction marked with a circle in Fig. 10 as a possible cause of traffic jams. More precisely, we observe that the road outlined by the blue box features limited space for vehicles awaiting the traffic light. This implies that it only accommodates a few vehicles before the entire space is occupied. Consequently, if a vehicle approaching from the junction intends to proceed toward that street, it will get stuck behind the existing vehicles, resulting in a queue even when the traffic light is green. For this reason, we decide to turn off the traffic lights at the junction in the circled area, making the queue to clear more efficiently in the blue rectangle. The parameters set up in SUMO to achieve the vehicular traffic simulations are reported in Table 2.

LiDAR simulation LiDAR simulation is performed in CARLA. Each vehicle mounts a LiDAR 20 cm above the vehicle's roof, as depicted in Fig. 3b. This sensor emulates a rotating LIDAR through ray-casting. The computation of points involves adding a laser for each plane distributed within the vertical field of view. The simulated rotation is achieved by calculating the horizontal angle through which the LiDAR rotates in a frame. The point cloud is generated by performing a ray-cast for each laser during each step. The simulation parameters used for the LiDAR simulation in CARLA are provided in Table 3. For further details, we refer the reader to the official CARLA documentation [45].

V2V communication channel simulation The simulation of the communication channel is achieved by means of the Geometry-based, efficient propagation model for vehicle-to-vehicle communication (GEMV^2) software [15]. GEMV^2 allows to directly import vehicular mobility information from the SUMO simulator floating car data (fcd). In order to perform V2V channel simulation, GEMV^2 requires the 2D outline of the buildings on the ground, assuming that the buildings are sufficiently tall with respect to the transmitters and receivers' positions—as it commony happens when considering vehicle-to-vehicle communications. To fulfil this condition, we extracted the mesh of buildings from CARLA Unreal Engine interface and we processed the mesh to represent the buildings by means of 3D bounding boxes. We then extracted the 2D building outlines intersecting with the ground plane and we used a modified version of GEMV^2 to import the generated outlines and to perform V2V channel simulation.

GEMV^2 distinguishes three kinds of links: (i) line of sight (LOS), i.e., direct communication paths between transmitter and receiver, (ii) non-LOS due to vehicles (NLOSb), and (iii) non-LOS due to buildings (NLOSb). Moreover, it allows to deterministically compute large-scale signal variations owing to path-loss and shadowing, and provides an approximation of small-scale signal variations using the number and the size of the objects that are around the communicating vehicles [15].

In a communication link where vehicle v is the transmitter and vehicle v' is the receiver, we retrieve through

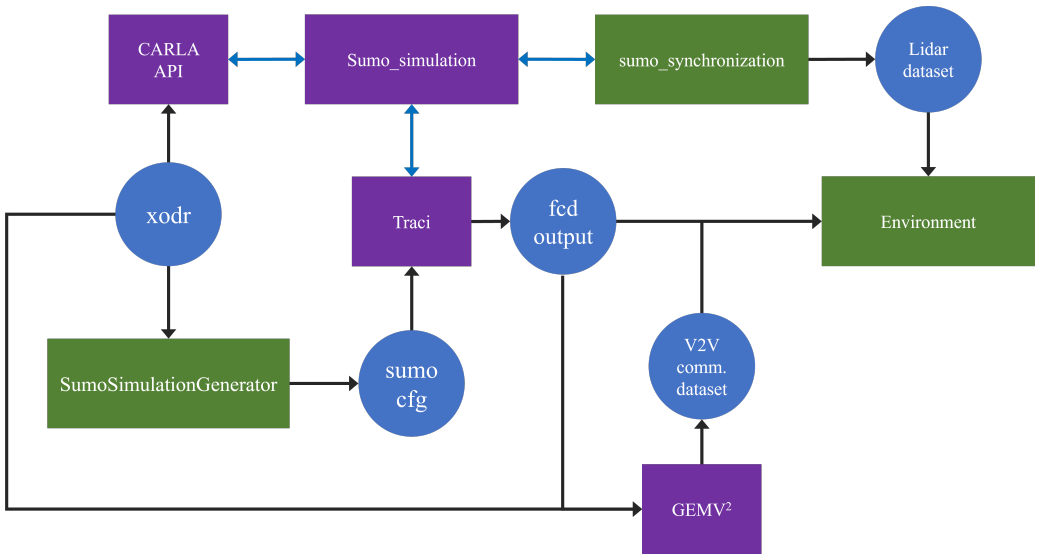


Figure 9: Block diagram for the simulator pipeline. A block colored with purple indicates an already implemented block. The green blocks are the components we implemented. The blue circles indicate a data type that can be an output or an input of a component. The blue arrows indicate that two components are communicating.

Parameter	Value	Parameter	Value
laneChangeMode	512	lcLookaheadLeft	10000
lcOpposite	1	sigma	0.5
lcSigma	0	departPos	random_free
jmIgnoreKeepClearTime	-1	departSpeed	max
jmIgnoreFoeProb	1	departLane	best
impatience	0	seed	3

Table 2: Vehicular generation settings in the SUMO traffic sumulation.

GEMV^2 the received power $P_{v',R}$ at v':

$$P_{v',R} = P^l_{v',R} + P^s_{v',R}, \tag{46}$$

obtained as the sum (in dB) of a large-scale signal variation term $P^l_{v',R}$ (depending on path-loss and shadowing) and a small scale signal variation term $P^s_{v',R}$ (owing to the geometry of the environment).

Table 4 reports the parameters selected for V2V channel simulation in GEMV^2. In particular, as assumed in Section 3.1, we notice that we employ a 28 GHz carrier frequency for simulation, which is set in the mmWave frequencies range.

6.2. Training procedure

We discuss here the procedure adopted to train the GGCN-based model assumed in Section 5.3. We split the training task into two complementary subtasks:

- Spatial task: the aim is to learn the relationship between the position of the sender vehicle, the receiver vehicle's position, and the sender's acquired points. In other words, we are interested in learning which, among the sensed points by the sender, are the points the receiver is spatially interested in.
- Temporal task: the goal is to learn the influence of points with high AoT on close points with low AoT. Thus, the sender vehicle should learn to avoid sending points belonging to the same spatial regions for consecutive steps.

Figure 10: Town10 CARLA scenario viability. The circle indicates the junction where traffic lights are turned off.

Parameter	Value	Parameter	Value
channels	64	horizontal fov	360°
rotation frequency ($1/\delta$)	20 Hz	atmosphere attenuation rate	0.004
points per channel	300	drop off general rate	0
range	50 m	drop off intensity limit	0
upper fov	10°	drop off zero intensity	0
lower fov	−30°	noise standard deviation	0

Table 3: LiDAR settings in the CARLA simulation.

In the following, we will refer to the whole training task as joint task.

We split the model training into two distinct training phases: the first phase involves performing offline training to learn the spatial task; during the second phase, we perform online fine-tuning to learn the joint task.

The reasons behind this decision are multiple. First, learning to solve the main task can be harder than its subtasks. Thus, the model can slowly converge when trained to solve the former. Furthermore, learning the main task can be time-consuming because the model must be trained online. In online training, the simulator introduces a nonnegligible overhead, which slows down the training procedure—a simulation step can require up to 4 times the time needed for our model forward pass. However, learning the spatial task does not require online training, as it does not demand a dataset capturing the temporal correlation between two consecutive timesteps of a communication.

It is essential to highlight that the simulator overhead is only due to the simulation of the environment and the computation of the novelty score $\eta(p)$. In contrast, the preprocessing on the point cloud is a set of vectorized operations that are computationally negligible. The computation of $\eta(p)$ does not represent a critical operation, as we assume our goal is to train the model in a simulated environment where the computation time is a soft constraint. Therefore, if the model is deployed in a real scenario, it would not require further training, meaning that no novelty scores must be computed.

We fine-tune our model in an online manner. For this purpose, the simulation is split into n episodes of horizon h. Each episode is composed of several trajectories, each representing a communication between two vehicles. For each episode, we predefine the vehicle pair communications. Vehicles can communicate only with the predefined paired vehicle for the entire episode. At the end of the trajectory, the state of the point cloud is reset, which means, at the beginning of each trajectory, acquired points of each vehicle are the ones acquired at

Parameter	Value	Parameter	Value
Max. communication range	100 m	Max. NLOSv range	90 m
Max. NLOSb range	80 m	Antenna height (on top of the vehicles)	0.1 m
Carrier frequency	28 GHz	Antenna pattern	isotropic
Tx power	12 dBmW	Use reflections and diffractions	true
NLOSf	2.7	NLOSb	2.9
NF	6 dB	Δt	0.001 s
τ_{RSU}	0.5 s	τ_v	0.05 s

Table 4: GEMV^2 simulation settings.

the beginning of that trajectory, i.e., points from previous timesteps are forgotten, and the AoT of each point is set to 0.

The sender vehicle sends data to the paired vehicle using an ϵ-greedy policy. Thus, given the points $\mathcal{P}_s$ owned by the sender vehicle, it will send a set of points $\mathcal{P}_{\epsilon-max}$ with cardinality ν given by eq. (10), such that $100 \cdot (1-\epsilon)\%$ of them maximizes:

$$\bar{\mathcal{S}}(p) = \mathcal{R}_{v_{rx}}(p) \cdot \alpha(p) \cdot \mathcal{M}(p, \mathcal{G}), \tag{47}$$

where $\mathcal{M}$ is the proposed model, which approximates the function defined by eq. (37). We employ an ϵ-greedy policy to improve the exploration of the agents.

To build our offline dataset, we collect the data from a simulation where trajectories have a horizon $h = 2$ and no bandwidth constraint, meaning all the points are sent. We believe that the first step of the trajectory is important to make the model learn the spatial task. In contrast, the second step aims to alert the model to the significance of AoT as a relevant feature. Consequently, we aim to prevent the model from setting all parameters related to AoT to 0.

We can predefine the matched vehicle communication pairs for each episode, as stated in Section 5.4. Since all trajectories come from the same simulation, a high spatial correlation exists between two temporally contiguous episodes. Hence, maintaining the same matched vehicles across temporally contiguous episodes can result in highly correlated trajectories, which can have a negative impact during the training procedure. Indeed, having two highly correlated trajectories in the training set reduces the variety of data, which is essential for the training process. At the same time, having one trajectory in the training set and one highly correlated to it in the validation set must be avoided, as the metrics measured on the validation set would not be reliable.

On the other hand, if we force to have a communicating pair assigned only to a single episode, we risk having several trajectories involving matches between vehicles that are too far to transmit meaningful information. For this reason, we adopt the following heuristic to minimize the correlation among trajectories without losing relevant data:

1. We split the simulation into n episodes of h timesteps without overlaps.
2. We iterate over all the episodes in temporal order.
3. For each episode, by exploiting the Edmonds-Blossom algorithm, we compute the maximum matching, which defines the trajectories for the analyzed episode.
4. Once the matching is determined, we remove all the involved pairs from the available pairs for the following ten contiguous episodes.
5. Finally, we insert two trajectories involving the same pair less distant than 15 seconds (300 timesteps in our setting) into the same set. This heuristic further reduces the correlation between trajectories belonging to different datasets.

6.3. Experimental results

After validating the GGCN architecture on a segmentation task based on data from our simulator (see Appendix A), we believe the hyperparameters used by the GGCN's authors are a good starting point for our experiments. For what concerns the hyperparameters of ϕ^u multilayer perceptrons (MLPs), we decide to use the same hyperparameters used for the MLPs $\mathcal{M}$ defined by eq. (5), which are used to extract point features. We report all the architecture hyperparameters in Table 5.

Offline training For offline training, the Adam optimizer is employed with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-5}, and a learning rate of 10^{-3}. Furthermore, we perform data augmentation to improve the generalization. Data augmentation involves random rotations along the z-axis on the point clouds, maintaining geometric consistency by applying a corresponding rotation on the position and heading of the receiver embedded in the global node.

As shown in Fig. 11, the model exhibits fast learning in the initial epochs, with a slowdown in progress after epoch 5. The training accuracy gradually increases from 85% to 90% over nearly 50 epochs, while the validation accuracy peaks at 87%, indicating minimal overfitting despite the model's moderate performance on the training set.

Online training For online training, the learning rate of the Adam optimizer is set to 10^{-4}, and training is conducted on episodes with a horizon of $h = 10$. As we can see from Fig. 12, learning the joint task starting with a model pre-trained on the spatial task is easier than training a model only on the joint task. Indeed, the fine-tuned model achieves an 83% accuracy after a few epochs of online training. On the other hand, the model trained from scratch shows slow learning, which is prohibitive, given the computation constraints.

Joint task relevance To illustrate the significance of learning the joint task as opposed to just the spatial task, we compare the Transfer Learning (TL) model (trained only on the spatial task) against the model fine-tuned on the joint task. Fig. 12 shows that a model trained to learn the spatial task drops in accuracy by about 13% if used to solve the full task. Nevertheless, the pre-trained model is a decent start as it shows 74% accuracy in the main task before the fine-tuning. Fig. 14a shows the main drop in accuracy is caused by the following steps of the communication, as we expect. In contrast, in Fig. 14b, the fine-tuned model shows almost the same accuracy for all the steps during the communication. This is also confirmed by Fig. 13a. Indeed, the accuracy drop is induced by the fact that the number of redundant points sent increases with the simulation steps. Furthermore, as we can see from Fig. 13b, it is interesting to note that the TL model proves to be more effective than the fine-tuned model during the first step of communication. This phenomenon indicates a catastrophic forgetting problem during the fine-tuning.

Training for different bandwidths Finally, we compare the training in a simulated scenario for different communication bandwidths. From Table. 6, we assess that as the band used in the simulation varies, the performance of the models does not vary significantly. The maximum accuracy in the validation set is 0.83%, achieved with a bandwidth of 5 MHz, while a little drop in the performance is perceived for higher bandwidths.

Computation time We notice that the primary challenges involve the speed of convergence and the time required for training. Training the model using a Nvidia Quadro RTX 6000 GPU takes approximately one hour per epoch for offline training and up to four hours per epoch for online training. The difference in computation time between the online and offline train highlights the heavy overhead introduced by the simulator. Computational constraints prevent exhaustive research over hyperparameters to be performed by means of heuristic approaches.

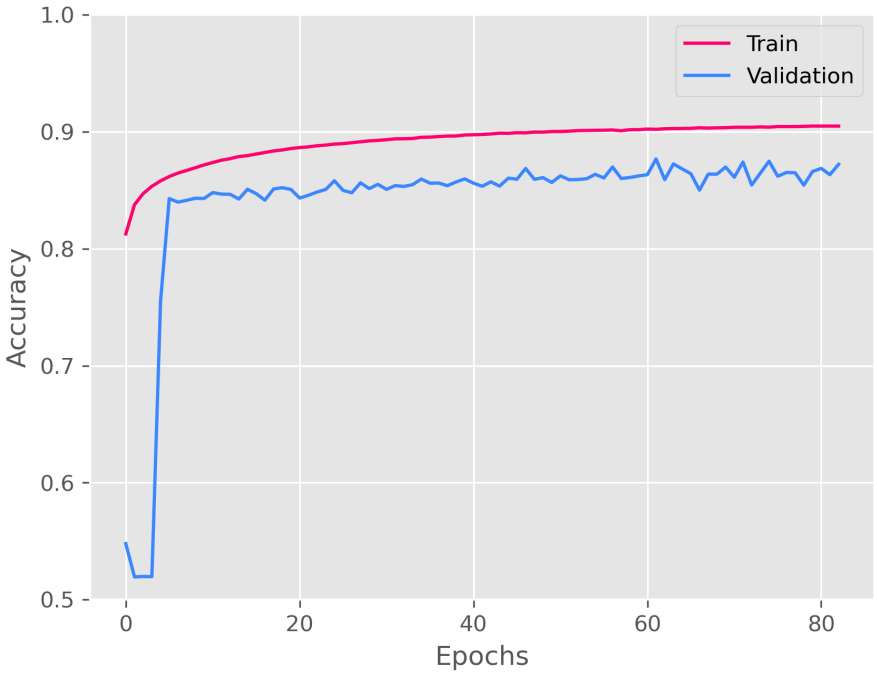


Figure 11: Accuracy curve during the offline training.

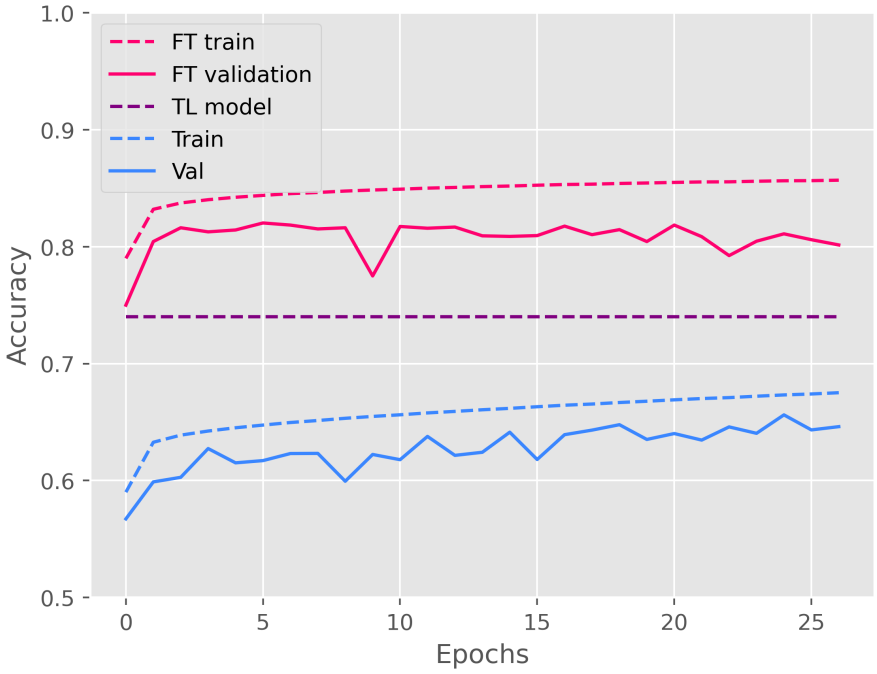
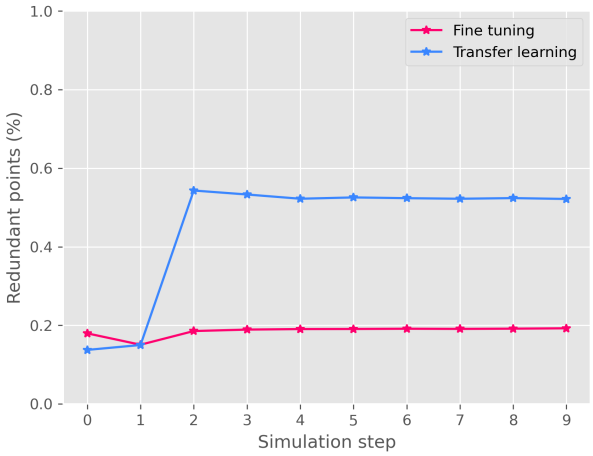
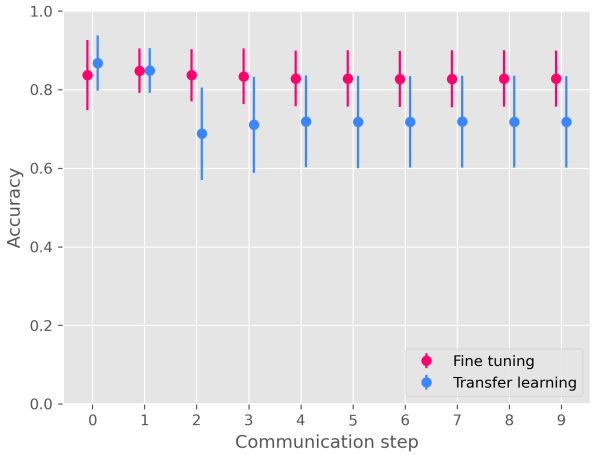


Figure 12: Accuracy curve during the online training. The dashed purple line indicates the performance of the model trained exclusively on the spatial task. The learning curves compare the fine-tuning against a model directly trained for the complete task.

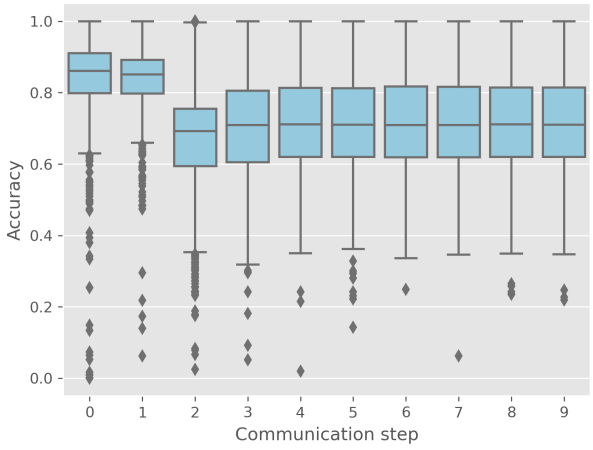


(a) Percentage of redundant points sent per communication step. The percentage is computed as the ratio between redundant points sent and the total points sent.

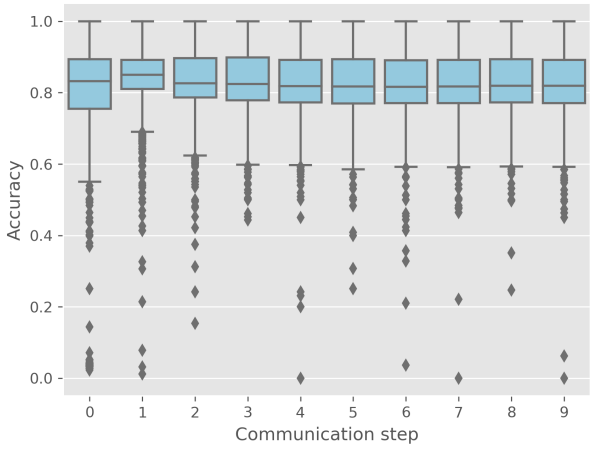


(b) Winsored error bars measuring the accuracy. Error bars are not perfectly aligned to their corresponding step tick. This is intended to make more clear the comparison of the two models for the same tick.

Figure 13: Performance comparison of the fine-tuning model against the transfer learning model across all the communication steps.



(a) Accuracy boxplot of fine-tuning in validation.



(b) Accuracy boxplot of fine-tuning in validation.

Figure 14: Distribution of the accuracy per point cloud for each step of the communication.

Hyperparameters	$\mathcal{L}_1^{\downarrow}$	$\mathcal{L}_2^{\downarrow}$	$\mathcal{L}_3^{\downarrow}$	$\mathcal{L}_4^{\uparrow}$	$\mathcal{L}_5^{\uparrow}$	$\mathcal{L}_6^{\uparrow}$
v_x	0.05	0.13	0.4	0.4	0.13	0.05
v_y	0.05	0.13	0.4	0.4	0.13	0.05
v_z	0.05	0.13	0.4	0.4	0.13	0.05
M	1024	256	24	256	1024	15000
K	64	32	32	5	5	5
ϕ^u output neurons	$[32, 32, 64]$	$[64, 64, 128]$	$[128, 128, 256]$	$[128]$	$[128]$	$[128]$
$\mathcal{M}$ output neurons	$[32, 32, 64]$	$[64, 64, 128]$	$[128, 128, 256]$	$[128]$	$[128]$	$[128]$
$\mathcal{M}_{sem}$ output neurons	$[16]$	$[32]$	$[64]$	$[32]$	$[32]$	$[32]$
$\mathcal{M}_{geo}$ output neurons	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{M}_{att}$ output neurons	$[64]$	$[128]$	$[256]$	$[128]$	$[128]$	$[128]$
$\mathcal{A}_{ctx}$ type	max	max	max	max	max	max
$\mathcal{A}$ type	max	max	max	max	max	max
$\rho^{e \to u}$ type	max	max	max	max	max	max

Table 5: Hyperparameters for GGCN. $\mathcal{L}_i^a$ is the i-th layer of the model and the a indicates whether the layer is an upsampling layer ($a = \uparrow$) or a downsampling layer ($a = \downarrow$). For the terminology in this table, refer to Section 2.3 and 5.3.

Total bandwidth	**Train accuracy**	**Validation accuracy**
100 MHz	0.86	0.82
50 MHz	0.84	0.80
20 MHz	0.84	0.79
5 MHz	0.87	0.83

Table 6: Model accuracies trained at different bandwidths. The values in the table refer to the model performance after convergence, with patience set to 10.

7. Conclusions

In this paper, we introduced a cooperative perception method wherein connected vehicles effectively choose LiDAR points to transmit, mitigating network overload. Our approach involves learning by means of a graph neural network the points that the receiving vehicle is interested in but cannot perceive due to occlusions. Additionally, we proposed the concept of the Age of Transmission (AoT) to reduce redundant data transmission across multiple communication steps.

We developed a simulation framework based on the SUMO vehicular simulator, the CARLA automotive simulator, and the GEMV^2 communication channel simulator to generate a realistic synchronized dataset of LiDAR acquisitions and V2V channel data. Experimental results evaluated on this dataset show that our algorithm can detect important areas that cannot be perceived by the receiver vehicle with mean 81% validation accuracy over different communication bandwidths. Furthermore, it is shown that by introducing the AoT, data redundancy is minimized as, on average, only 20% of the available redundant points are sent. Comparing the model over different communication bandwidths, we noticed that the performance of the model does not vary significantly, with a maximum validation accuracy of 83% achieved for a 5 MHz bandwidth and a little drop in performance for higher bandwidths up to 100 MHz.

While the findings are promising, certain limitations highlight areas for potential improvement. Firstly, the model is trained on data from a single simulation within the same urban scenario. Expanding the training to encompass multiple scenes is expected to enhance the model's performance. Additionally, the substantial overhead introduced by the simulator prevents on exhaustive exploration of the hyperparameters. As a prospective work, we suggest a more in-depth investigation into new hyperparameter settings and different architectures is warranted.

References

[1] SAE international. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. , Jan. 2021.

[2] Ralph H Rasshofer and Klaus Gresser. Automotive radar and lidar systems for next generation driver assistance functions. *Advances in Radio Science*, 3:205–209, 2005.

[3] Damon Lavrinc. This is how bad self-driving cars suck in rain. *Jalopnik. Accessed: May*, 9:2021, 2014.

[4] U.S Department of Transportation. Technical report, tesla crash. 2017. URL `Available:https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF`.

[5] Alex Davies. Google's self-driving car caused its first crash. In *Wired*. 2016.

[6] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. doi: 10.1109/ACCESS.2020.2983149.

[7] Jihong Park, Sumudu Samarakoon, Hamid Shiri, Mohamed K Abdel-Aziz, Takayuki Nishio, Anis Elgabli, and Mehdi Bennis. Extreme urllc: Vision, challenges, and key enablers. *arXiv preprint arXiv:2001.09683*, 2020.

[8] Michele Rondinone, Thomas Walter, Robbin Blokpoel, and Julian Schindler. V2x communications for infrastructure-assisted automated driving. In *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 14–19. IEEE, 2018.

[9] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang Jr. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.

[10] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5661–5670, 2020.

[11] Shunsuke Aoki, Takamasa Higuchi, and Onur Altintas. Cooperative perception with deep reinforcement learning for connected vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 328–334. IEEE, 2020.

[12] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589. IEEE, 2022.

[13] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL `https://elib.dlr.de/124092/`.

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[15] M. Boban, J. Barros, and O. Tonguz. Geometry-based vehicle-to-vehicle channel modeling for large-scale simulation. *IEEE Transactions on Vehicular Technology*, 63(9):4146–4164, Nov 2014. ISSN 0018-9545. doi: 10.1109/TVT.2014.2317803.

[16] Shunli Ren, Siheng Chen, and Wenjun Zhang. Collaborative perception for autonomous driving: Current status and future trend. In *Proceedings of 2021 5th Chinese Conference on Swarm Intelligence and Cooperative Control*, pages 682–692. Springer, 2022.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

[18] Mohamed K Abdel-Aziz, Cristina Perfecto, Sumudu Samarakoon, Mehdi Bennis, and Walid Saad. Vehicular cooperative perception through action branching and federated reinforcement learning. *IEEE Transactions on Communications*, 70(2):891–903, 2021.

[19] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence*, volume 32, 2018.

[20] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1852–1864, 2020.

[21] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 605–621. Springer, 2020.

[22] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[24] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[25] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017.

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[27] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12): 4338–4364, 2020.

[28] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

[30] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), oct 2019. ISSN 0730-0301. doi: 10.1145/3326362. URL `https://doi.org/10.1145/3326362`.

[32] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[33] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[34] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[35] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[36] Don O Loftsgaarden and Charles P Quesenberry. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

[37] Barnabás Póczos and Jeff Schneider. On the estimation of α-divergences. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 609–617. JMLR Workshop and Conference Proceedings, 2011.

[38] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[39] Zvi Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys (CSUR)*, 18(1):23–38, 1986.

[40] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524. IEEE, 2019.

[41] Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, SEC '19, page 88–100, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367332. doi: 10.1145/3318216.3363300. URL `https://doi.org/10.1145/3318216.3363300`.

[42] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[43] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176, 2020.

[44] Marius Dupuis, Martin Strobl, and Hans Grezlikowski. Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks. In *Proc. of the Driving Simulation Conference Europe*, pages 231–242, 2010.

[45] Carla documentation. `https://carla.readthedocs.io/en/latest/ref_sensors/#lidar-sensor`. Accessed: 2023-11-20.

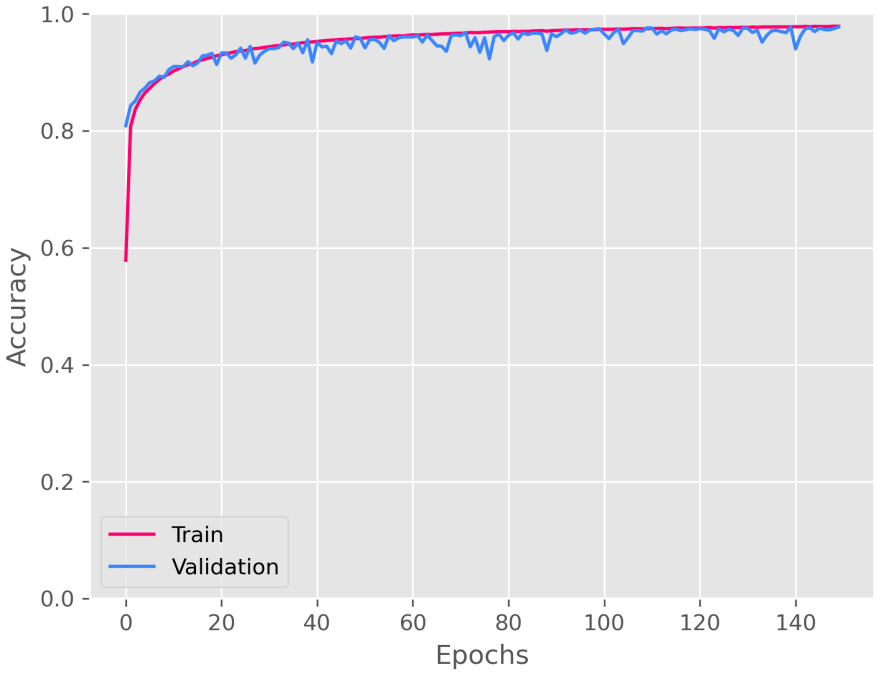


Figure 15: Accuracy curves during training of GGCN for multiclass segmentation task.

A. Grid-GCN validation on segmentation tasks

In this appendix, we present the results of training and validating the Grid-GCN model on the dataset generated by our simulator for a semantic segmentation task. Fig. 15 and Fig. 16 illustrate the accuracy and categorical cross-entropy during the training phase, demonstrating remarkable performance, with a peak validation accuracy of 97%.

The confusion matrix for the best model, displayed in Fig. 17, indicates an overall good understanding of the semantic meaning of objects. However, there are some classes where the model struggles with correct classification, particularly confusing sidewalks with roads. Additionally, there is difficulty distinguishing poles and traffic signs from traffic lights. Nevertheless, we consider these errors negligible as the confused classes are quite similar.

B. Problem's hyperparameters

In this appendix, we provide a brief discussion of how the hyperparameters for the problem have been selected.

B.1. Novelty score hyperparamters

The density estimator based on the kNN algorithm:

$$\hat{f}_{\mathcal{U}}(p) = \frac{k}{m\bar{c}\rho_k(p)} \cdot \frac{1}{c}, \tag{48}$$

depends on two hyperparameters: the k and c. In the following paragraphs, we describe how we tune these hyperparameters.

kNN tuning The density estimator based on the kNN algorithm depends on the hyperparameter k of the kNN algorithm. The wrong choice of the k hyperparameter can lead to a noisy and unreliable density estimation. To understand the degree of reliability for a given k, we rely on the following heuristic. Given a point cloud $\mathcal{P}$, we compute the $\hat{f}_{\mathcal{U},k}(p)$ estimator for each point $p \in \mathcal{P}$ for $k \in [0, 30]$. Then, for each point, we define a sliding window of size $w = 3$, and we compute the estimator variance as follows:

$$\sigma_k(p) = \mathbb{V}ar\left[\mathbf{f}_k(p)\right] \quad \forall p \in \mathcal{P}, \forall k \in [0, 28]\,, \tag{49a}$$

$$\mathbf{f}_k(p) = \{\hat{f}_{\mathcal{U},k+i}(p) \mid i \in [0, 1, 2]\}. \tag{49b}$$

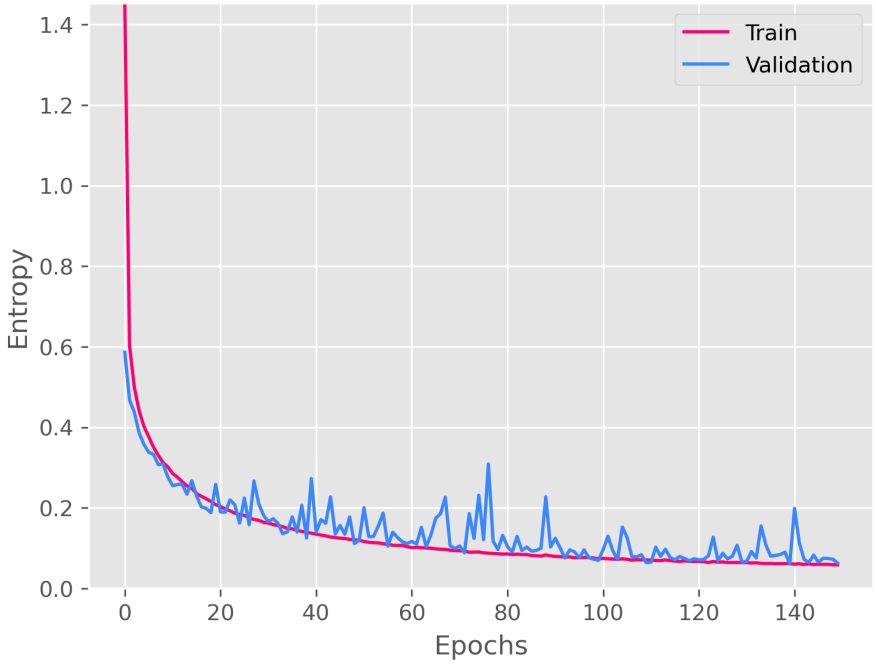


Figure 16: Accuracy curves during training of GGCN for multiclass segmentation task.

Here, $\sigma_k(p)$ measures how sensible is the estimator $\hat{f}_{\mathcal{U},k}(p)$ for a value k and point p. The higher is $\sigma_k(p)$, the more sensible is the estimator. Finally, for each k we average $\sigma_k(p)$:

$$\begin{aligned} \hat{\sigma}_k &= \mathbb{E}\left[\sigma_k\right], \quad \forall k \in [0, 28], \\ \sigma_k &= \{\sigma_k(p) \mid \forall p \in \mathcal{P}\}. \end{aligned} \tag{50}$$

This average quantifies the average estimator sensibility for each value k. From Fig. 18, we note that for $k > 5$, the sensibility is negligible. We decide to choose $k = 8$ for computational efficiency. Indeed, high values of k increase the computation time for the estimator.

Normalization constant The normalization constant c distributes the density values in the interval $[0, 1]$. This is done so that points belonging to dense regions have a value $\hat{f}_{\mathcal{U}}(p)$ close to 1, while points in sparse regions have $\hat{f}_{\mathcal{U}}(p)$ close to 0. From Fig. 19a, we notice the estimator distribution is strongly skewed, and there is a relevant presence of outliers with high values. Setting c to the highest value risks pushes the majority of estimations towards 0, even for points in dense regions. From our experiments, setting c to the 25th percentile of the estimator values leads to reasonable results. As we can see from Fig. 20, for higher values than the 25th percentile, even points close to the vehicle—which belongs to the most dense regions—are set to low-density values.

B.2. RoI hypeparamters

The choice of the hyperparameters used to model the RoI are found empirically so that the shape of the RoI reflects the assumption we made in Section 5.1. In Fig.s **??** we show how the RoI shape changes as the main hyperparameters κ and γ changes.

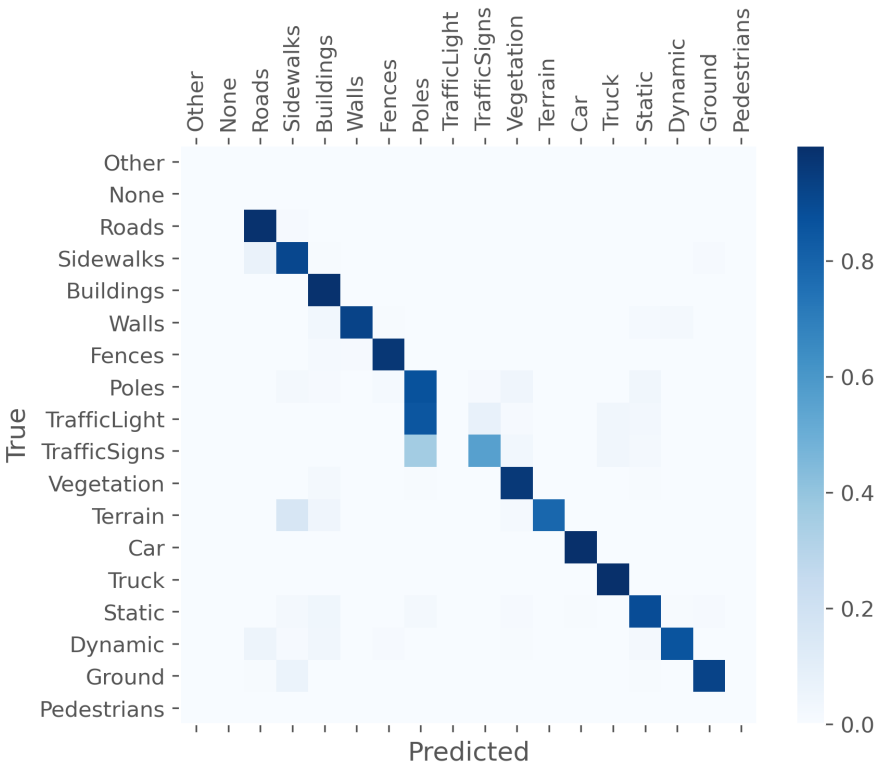


Figure 17: Confusion matrix of GGCN for multiclass segmentation task.

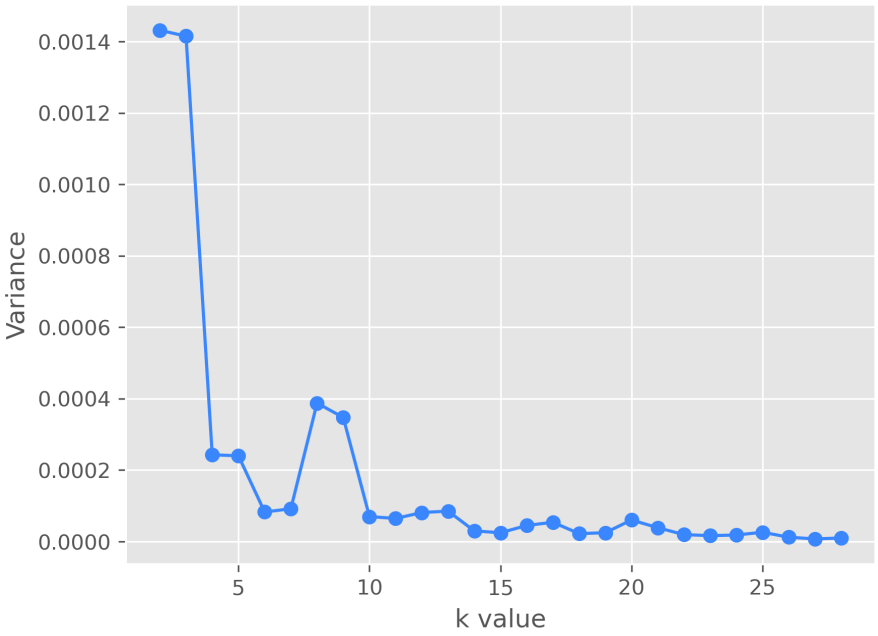
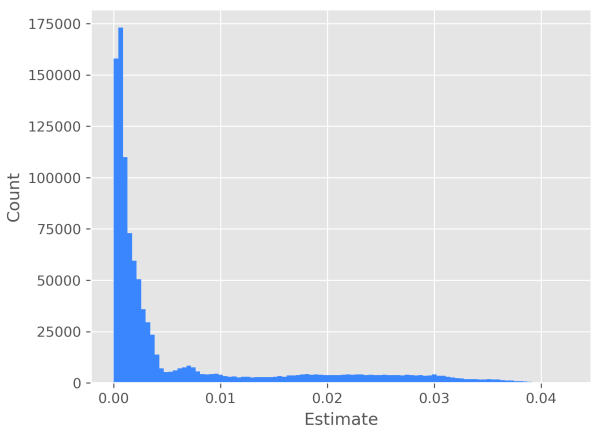
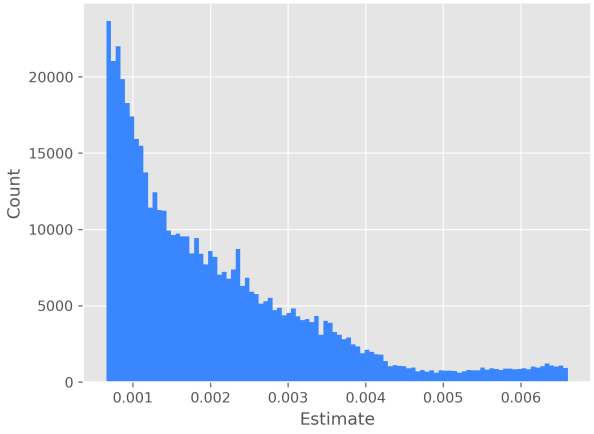


Figure 18: Representation of the density estimator variance for each k. Less variance implies more reliability



(a) Distribution of the estimator values.



(b) Distribution of the estimator values inside the interquartile range.

Figure 19

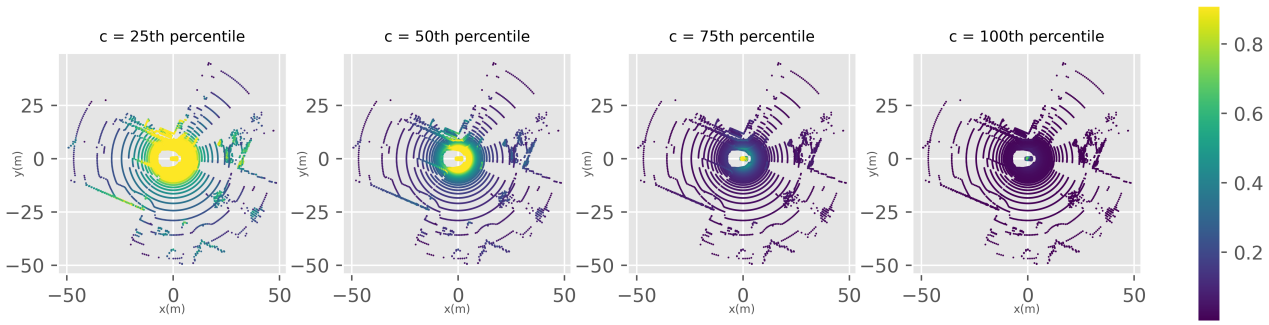


Figure 20: Point cloud plots. The color of the point is related to its estimated density. Brighter colors are associated with high density and darker colors with low density. These plots compare the estimated density value when the constant normalization changes.

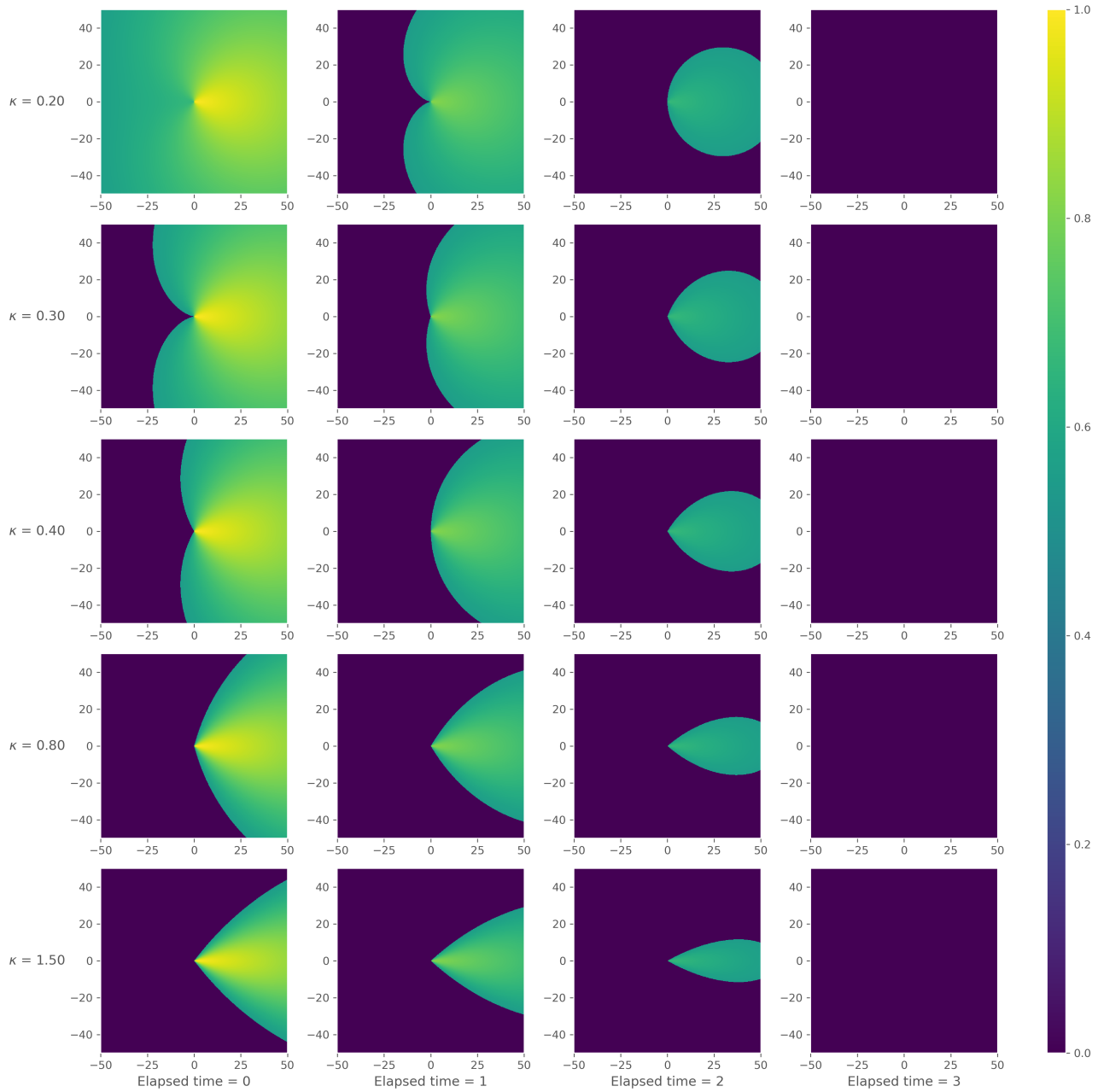


Figure 21: Heatmpas representing the RoI as κ and the time elapsed change.

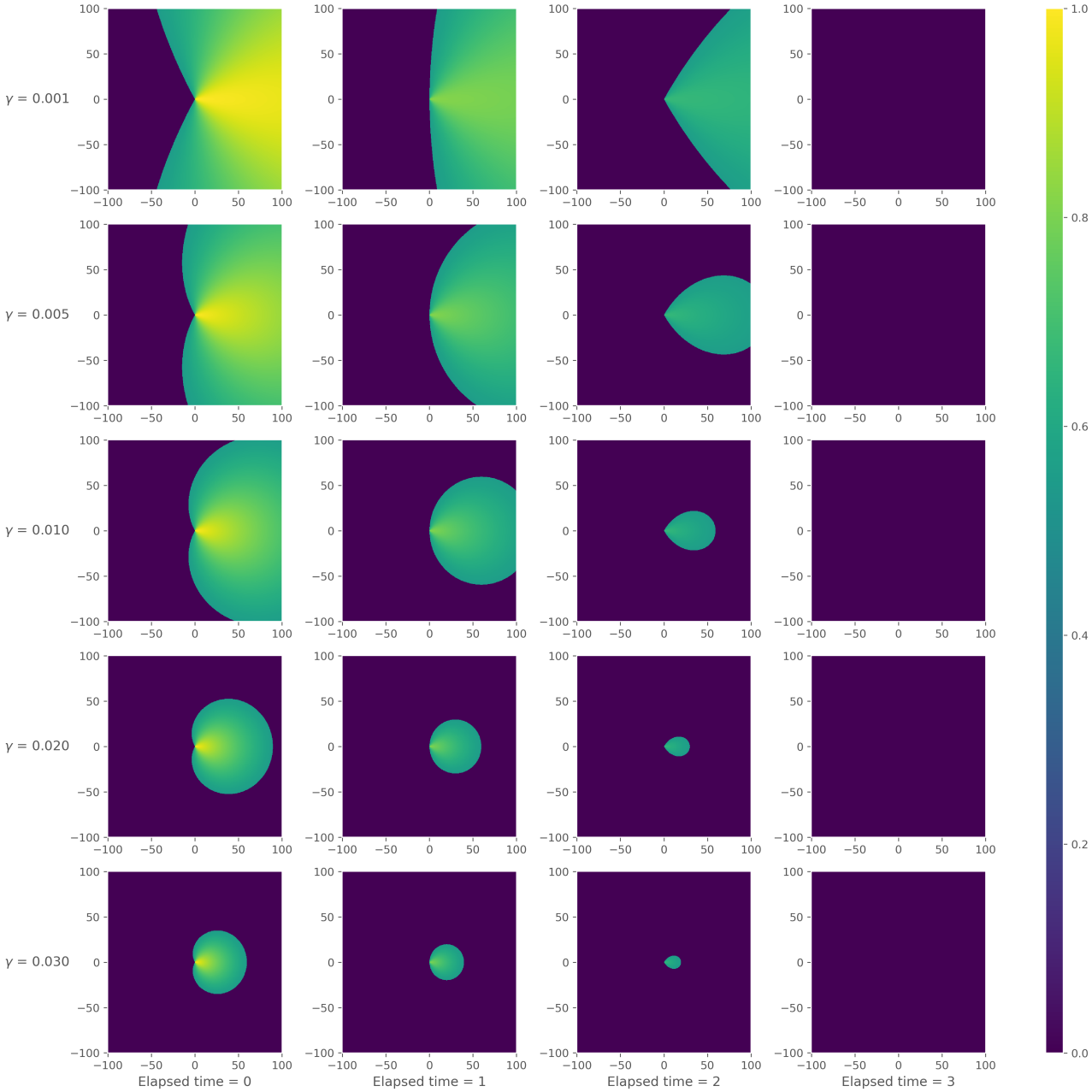


Figure 22: Heatmpas representing the RoI as γ and the time elapsed change.

Abstract in lingua italiana

L'imminente ascesa della guida autonoma, la cui realizzazione è prevista entro il 2030, promette un'era trasformativa caratterizzata da maggiore sicurezza, comfort ed efficienza operativa. Il percorso che porta dall'assistenza alla guida ai sistemi completamente autonomi presenta delle sfide e la tecnologia LiDAR svolge un ruolo cruciale. In questo articolo esploriamo le strategie di percezione cooperativa per le nuvole di punti acquisite tramite LiDAR, affrontando sfide come l'occlusione e la condivisione limitata dei dati. Gli attuali lavori sulla percezione cooperativa propongono un contesto in cui ogni veicolo condivide la posizione degli oggetti rilevati. Questi metodi presentano errori di stima e rumore significativi, dovuti a un'osservazione locale insufficiente. Altri studi sviluppano reti neurali per comprimere e ricostruire intere nuvole di punti, minimizzando l'errore di ricostruzione. Anche se questi studi si dimostrano efficaci, il corretto funzionamento dell'algoritmo richiede la sua implementazione su tutti i veicoli connessi. Lo scopo di questo lavoro è quello di studiare un approccio alternativo in cui i dati grezzi possano essere trasmessi senza fare affidamento su quest'ultimo requisito. Progettiamo e testiamo un algoritmo di selezione dei punti basato su una rete neurale a grafo che mira a identificare quali punti appartenenti a una nuvola di punti acquisita da un veicolo meritano di essere trasmessi a un altro veicolo. I nostri esperimenti sono stati condotti in uno scenario urbano veicolare simulato, basato su simulatori realistici di LiDAR e di comunicazioni V2V. I risultati sperimentali mostrano che il nostro algoritmo è in grado di rilevare aree importanti che non possono essere percepite dal veicolo ricevente con un'accuratezza dell'81%, riducendo la trasmissione di punti ridondanti. Sono state riconosciute le sfide relative alla velocità di convergenza dell'addestramento e alla ricerca di iperparametri, suggerendo strade per ulteriori sviluppi.

Parole chiave: percezione cooperativa, LiDAR, veicoli autonomi, segmentazione di nuvole di punti, reti neurali a grafo

Acknowledgements

I want to express my heartfelt gratitude to Professor Matteucci for opening the door to this thrilling field and giving me the chance to delve into it. A special thank you to Lorenzo for being my guide throughout these intense nine months, offering invaluable advice and support through the challenges of my thesis. To my parents, Roxana and Mihail, your unwavering cheer and support over the years have meant the world to me. Big thanks to my friends Andrea and Federico for standing by me, even when I couldn't always be present. And to all my friends at Politecnico di Milano, thank you for sharing this long journey as we pursued our dream of becoming engineers. I am truly grateful for the wonderful people who surrounded me during these years.