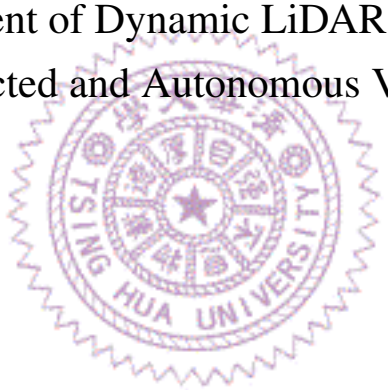
國立清華大學電機資訊學院資訊系統與應用研究所
碩士論文
Institute of Information Systems and Applications
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

用於聯網自動駕駛汽車的動態光達點雲錯誤隱藏
Error Concealment of Dynamic LiDAR Point Clouds for Connected and Autonomous Vehicles



石桂華
Guihua Shi

學號：110065466
Student ID:110065466

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 112 年 6 月
June, 2023

國立清華大學
資訊系統與應用研究所

碩士論文

用於聯網自動駕駛汽車的動態光達點雲錯誤隱藏

石桂華

Acknowledgments

Time flies, and my two-year master's life is coming to an end. Looking back on the past few years of studying, I deeply feel my growth under the care of teachers, classmates, relatives, and friends. Here, I would like to thank all the people during my time at university and master's study. First, I would like to thank my college friends for adapting me to life. Then I would like to thank all the people in the lab, who have accompanied me through these significant two years. Thanks to Chih-Chun Wu and Sheng-Ming Tang for adding a lot of fun to my research life in the past two years, discussing research issues, eating good food, and chatting together. Special thanks to Chih-Chun Wu, who took the trouble to help me with the paper-related matters during the submission period. Finally, I would like to thank Prof. Hsu, who has always helped me point out the direction and pointed out my mistakes. In terms of writing, he helped me to revise it meticulously and attached a lot of advice. Thanks to National Tsinghua University, thanks to the lab, thanks to classmates, and thanks to professors, you are an indispensable part of my life over the past few years. In addition, I would like to thank my parents and girlfriend. I am sorry that I did not go home during COVID-19, but you will not complain. You have always trusted me and supported me, which allows me to do research with peace of mind.

致謝

時光飛逝，兩年碩士生活已接近尾聲。回想起唸書這幾年的點點滴滴，我深深感受到自己在老師、同學和親友們關愛下的成長。在此，我要感謝我在大學和研究所期間認識的所有人。首先，我要感謝我的大學朋友們，謝謝你們帶我融入生活。然後我要感謝實驗室中所有的人，你們陪伴我度過了這非常具有意義的兩年。謝謝吳仕群和唐盛銘，在這兩年中給我的研究生活增添了許多色彩，一起討論研究問題，一起吃好料，一起閒聊。特別感謝吳仕群，在投稿期間不厭其煩地幫我處理論文的相關事宜。最後，我要感謝徐正炘教授，您一直幫我指出錯誤和指明方向，在寫作方面更是細緻的幫我檢查並附上滿滿的建議。感謝清華大學、感謝實驗室、感謝同學們、感謝教授，你們是我這幾年生活不可或缺的一部分。另外，我要感謝我的家人和女朋友，很抱歉在疫情期間都沒有回家，但你們不會抱怨，一直都很信任我、支持我，這讓我可以安心地做研究。

Abstract

Connected and Autonomous Vehicles (CAVs) often come with sensors, such as LiDARs, to improve road safety. Although dynamic LiDAR point clouds could be streamed over wireless networks from CAVs to edge servers for computationally-intensive classification tasks, the problem of incomplete point cloud frames caused by unreliable wireless networks has yet to be investigated in the literature. In this thesis, we propose the very first LiDAR Error Concealment (LEC) algorithm, to conceal incomplete point cloud frames due to lost or late packets for minimizing the Chamfer distance between the concealed and original point cloud frames. Driven by machine learning techniques, our LEC algorithm adaptively performs Temporal Prediction (TP), Spatial Interpolation (SI), or Temporal Interpolation (TI) to conceal incomplete point cloud frames by comparing the incomplete ratios of adjacent point cloud frames. We evaluate the performance of our LEC algorithms with a comprehensive co-simulator built upon the popular CARLA and NS-3 and the KITTI Odometry dataset. Our simulation results reveal that the proposed LEC algorithm outperforms the TP, SI, and TI algorithms by up to 82.68% and 30.17% in Chamfer and Hausdorff distances, and terminates in 360–570 ms in a C-V2X network. Moreover, our LEC algorithm also outperforms other algorithms by up to 87.43% and 66.58% in Chamfer and Hausdorff distances in a DSRC network.

中文摘要

聯網自動駕駛汽車 (CAV) 通常配備各種感測器（RGB相機、光達、毫米波雷達等），以提升駕駛安全性。儘管動態光達點雲可以透過無線網路從 CAV 流式傳輸至邊緣伺服器以執行復雜運算的分類任務，但文獻中尚未討論由不可靠的無線網路引起的點雲幀不完整的問題。在本文中，我們首創了一個光達的錯誤隱藏 (LEC) 算法，以修復由於丟失或延遲封包導致的不完整點雲幀，以最小化點雲幀與原始點雲幀之間的倒角距離。利用機器學習技術，我們的 LEC 算法透過比較相鄰點雲幀的不完整比率自動地執行時間預測 (TP)、空間插值 (SI) 或時間插值 (TI) 等策略來修復不完整的點雲幀。我們使用由我們實作的協同模擬器以及 KITTI 里程計資料集來評估我們的 LEC 算法的效能。我們的模擬結果表明，所提出的 LEC 算法在倒角和豪斯多夫距離方面優於 TP、SI 和 TI 算法高達 82.68% 和 30.17%，並且在 C-V2X 網路中執行時間始終介於360–570 ms。此外，我們的 LEC 算法在 DSRC 網路中的倒角和豪斯多夫距離方面也優於其他算法高達 87.43% 和 66.58%。

Contents

List of Figures

List of Tables


TSING HUA UNIVERSITY

Chapter 1

Introduction

Connected and Autonomous Vehicles (CAVs) leverage Machine Learning (ML) algorithms for various *classification tasks*, such as object detection [16, 60], semantic segmentation [16, 60], shape classification [70, 83], lane detection [42, 62], and object tracking [56, 59] to improve road safety. These ML algorithms analyze data from sensors to understand scenes, to enable different degrees of driving automations, and avoid accidents due to driver fatigue [29, 77]. While various data representations can be used as inputs to these classification tasks, it is found that emerging 3D representations carry richer information than traditional 2D representations, leading to higher classification accuracy [1, 24]. Among 3D representations high-precision 3D point clouds can be natively produced by LiDARs (Light Detection and Ranging), and are more suitable CAVs. Each LiDAR point cloud is represented by a set of reflection points, where each point p consists of coordinates $p.x$, $p.y$, and $p.z$ and intensity $p.i$ measured by a rotating laser beam.

Although modern vehicles often come with LiDARs, they do not have to work alone when it comes to *scene understanding*. Instead, wireless networks are often used to facilitate data exchanges among vehicles for *cooperative perception* [18, 36] to avoid misclassification due to vision occlusions. Oftentimes, LiDAR point clouds are streamed from multiple vehicles over wireless networks, such as DSRC (Dedicated Short Range Communications) [40] and NR C-V2X (New Radio Cellular Vehicle-to-Everything) [15], to a more resourceful edge server for analysis. Such *vehicular networks*, however, support a limited total bandwidth ≤ 80 Mbps, which may be insufficient for vehicles to stream LiDAR point clouds to the edge server.

In traditional network communication, retransmission [52] and redundancy [6] are usually used to cope with lost or late packets. However, retransmission and redundancy are not ideal solutions in LiDAR point cloud streaming for the following reasons: (i) CAVs need real-time scene understanding, and retransmission incurs higher latency, while (ii) transmitting point clouds already consume non-trivial bandwidth, and redundancy

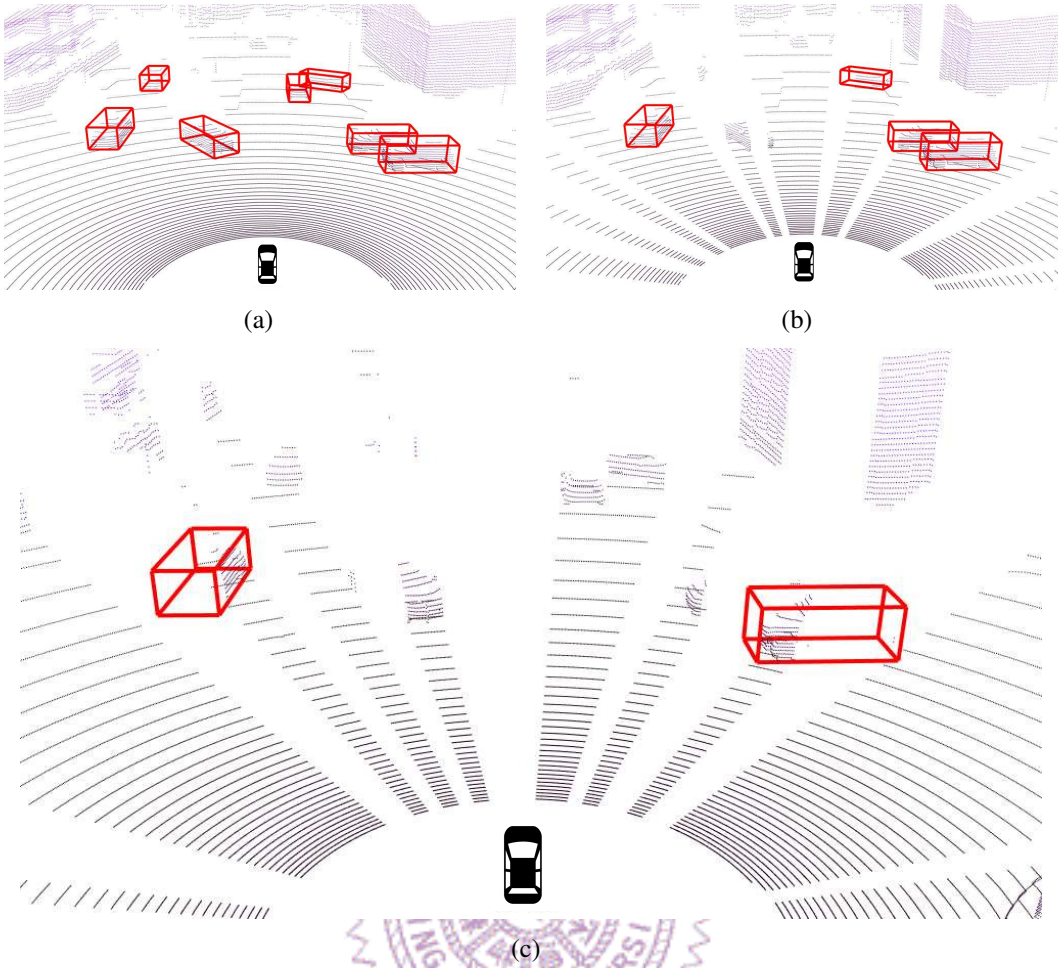


Figure 1.1: The results of object detection with different packet loss rates: (a) 0%, (b) 25%, and (c) 33%. Each bounding box represents a detected vehicle.

takes up even more bandwidth, which may lead to network congestion and cause safety concerns.

To partially cope with this issue, researchers have proposed to: (i) exchange high-level features [7, 12, 20, 49] extracted from 3D *points* or (ii) selectively transmit points [82] or features [21, 37, 65, 66] at reduced temporal or spatial resolutions. These studies assumed that wireless networks among vehicles are reliable for data streaming, which is *unrealistic*, as wireless networks are vulnerable to fading, shadowing, and interference, which often results in bursty packet loss. To understand the negative impacts of packet loss, we conducted a small-scale simulation of an intersection with eight vehicles using the CARLA (Car Learning to Act) [25] simulator. We placed a LiDAR on a vehicle at the center, which streams dynamic point clouds to an edge server over a DSRC network. The edge server runs an object detection algorithm [60] on the received point clouds. Fig. 1.1 shows the detected vehicles under different packet loss rates emulated by the Gilbert-Elliot model, which reveals that only 71.43% and 42.85% of vehicles are detected when one-fourth and one-third of packets are lost. To maintain high accuracy of scene under-

standing, affected point clouds must be *concealed* before being sent to corresponding ML algorithms.

In this thesis, we study the error concealment problem of dynamic LiDAR point clouds streamed from a CAV to a nearby roadside edge server. We do not advocate sharing high-privacy sensor data (such as RGB images, etc.), because this will cause the risk of privacy leakage. Given that a wide spectrum of classification tasks may need to be performed at edge servers to enable driving automations, we opt to exchange 3D point clouds from LiDAR, rather than extracted features that only work for pre-determined classification tasks. Therefore, our problem is more general, compared to projects exchanging high-level features [7, 12, 20, 49], as diverse and on-demand classification tasks can be performed on edge servers. Furthermore, selective point cloud [82] or feature [21, 37, 65, 66] exchanges are orthogonal to our work. *To the best of our knowledge, the considered research problem has never been studied in the literature.*

Typical rotating LiDARs horizontally divide each 360° point cloud *frame* into multiple equal-size *sectors*. Each sector is then encapsulated in one or multiple packets before being streamed. During dynamic point cloud streaming, packet loss results in *lost*[1] sectors and *incomplete* frames. Concealing incomplete frames is no easy task for several reasons: (i) vehicles are moving, which complicates the transformation between *relative* and *world* coordinates, (ii) some incomplete point cloud frames may contain too many lost sectors, making *spatial interpolation* less effective, if possible at all, and (iii) point cloud frames meant to be the inputs to *temporal interpolation* could contain too many lost sectors, degrading the performance of such interpolation. **Because of these challenges, existing solutions on *caching* [34, 58], *completion* [71, 78], and *interpolation* [10, 47, 84] cannot directly solve our error concealment problem. In particular, caching solutions [34, 58] did not consider moving LiDARs and always returned previously received point cloud frames as-is. Completion solutions [71, 78] focused on upsampling sparse point clouds, rather than fixing points in lost sectors. Moreover, the completion solutions employ some semantic labels that are not available in our considered problem. Last, interpolation solutions [10, 47, 84] assumed that the previous and next frames are *complete*, which may not always be the case.**

To solve the above three challenges, we design, implement, and evaluate a LiDAR Error Concealment (LEC) algorithm by fusing three error concealment approaches: (i) *temporal prediction*, which takes the previous frame as input to conceal the current frame, (ii) *spatial interpolation*, which takes the received sectors of the current frame as input to conceal the lost sectors in the same frame, and (iii) *temporal interpolation*, which takes

[1] Since the classification tasks are real-time, late sectors are of no use and are considered lost throughout this thesis.

the previous and next frames as inputs to conceal the current frame. Out of LiDAR, our system is compatible with other sensors, which are also commonly used in CAVs [11, 80]. For example, Mmwave (Millimeter-wave) radar can generate a 2D point cloud in a short distance, and the RGBD camera can capture the surrounding depth information to build a 3D environment map. *Our error concealment algorithm can be quickly adapted to other sensors.*

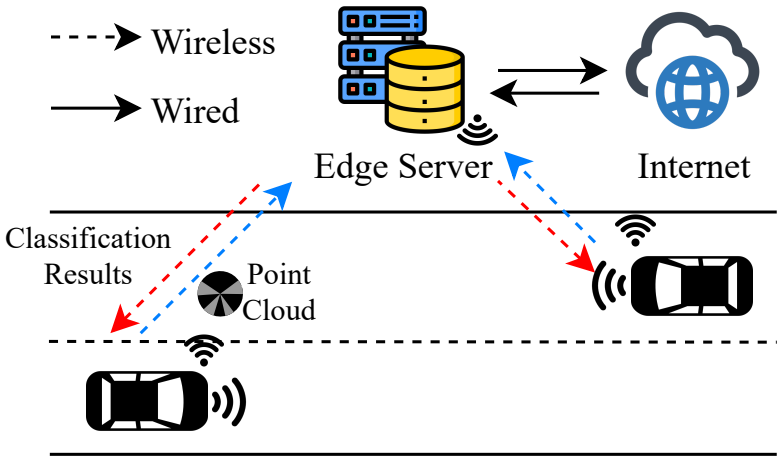


Figure 1.2: Sampled usage scenarios of our system.

We explain our system using the simple scenario shown in Fig. 1.2, consisting of two vehicles on the road, both of which are equipped with LiDAR to generate dynamic point clouds. We install an edge server on the side of the road to receive the point clouds sent by CAVs based on wireless networks. CAVs will upload the point cloud to the nearby edge server, and one edge server will receive from multiple vehicles at the same time. The edge server will store the point cloud locally from each packet and upload the point clouds to the internet for persistent storage when the network load is low, which is beneficial for traffic accident investigation and traffic analysis. In addition, the edge server will perform classification tasks on the received point clouds to generate classification results and broadcast them to nearby CAVs.

1.1 Contributions

Our LEC algorithm adaptively employs Temporal Prediction (TP), Spatial Interpolation (SI), or Temporal Interpolation (TI) approaches to minimize Chamfer distance. We make the following contributions:

- We study the error concealment problem of dynamic LiDAR point cloud streaming, which to the best of our knowledge has never been investigated.

- We propose a LiDAR Error Concealment (LEC) algorithm to adaptively select the most promising error concealment approach using an ML model.
- We evaluate our error concealment algorithm using both a pre-recorded real dataset and an autonomous-driving and networking co-simulator. Our simulation results reveal the merits of our proposed algorithm, which reduces the Chamfer distance by up to 82.68%, and achieves a small optimal gap of 0.75% on object detection accuracy.

1.2 Limitations

As the very first work on error concealing dynamic LiDAR point clouds, this thesis makes the following assumptions:

- The localization of CAVs is reliable, backed up by recent research on localization technologies [22, 38], which achieve traffic lane-level accuracy.
- The system clocks on CAVs and edge servers are synchronized. Various time synchronization protocols, such as NTP (Network Time Protocol) [50], PTP (Precision Time Protocol) [23], and TSP (Time Stamp Protocol) [2] can be used.

1.3 Organization

In this thesis, we first introduce the motivation of our work in Chapter 1. Chapter 2 summarizes the background knowledge of our work, including Connected and Autonomous Vehicles (CAVs), driving automation datasets, machine learning for scene understanding, and error concealment. Then, we present related work on streaming LiDAR point cloud error concealment in Chapter 3. Chapter 4 proposes a streaming system for dynamic LiDAR point clouds and formulates the research problem. In Chapter 5, we describe alternative algorithms and our TLEC and LEC algorithms. Finally, we introduce our co-simulator and evaluation setup to demonstrate the superiority of our LEC algorithm in Chapter 6. Chapter 7 concludes the thesis.

Chapter 2

Background

This chapter provides background knowledge of the thesis work.

2.1 Connected and Autonomous Vehicles (CAVs)

In recent years, driving automation has become one of the inventions of the new age. The Society of Automotive Engineers (SAE) proposed 5 levels for autonomous driving vehicles [61]: (i) L1: the vehicle performs a single operation on the control system, and the driver is still responsible for the rest of the driving actions, (ii) L2: the vehicle performs multiple operations on the control system, but the driver still needs to be responsible for the rest of the driving actions, (iii) L3: the vehicle completes most of the driving operations, but the driver needs to keep observing to take over the vehicle, (iv) L4: the vehicle can achieve fully automatic driving in specific scenarios, and the driver does not need to attend, (v) L5: the vehicle can complete fully automatic driving in all scenarios.

At present, the most popular autonomous driving vehicle Tesla still at the level of L3 and cannot achieve fully autonomous driving, Tesla is still at still depends on its sensors, and due to cost limitations, the field of view is often very limited. Some traffic accidents caused by autonomous driving have caused controversy [8, 17]. When the sensor is blocked, it may not work properly. In addition, some sensors cannot see objects that are too far away, objects further away become blurred, and extreme weather can degrade sensor performance. To clear these limitations, some researchers have proposed *cooperative perception* [18, 36], which uses wireless networks to share perception messages to expand the field that can effectively avoid accidents and traffic congestion. Researchers propose different types of perception messages, which can be defined into three levels: (i) sensor level [13, 19, 41]: sharing the sensor data and analyzing the received data with self-sensor data, which has the highest amount of information and the highest network bandwidth required, (ii) feature level [12, 49]: share features of sensor data with medium

required network bandwidth, but the amount of information will be compressed, and object level [20, 66]: an advanced data type that only shares the final result of the analysis, but may share incorrect results due to vision occlusions, and the receiver cannot perform secondary analysis, which is heavily dependent on the sender's analysis capabilities.

2.2 Driving Automation Datasets

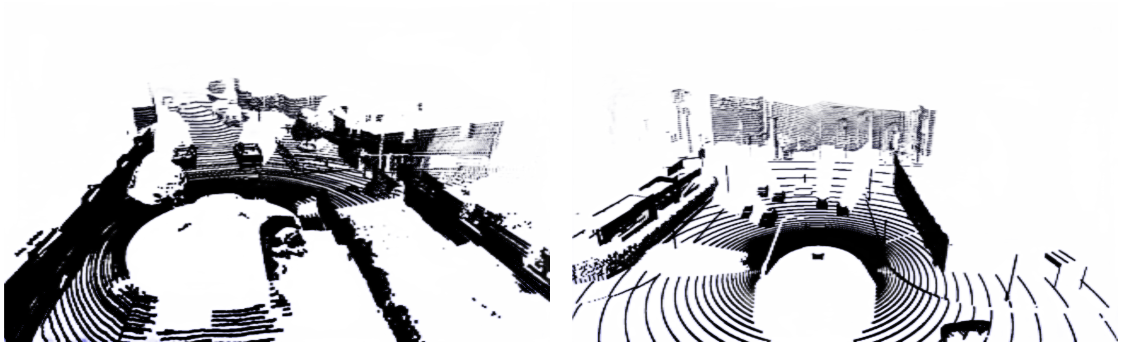
Datasets are very important in machine learning and computer vision. Datasets can be used to train and evaluate models to demonstrate the superiority of their algorithms. For driving automation, researchers have proposed some datasets as benchmarks, such as Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [30], Semantic3D [33], etc.

KITTI provides several datasets for optical flow, visual odometry, 3D object detection, and tracking. These data are collected on actual roads, including urban, rural, and highways. For each acquisition vehicle, use (i) 2 $\times$ PointGray Flea2 grayscale cameras (FL2-14S3M-C), (ii) 2 $\times$ PointGray Flea2 color cameras (FL2-14S3C-C), (iii) 4 $\times$ Edmund Optics lenses, (iv) 1 $\times$ Velodyne HDL-64E rotating 3D laser scanner, and (v) 1 $\times$ OXTS RT3003 inertial and GPS navigation system to collect *gray images*, *RGB images*, *3d point clouds*, and *localization information*. Annotators label 3D point cloud objects as `Car`, `Van`, `Truck`, `Pedestrian`, `Person Sitting`, `Cyclist`, `Tram`, `Misc`, and `Dontcare`. If the object cannot be identified due to occlusion or distance, it will be marked as `Dontcare`. KITTI also provides a semantic segmentation dataset as Semantic-KITTI [9], which provides continuous point cloud frames, and each point is labeled into 28 types such as `Car`, `Truck`, `Motorcycle`, `Pedestrian`, and `Bicyclist`, etc. It is worth noting that each object has a unique identification code, even if the object appears in different frames.

Although many large-scale benchmark datasets are available, they are pre-recorded datasets that are not conducive to real-time analysis and complex scenarios. To fix this limitation, researchers have developed several automated driving simulators based on virtualization technology [5, 48, 54, 75, 76] to generate scalable datasets in real-time.

2.3 Machine Learning for Scene Understanding

Machine learning is widely used in scene understanding. It can help driving automation systems to understand better and infer scenes, thus supporting the development of object detection, semantic segmentation, shape classification, lane detection, object tracking, and other technology. Object detection focuses on finding all objects of interest in an



(a) (b)

Figure 2.1: Sample point clouds from: (a) the KITTI Dataset, (b) the CARLA simulator.

image, which requires not only determining the location of the object but also accurately identifying the type of object. In 3d point cloud object detection, the detected objects are marked by a 3d bounding box, and the points in the box are considered to belong to this object. Intersection over Union (IoU) is commonly used to determine the accuracy of object detection algorithms. It defines the overlap rate of two boxes, i.e. the ground truth box and the predicted box. When the IoU is greater than the threshold, the object is considered to be successfully detected. Object detection provides coarse object-level results. To obtain finer perceptual results, semantic segmentation is used to classify each point in the point cloud into different semantic categories, such as buildings, roads, vehicles, pedestrians, etc.

Several deep learning techniques are often used for scene understanding, such as CNN, RCNN, FastRCNN, and FasterRCNN. In general, all of these techniques can extract features from the original images for classification tasks, but all have some limitations. CNN is widely used for image classification and detection, but may not perform well for objects in irregular regions. To address this limitation, RCNN first partitions the image into multiple RoI (Regions of Interest) and normalizes them to meet the requirements of CNN, and then performs feature extraction for each region. However, the image is segmented into multiple regions for feature extraction, which means more running time. Fast RCNN only needs to perform feature extraction once to get all the RoI features, but it still requires more runtime based on selective search. Faster RCNN uses RPN (Region proposal network) to select RoI, which greatly reduces the runtime.

2.4 Error Concealment

Error concealment [3, 4, 35] was designed to reduce visual and auditory data degradation caused by lost or corrupted data. It is widely used in video meetings, streaming media, mobile communication, and broadcasting to provide a better user experience. When

data is not received in its entirety due to channel errors, packet loss, or being late, error concealment can be used to recover as much of the original data as possible. In video transmission, data is usually transmitted as packets. When packets cannot be received, these four methods are usually used: (i) *duplication*: filling the lost packets with adjacent packets, applicable to packets with similar content, (ii) *interpolation*: estimating lost packets based on the received packets by interpolation algorithms, such as linear, polynomial, and spline interpolation, etc. (iii) *motion compensation*: estimating the motion vectors of objects in the video, using adjacent frames and motion vectors to generate lost frames, and (iv) *reconstruction*: using the received frames to extract features to build a model, which requires more sophisticated algorithms.

Chapter 3

Related Work

Although the considered error concealment problem has never been studied in the literature, prior works on point cloud caching, completion, and interpolation inspired the development of our temporal prediction, spatial interpolation, and temporal interpolation approaches. In this chapter, We survey prior works on point cloud caching, completion, and interpolation, which inspired the development of our error concealment algorithm.

3.1 Cooperative Perception

Chen et al. [13] proposed a V2V cooperative perception system that transmits point clouds to improve the accuracy of object detection. Sample point clouds reduce the amount of network transmission, and fuse point clouds from two vehicles are used for object detection to avoid omissions. Zhang et al. [82] proposed a cooperative perception system with edge servers. The edge server considers bandwidth to split multiple regions for vehicle transmission, and vehicles upload non-overlapping and compressed point clouds for global view object detection. Qiu et al. [57] broadcast the points of objects and their motion vector, and the receiver will reconstruct and fuse objects. Use the 3D map generated based on crowdsourcing to confirm the relative position between vehicles, extract the points of objects based on pixel features and use optical flow to extract the motion vectors. Arnold et al. [7] proposed a cooperative system with multiple message types that adaptively selects different message types based on the distance of points. A central server is used to receive and fuse messages from edge servers to perform object detection and broadcast to nearby vehicles. The system does not consider communication loss. ETSI (European Telecommunications Standard Institute) [64] proposes the first rules of message generation, which define when the vehicle should send the message and what the content should contain. In addition, some researchers [21, 65, 66] have improved these rules.

3.2 Point Cloud Caching

Han et al. [34] proposed a point cloud caching mechanism based on Long Short-Term Memory (LSTM) for 3D object detection. They used LSTM to accumulate features from previous sectors at the same angle. The received LiDAR point clouds are enriched with the accumulated features. Qu et al. [58] built a point cloud streaming system to reduce the latency of LiDAR scans. They first pushed sectors into a buffer immediately after being received and then invoked *Iterative Closest Point (ICP)* algorithms to generate a new frame. By doing so, a point cloud frame can be generated before receiving all of its sectors. Frossard et al. [28] proposed to generate voxels from LiDAR point clouds, and applied a convolutional backbone network to process the relevant sectors. They read and updated point clouds using scalable spatial memory, and a hash function for spatial location matching. Chen et al. [14] employed wedge-shaped point cloud sectors and polar coordinates for point cloud caching. More precisely, they used polar columns to encode a point cloud sector and represent it in a wedge shape for lower time and space complexity. They then extracted features from these wedge-shaped sectors. *These point cloud caching papers [14, 28, 34, 58] inspired us to develop the temporal prediction concealment approach.*

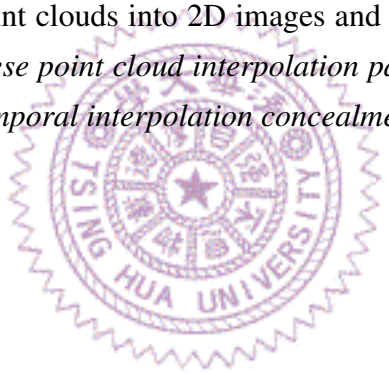
3.3 Point Cloud Completion

Wen et al. [71] proposed to fuse the features of known and missing points using the skip-attention mechanism. In particular, sparse point clouds were analyzed using PointNet++ [55] for feature extraction, while the resulting features were used to complete a point cloud frame without affecting the multi-resolution structure. Yu et al. [81] presented a completion solution using set conversion. More specifically, they applied transformer networks and the *K-Nearest-Neighbor (KNN)* [53] algorithm on geometric relations. They next employed FoldingNet [79] for restoring sparse point clouds. Yan et al. [78] gave a two-stage neural network, which generates complete point clouds from sparse frames. Their solution was built upon hierarchical graph and refinement algorithms to generate the point cloud in the sparse sectors. *These point cloud completion papers [71, 78, 81] inspired us to develop the spatial interpolation concealment approach.*

3.4 Point Cloud Interpolation

Zhao et al. [85] presented a low-complexity frame interpolation framework based on range images and optical flows. In particular, they first projected two frames into range images

and then performed optical flow extraction [43]. The extracted features were used for intermediate frame prediction. Lu et al. [47] showed a framework based on *3D scene flows*. Particularly, they applied FlowNet3D [45] to get 3D scene flows and fuse two consecutive frames into the intermediate one. Xu et al. [74] applied hierarchical warping to generate the intermediate frame using two adjacent frames. More precisely, they adopted self-attention [69] to reconstruct spatial information for minimum chamfer distance. Zhao et al. [84] found that the existing point cloud interpolation studies heavily relied on 2D optical and 3D scene flows, which are both resource hungry. To cope with this issue, they: (i) projected two point cloud frames into a 2D depth image, (ii) performed feature extraction and aggregation, and (iii) re-projected the features back to the intermediate frame. Xu et al. [73] developed a self-supervised interpolation neural network by combining FlowNet3D [45] for computing 3D scene flows and KNN for extracting local features to predict the intermediate frame. Zheng et al. [10] applied the techniques of interpolating video frames for point cloud interpolation. In particular, they adopted spherical projection to convert point clouds into 2D images and convolution to improve point cloud feature extraction. *These point cloud interpolation papers [10, 47, 73, 74, 84, 85] inspired us to develop the temporal interpolation concealment approach.*



Chapter 4

LiDAR Point Clouds Streaming

This chapter first introduces the characteristics of point clouds and the measurement principles of LiDAR, which helps the reader to understand our error concealment algorithm. Then, we introduce our dynamic LiDAR point cloud streaming system and formulate our research problem.

4.1 Properties of LiDAR Point Clouds

Point clouds are a common representation in driving automation and have the following characteristics:

- **High dimensionality**: A set of points in 3D space, and each point has three coordinates, which are high-dimensional data.
- **Unordered**: The points are not in order, and modifying the order will not affect the result.
- **Interaction between points**: Unlike another image, a single point is meaningless, and the features need to consider its structure and context.
- **Invariance under transformations**: For points in the point cloud, their absolute position does not matter, and the overall rotation, transformation, and scaling does not modify the structure.

In addition, compared to dense point clouds [44], LiDAR point clouds tend to be sparser, have lower framerates, and have fewer points (see Fig. 4.1).

(a) (b)

Figure 4.1: 3D representations of (a) dense point clouds and (b) LiDAR point clouds.

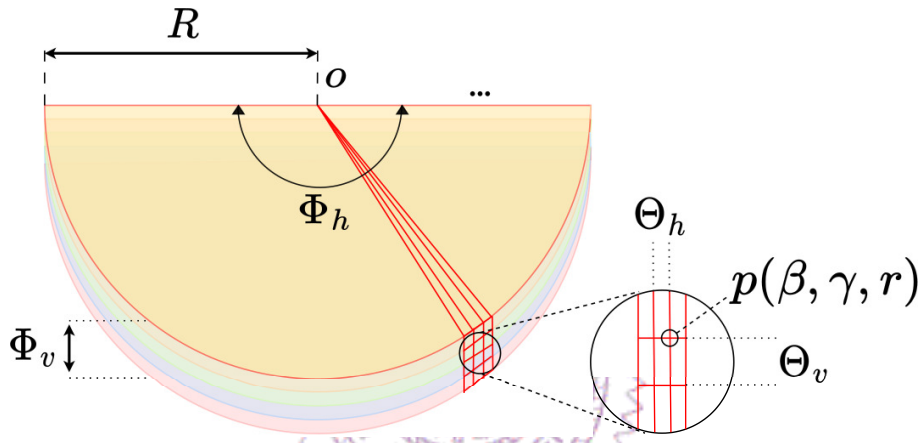


Figure 4.2: Sampled measurements of a typical LiDAR.

4.2 Measurements of LiDARs

We use o to denote the coordinates of a LiDAR center, which is $(0, 0, 0)$ in relative coordinates shown in Fig. 4.2. Let R be the range of LiDAR laser beams for measuring distance. Whenever a laser beam is not bounced back, the LiDAR either: (i) records infinity or (ii) skips that measurement, which should be properly interpreted by the ML classification algorithms. Without loss of generality, we assume that those measurements are skipped. LiDARs employ uniform sampling resolutions[1]: a distance is measured every Θ_h and Θ_v degrees horizontally and vertically. Moreover, a LiDAR covers horizontal and vertical ranges of Φ_h and Φ_v, usually $0^\circ < \Phi_h \leq 360^\circ$ and $0^\circ < \Phi_v \leq 180^\circ$. For better manageability, the measured points are horizontally grouped into sectors with a fixed center angle of Ψ. That is, a frame f_i, where $i \in \mathbb{Z}^+$, contains sectors $s_{i,j}$, where $j = 1, 2, \ldots, \Phi_h/\Psi$. For simplicity, we assume divisions are without remainders. Here, each sector $s_{i,j}$ has up to $(\Psi/\Theta_h)(\Phi_v/\Theta_v)$ measured points. For each measured point p, we define its *pitch* $p.\beta$

[1]We consider such LiDARs for the sake of presentation, while our derivation can be readily generalized to non-uniform sampled LiDARs.

and *yaw* $p.\gamma$ as the angles along the horizontal and vertical axes, and *distance* $p.r$ as the distance to the reflection point. We can transform $p.\beta$, $p.\gamma$, and $p.r$ into $p.x$, $p.y$, and $p.z$ by:

$$p.\beta = arctan(\frac{z}{\sqrt{x^2+y^2}}); \tag{4.1a}$$

$$p.\gamma = atan(\frac{y}{x}); \tag{4.1b}$$

$$p.r = e(p, o); \tag{4.1c}$$

$$p.x = p.r \times cos(p.\beta) \times cos(p.\gamma); \tag{4.1d}$$

$$p.y = p.r \times cos(p.\beta) \times sin(p.\gamma); \tag{4.1e}$$

$$p.z = p.r \times sin(p.\beta). \tag{4.1f}$$

4.3 System Overview

We consider a point cloud streaming system as shown in Fig. 4.3. CAVs are equipped with LiDARs for capturing point clouds, as well as GPS (Global Positioning System) and IMU (Inertial Measurement Unit) for localization. The LiDAR point clouds are sent to a ground removal module to filter out irrelevant and redundant points. The remaining points along with vehicle locations are encoded and sent to the edge server. At the edge server, the point clouds are received and decoded before being pushed into the buffer. The buffered point clouds are used for error concealment at a later time. We propose to include an *error concealment* module to conceal the incomplete frames due to packet loss, which to the best of our knowledge has never been done before. A merger module combines point clouds from multiple vehicles for global views before sending them to a classifier module. The classification results are saved in the *storage*, before being sent back to CAVs whenever the network conditions are better. Note that the classification results can be broadcast and shared by nearby CAVs, although the optimal strategy of broadcasting is beyond the scope of this thesis. Upon receiving the classification results, CAVs pass them to controller modules for driving automations.

In this thesis, we focus on designing, implementing, and evaluating the error concealment module for CAVs. Other modules in the system can be implemented using existing solutions, e.g.,

- *Ground removal:* LiDAR scanning objects will generate many redundant ground points, which are not conducive to the analysis and will occupy a large amount of bandwidth. Algorithms like RANSAC (Random Sample Consensus) can be used [27], which iteratively fits a model to sampled point clouds and then removes the points with high fitting errors.

- *Encoder and decoder:* Multiple point cloud compression algorithms [72], such as Draco [32], can be employed. Draco is Google's open-source 3D graphics compression library, which is widely used in the Chrome browser.

- *Merger:* Point clouds from a single vehicle often provide very limited views, and merging point clouds from multiple vehicles can help eliminate visual blind spots. Previous work [82] has demonstrated that merging point clouds from multiple vehicles can improve object detection accuracy by about 40%. Various point cloud merging algorithms, such as Zhang et al. [82] can be used.

- *Classifier:* Driving automation requires real-time classification tasks to identify objects, which can help vehicles predict the behavior of other traffic participants and take appropriate measures to ensure safe driving. Various classification tasks can be performed, such as object detection algorithms [60]. Notably, the classification tasks in CAV are not limited to object detection but can be extended to more complex classification tasks, such as semantic segmentation, lane detection, and object tracking. The goal of these tasks is to help CAVs better understand their surroundings and make accurate decisions.

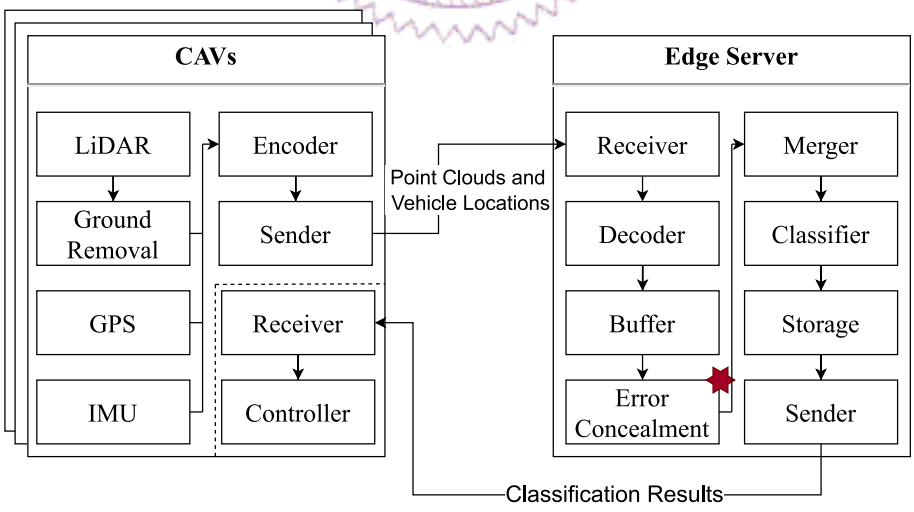


Figure 4.3: The considered point cloud streaming system.

In our evaluations, we adopted the abovementioned solutions. Our error concealment modules can be added to other cooperative perception solutions as extensions to improve system robustness and increase the maximum number of vehicles as the most advanced cooperative perception solution [82] still only supports 6 vehicles for sensor data sharing.

4.4 Network Protocols

We define an application-layer protocol to facilitate communications between the CAVs and edge servers. There are three key message types.

- *Location Update (LU)*: reports the latest LiDAR center location from GPS and IMU. LU messages are sent by a CAV when: (i) the location difference exceeds $\triangle_d$ or (ii) the last update message has an age beyond $\triangle_t$. Both $\triangle_d$ and $\triangle_t$ are system parameters.
- *Point Update (PU)*: contains the point clouds of a sector, which is sent once the points in that sector are encoded.
- *Classification Result (CR)*: contains the classification outcomes produced at the edge server. Whenever the edge server performs a classification task, it broadcasts CR messages to nearby CAVs

Sample message exchanges are given in Fig. 4.4. Notice that we focus on concealing errors between a single vehicle and its edge server. Hence, in the rest of the thesis, we mainly concentrate on the PU messages, which dominate the network traffic amount.

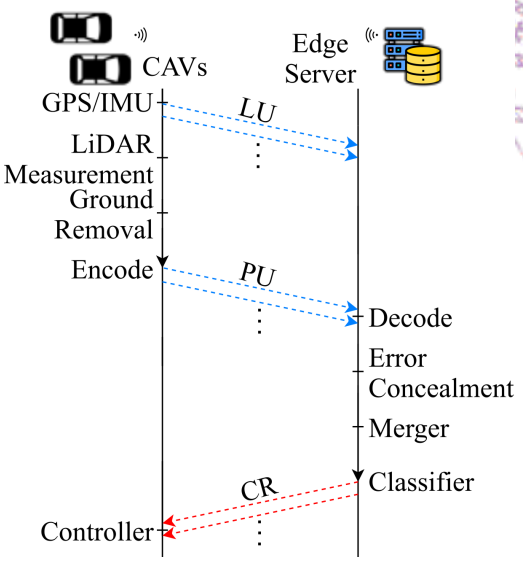


Figure 4.4: A sample point cloud streaming session.

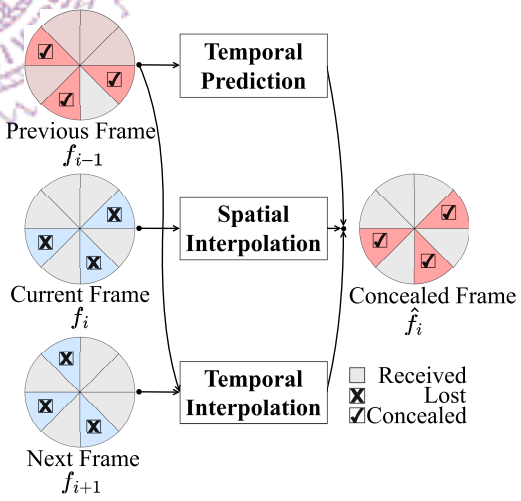


Figure 4.5: The input/output of our error concealment algorithm.

4.5 Problem Statement

The sectors of LiDAR point clouds are streamed in order from CAVs to an edge server, where points in sectors are fused for ML-based analytics to enable driving automation.

Each LiDAR mounted on a CAV generates a sequence of PU messages, where each message carries all points within a sector. Here, every point p is represented in coordinates: $p.x$, $p.y$, and $p.z$. Whenever the edge server detects an incomplete frame f_i due to packet loss, it performs error concealment using frames: f_{i-1}, f_i, and f_{i+1}. Since the edge server conceals lost packets sequentially, all the sectors $s_{i-1,j}$ in the previous frame f_{i-1} are either received or concealed, while some of the sector $s_{i+1,j}$ of the next frame f_{i+1} may be empty. We use frames as the unit of error concealment and conceal lost sectors sequentially. The LiDAR speed is usually 10~20 HZ, which means that the interval between two frames is only 50~100 ms. We let n_i be the number of points in frame f_i. The distortion of a concealed frame $\hat{f}_i$ can be written as the Chamfer distance [72] to the ground truth frame f_i, which is:

$$d^C(\hat{f}_i, f_i) = \frac{\sum\limits_{p\in\hat{f}_i} \min\limits_{p'\in f_i} \|p-p'\|_2^2}{\hat{n}_i} + \frac{\sum\limits_{p\in f_i} \min\limits_{p'\in\hat{f}_i} \|p-p'\|_2^2}{n_i}. \tag{4.2}$$

Note that, if an error concealment algorithm generates a huge number of points, its Chamfer distance may be biased. More importantly, doing so may confuse ML classification algorithms. Hence, when a concealed sector has more than $(\Psi/\Theta_h)(\Phi_v/\Theta_v)$ points, we uniformly downsample it. The objective of our error concealment problem is to find the optimal $\hat{f}_i^* = \underset{\hat{f}_i}{argmin}\ d^C(\hat{f}_i, f_i)$ to minimize the Chamfer distance.

Chapter 5

Error Concealment of LiDAR Point Clouds

In this chapter, we first list all alternative algorithms and categorize them into Temporal Prediction (TP), Spatial Interpolation (SI), and Temporal Interpolation (TI). Next, we propose the Threshold-based LiDAR Error Concealment (TLEC) and LiDAR Error Concealment (LEC) algorithms to adaptively select the most promising algorithms among the TP, SI, and TI alternatives.

5.1 Overview

Table 5.1 gives the symbols used throughout the thesis. Since ML classification tasks for CAVs are real-time, our error concealment algorithm on f_i must work under diverse sector availability of the current and adjacent frames f_{i-1} and f_{i+1}. As illustrated in Fig. 4.5, the core idea of our algorithms is to adaptively apply one of the three concealment approaches: (i) Temporal Prediction (TP), which only needs frame f_{i-1} and thus incurs the least latency; (ii) Spatial Interpolation (SI), which only needs (incomplete) frame f_i and thus has the highest applicability; and (iii) Temporal Interpolation (TI), which takes frames f_{i-1} and f_{i+1} as input for the richest information. We present alternative TP, SI, and TI algorithms in the following.

5.2 Temporal Prediction: TP

Since the edge server conceals lost packets sequentially, all the sectors $s_{i-1,j}$ in the previous frame f_{i-1} are either received or concealed. TP selectively *copies* points from sectors of f_{i-1} to conceal the lost sectors of f_i. Although the inter-frame time is only 50–100 ms, the movement of a CAV must be taken into account. We consider the following alternative

Table 5.1: Symbols Used in This Thesis

Symbol	Description
o	Coordinates of a LiDAR center
R	LiDAR maximum scan distance
Θ_h	Measure resolution in horizontal
Θ_v	Measure resolution in vertical
Φ_h	Measure degrees in horizontal
Φ_v	Measure degrees in vertical
f_i	A point cloud frame
$s_{i,j}$	A sector in point cloud frame f_i
Ψ	Angle of each sector
p	A point in point cloud frame
$p.\beta$	Pitch angle of the point
$p.\gamma$	Yaw angle of the point
$p.r$	Distance of the point
$p.m$	Motion vector between p' to p''
n_i	Number of points in frame f_i
$\hat{f}_i$	Concealed frame of f_i
$\hat{s}_{i,j}$	Concealed sector of s_i
M_i	Transformation matrix from f_{i-1} to f_i
M'_i	Transformation matrix from f_{i-1} to f_{i+1}
M''_i	Transformation matrix from f_{i-1} to f_i
w_1, w_2, w_3	Least square parameters
τ_{i}	Packet loss rate in frame f_i
T_p	TLEC algorithm threshold
T_n	TLEC algorithm threshold

algorithms:

- *Copyover Prediction (CP):* Directly copying the sectors of f_{i-1} does not require powerful algorithms and computing resources, and is suitable for real-time applications. We let $\hat{s}_{i,j} = s_{i-1,j}$, for any lost sector $s_{i,j}$.

- *Motion-compensated Prediction (MP):* We consider the location/orientation difference between LiDARs in f_{i-1} and f_i. Let M_i be the transformation matrix from f_{i-1} to f_i; we give $\hat{s}_{i,j} = s_{i-1,j}M_i$, for any lost sector $s_{i,j}$.

5.3 Spatial Interpolation: SI

SI employs the points in the current frame to *estimate* the measured distance $p.r$ for every given pitch $p.\beta$ and yaw $p.\gamma$. More specifically, we repeat the estimation for $(\Psi/\Theta_h)(\Phi_v/\Theta_v)$ points in each sector, where $p.\beta$ and $p.\gamma$ are well defined by the LiDAR's specifications (see grids in Fig. 4.2). For p with $p.r > R$, we skip it in the concealed frame $\hat{f}_i$. We consider the following alternative algorithms:

- *Nearest Neighbor (NN):* For point clouds, neighboring points often have the same features. For each point p of lost sectors in frame f_i, we find the closest point p^* from all received sectors in f_i. We let $p.r = p^*.r$.

- *Least Square (LS):* Least Square is a common linear valuation method that can be used to find the unknown distance of lost sectors. We fit all received points in frame f_i to $p.r = w_1 p.\beta + w_2 p.\gamma + w_3$, for parameters w_1, w_2, and w_3. We then use this equation to estimate $p.r$ for all points in lost sectors of f_i.

5.4 Temporal Interpolation: TI

TI analyzes frames f_{i-1} and f_{i+1} to locate every point p' in f_{i-1} and p'' in f_{i+1}, and uses every pair of p' and p'' to create a point in the lost sectors of f_i. Using p' and p'', we then calculate the motion vector $p.m$ as $p'' - p'$. Last, TI *interpolates* the concealed frame as $\hat{p} = p' + p.m/2$. To compute the motion vectors, we consider the following algorithms:

- *Point Matching (PM):* For each point p' in f_{i-1}, we find the closest point p'' in frame f_{i+1} in the Euclidean distance. Each pair of p' and p'' is used to estimate a point $\hat{p}$ in the concealed frame f_i.

- *Iterative Closest Point (ICP):* ICP is a well-known 3D point cloud registration algorithm, which can be used to compute a transform matrix from f_{i-1} to f_{i+1}, denoted as M_i'. Let M_i'' be the transformation matrix that shifts/rotates half of the displacement/angles of M_i'. Any lost sector $s_{i,j}$ can be concealed by $\hat{s}_{i,j} = s_{i-1,j} M_i''$

- *Scene Flow (SF):* FlowNet3d [45] is a popular scene flow algorithm to compute motion vectors of individual points in f_{i-1} to f_{i+1}. Upon the derived motion vectors, points in concealed frame $\hat{f}_i$ can be computed.

- *Bidirectional Scene Flow (BSF):* PointINet [47] computes scene flows in both directions (from f_{i-1} to f_{i+1} and from f_{i+1} to f_{i-1}), and fuses the two temporally interpolated frames into $\hat{f}_i$.

5.5 Threshold-based LiDAR Error Concealment Algorithm: TLEC

Intuitively, the selection of concealment approaches depends on the *incomplete ratio* $\tau_{\text{i-1}}$, τ_{i}, and $\tau_{\text{i+1}}$ of frames f_{i-1}, f_i, and f_{i+1}, which are the fractions of lost sectors. Both TP and TI take f_{i-1} as their input, which may limit their performance if $\tau_{\text{i-1}}$ is too high.

Similarly, TI may not perform well if $\tau_{\text{i+1}}$ is too high, while SI may perform better if τ_{i} is too low. Combining the above design rationales, we propose the *Threshold-based LiDAR Error Concealment (TLEC)* algorithm in Algorithm 1. The TLEC algorithm works as follows: when $\tau_{\text{i-1}}$ is greater than or equal to a threshold T_p, we execute SI. Next, if $\tau_{\text{i+1}}$ is less than another threshold T_n, we execute TI. We execute TP, otherwise. T_p and T_n are both system parameters. We vary thresholds $T_p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $T_n \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$ to find the best thresholds. To determine the best threshold T_p^* and T_n^* in C-V2X networks we perform a grid search on 25 combinations in one of our pilot simulations. We plot sample results from 3-vehicle simulations in Fig. 5.1, where the markers annotate the selected threshold. These results show that our TLEC algorithm leads to Chamfer and Hausdorff distances of 3.33 and 18.26 m, within a running time of 789 ms. According to the simulation results from 1-, 3-, 5-, and 7-vehicle, the best T_p^* values are: 0.6, 0.8, 0.6, and 0.6; and the best T_n^* values are: 0.5, 0.5, 0.4, and 0.6. We recommend $T_p^* = 0.6$ and $T_n^* = 0.5$ following majority votes. Nonetheless, the variance among T_p^* and T_n^* under different environments appears to be nontrivial. Hence, we conclude that the TLEC algorithm does not generalize to various and dynamic settings. In the rest of the thesis, we no longer consider the TLEC algorithm.

Algorithm 1 Threshold-based LiDAR Error Concealment (TLEC)

Input: Incomplete Ratios $\tau_{\text{i-1}}$, τ_{i}, $\tau_{\text{i+1}}$, and Point Cloud Frames f_{i-1}, f_i, f_{i+1}.

if $\tau_{\text{i-1}} \geq T_p$ **then** //f_{i-1} is seriously damaged
    Execute Spatial Interpolation (SI)
else if $\tau_{\text{i+1}} < T_n$ **then** //f_{i+1} is in a reasonable shape
    Execute Temporal Interpolation (TI)
else //f_{i+1} is seriously damaged
    Execute Temporal Prediction (TP)

Algorithm 2 LiDAR Error Concealment (LEC)

Input: Incomplete Ratios $\tau_{\text{i-1}}$, τ_{i}, $\tau_{\text{i+1}}$, and Point Cloud Frames f_{i-1}, f_i, f_{i+1}.

Let dm be the object of the decision model and P_i be the index of the promising error concealment algorithm.
P_i = dm.predict($\tau_{\text{i-1}}$, τ_{i}, $\tau_{\text{i+1}}$) ▷ Find the most promising concealment approach
if P_i == SPATIAL_INTERPOLATION **then**
    Execute Spatial Interpolation (SI)
else if P_i == TEMPORAL_INTERPOLATION **then**
    Execute Temporal Interpolation (TI)
else
    Execute Temporal Prediction (TP)

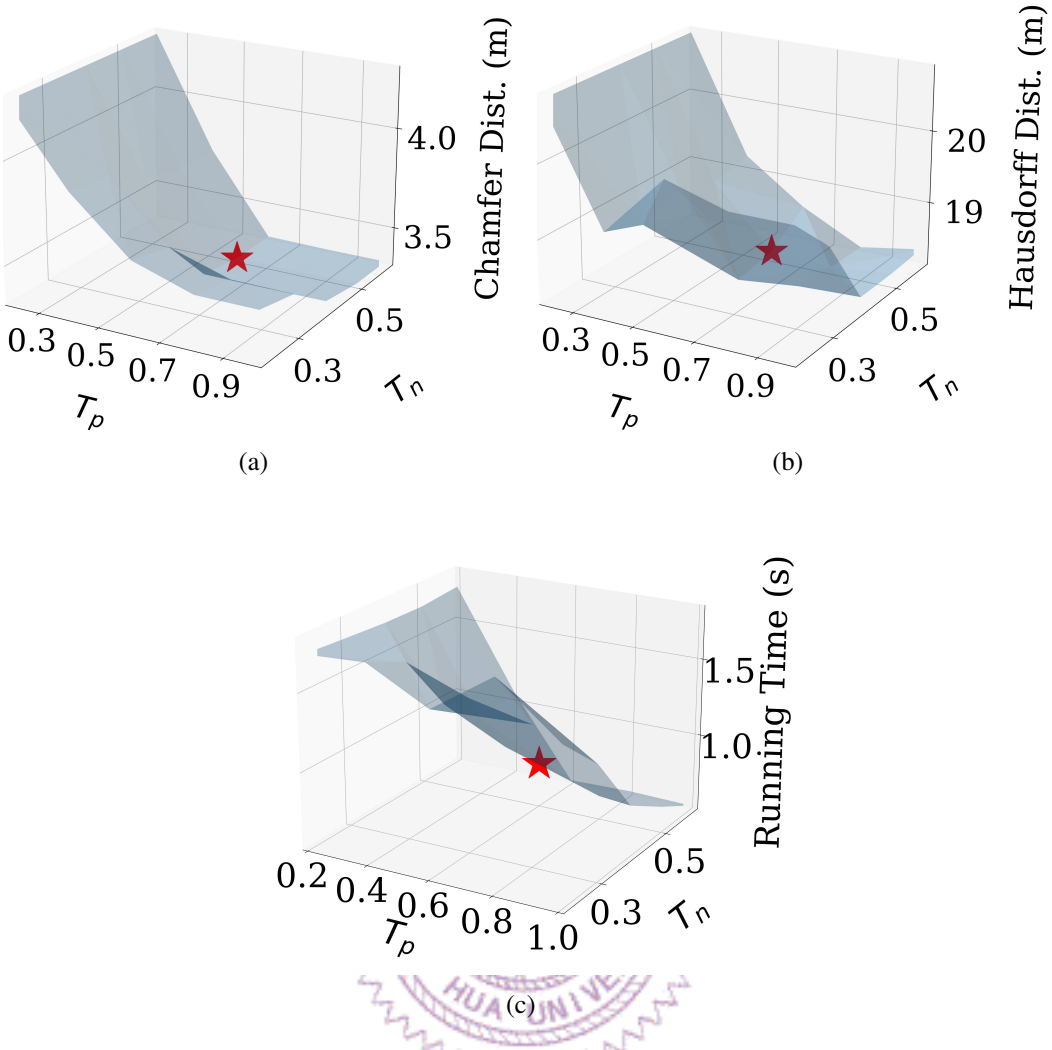


Figure 5.1: Simulation results from our early LEC algorithm in a C-V2X network. Sample performance results with 3 vehicles from difference threshold values: (a) Chamfer distance, (b) Hausdorff distance, (c) running time.

5.6 LiDAR Error Concealment Algorithm: LEC

The TLEC algorithm relies on two manually-selected thresholds and cannot perform well when nonlinear or high-dimensional decisions are needed. To address these limitations, we propose using ML algorithms, such as Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF) to adaptively choose the most promising concealment approach for minimum Chamfer distance. Among these ML algorithms, DT provides a highly interpretable and widely adaptable model through simple decision rules and feature segmentation. SVM can deal with nonlinear problems and classification tasks with strong robustness by finding the optimal hyperplane. By integrating the prediction results of multiple decision trees, RF has good generalization ability and is suitable for high-dimensional and large-scale data. As shown in Fig. 5.2, we use DT, SVM, or RF as the decision model to select the most promising concealment approach by incomplete ratio.

We give the LEC algorithm in Algorithm 2 and report the performance of DT, SVM, and RF under different hyperparameters in the next Chapter.

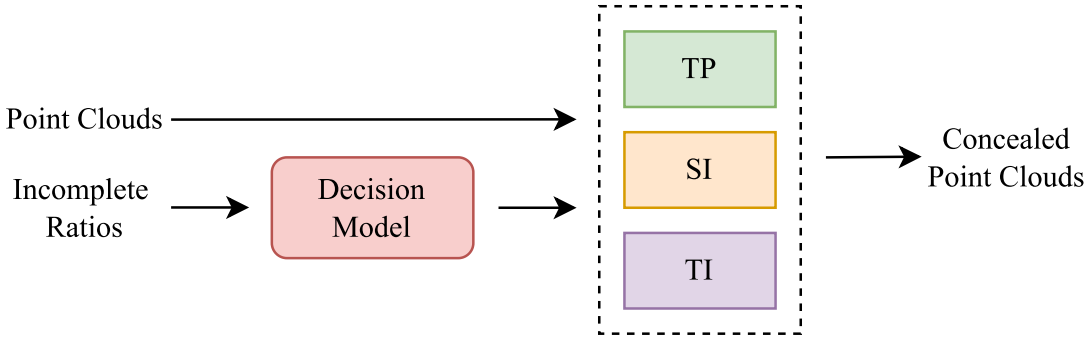


Figure 5.2: The overview of our LEC algorithm.

Chapter 6

Evaluations

In this chapter, we first present our co-simulator for evaluating all alternative algorithms and the LEC algorithm. Then we introduce the evaluation metrics, baseline, real-life dataset, and environment setup. Finally, we demonstrate the superiority of our LEC algorithm on the NR C-V2X and DSRC networks.

6.1 Implementations

Due to the high cost and complexity of CAVs, it is more convenient for researchers to use simulators for evaluation. Although there are many reliable modules [25, 46, 51, 68], combining these simulators is not an easy task. We designed and implemented a co-simulator to evaluate our error concealment algorithms. Different from other work [5, 48, 54, 75, 76], our co-simulator can provide: (i) real-time KITTI-compatible and Semantic3D-compatible ground truth frames, (ii) V2V, V2X, I2V communication modes, (iii) multiple network protocol, DSRC, and NR C-V2X, and (iv) online and offline simulation mode. In addition, we can quickly configure vehicles, pedestrians, and sensors based on the YAML configuration. We use CARLA and NS-3 as our autonomous driving and network simulators, respectively. ZeroMQ (ZMQ) is used as middleware to facilitate communication between CARLA and NS-3. CARLA is an open-source autonomous driving simulator that allows us to add vehicles and pedestrians to various urban maps; we can add different types of sensors and generate virtual sensor data. NS-3 is a packet-level network simulator; we patched NS-3 with Millicar [26] and WaveTest [40] modules for C-V2X and DSRC networks. ZMQ is a powerful cross-process communication library that provides a flexible socket interface to pass messages asynchronously between processes.

Our co-simulator operates in synchronous mode, and CARLA and NS-3 run at different times, as illustrated in Fig. 6.1. First, we start CARLA and NS-3; they will establish a connection through ZMQ. Let $syncTime$ be the acquisition unit time (default is 0.5 sec-

onds). When CARLA executes tick(), a new frame will be generated, and these frames are saved in KITTI-format and semantics3D-format until sufficient $collectionTime$ is reached. When $syncTime > collectionTime$, CARLA will stop generating and sends a resume signal to NS-3. NS-3 starts network simulation to generate network results when it receives the resume signal, and then sends them to CARLA. When CARLA receives the resume signal, CARLA will resume running.

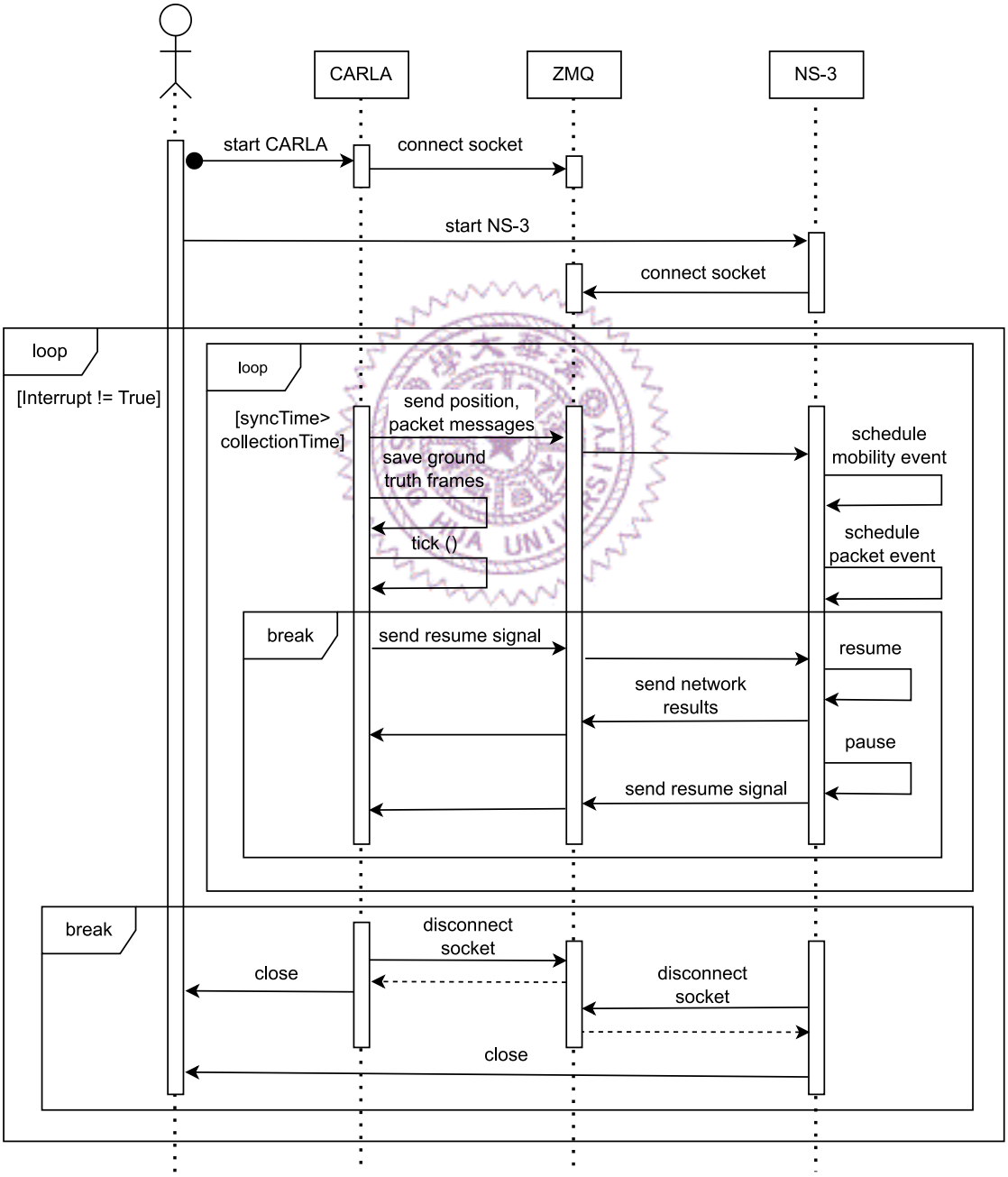


Figure 6.1: The sequence diagram of our co-simulator.

6.2 Experiment Setup

We have implemented the LEC and all alternative algorithms in Python. There exists no algorithm tailored for concealing dynamic LiDAR point clouds for comparison. We include an optimal (OPT) approach, which selects the smallest Chamfer distance among all TP, SI, and TI algorithms, as a benchmark. Algo. 2 demonstrates the pseudo-code of the OPT algorithm. OPT is not practical due to excessive computational complexity and dependency on the ground truth frames. We evaluate the LEC algorithm in two ways: (i) pre-recorded dynamic point cloud datasets, such as KITTI Odometry [30] and (ii) a detailed co-simulator implemented by us following an existing architecture [63] using CARLA [25] and NS-3 [51]. While pre-recorded datasets were captured in real life, they do not provide object detection labels, contain a single LiDAR-equipped vehicle, and cannot reflect interactions among nearby vehicles. Our co-simulator dynamically produces KITTI-compatible datasets, which are labeled with 3D bounding boxes of two relevant object classes: `Car` and `DontCare`. Based on the LiDAR's specifications, we relabel all bounding boxes that are: (i) 100+ m away, (ii) having ≤ 50 points, and (iii) with ≤ 2 visible bounding-box vertices as `DontCare`. This is because such objects are less critical/detectable.

Algorithm 3 Optimal (OPT)

Input: Point Cloud Frames f_{i-1}, f_i, and f_{i+1}.
Output: Concealed frame $\hat{f}_i$.

Let $\check{f}_i$ be the ground truth frame of f_i.
$\hat{f'}_i$ = TP (f_{i-1})
$\hat{f''}_i$ = SI (f_i)
$\hat{f'''}_i$ = TI (f_{i-1}, f_{i+1})
$\hat{f}_i^* = \underset{\hat{f}_i \in \{\hat{f'}_i, \hat{f''}_i, \hat{f'''}_i\}}{\arg\min} d^C(\hat{f}_i, \check{f}_i)$ ▷ Find the optimal concealed frame by Chamfer distance
return $\hat{f}_i^*$

We run simulations with alternative algorithms[1] of the three concealment approaches: TP, SI, and TI, as well as our LEC algorithm. We also give the OPT results in some figures for benchmarking. We adopt the *Town10* map (see Fig. 6.2) of CARLA and Velodyne HDL-64E S2 LiDAR [31] with sectors of $\Psi = 2°$. We deploy a C-V2X base station next to the edge server, which has a transmission range of ~500 m; and a set of DSRC APs separated by 20 m. We randomly select the initial positions of the CAVs with LiDARs, as illustrated in Fig. 6.3. In addition, 100 vehicles without LiDARs are added to the map. All vehicles move along the street and make random turns at the intersection. Each simulation

[1]The names of the alternative algorithms were given in Chapter 5.

lasts for 60 seconds at 10 Hz.

We consider the following performance metrics:

- *Chamfer distance:* Average deviation defined in Eq. (4.2).
- *Hausdorff distance:* Maximum deviation given by:

$$d^H(\hat{f}_i, f_i) = \max\left\{\sup_{p\in\hat{f}_i}\inf_{p'\in f_i}\left\|p-p'\right\|_2^2, \sup_{p\in f_i}\inf_{p'\in\hat{f}_i}\left\|p-p'\right\|_2^2\right\}.$$

- *Running time:* We measure the running time on an Intel Xeon-E5 2629 V4 CPU and an NVIDIA GTX 1080Ti GPU. Only SF and BSF leverage the GPU.
- *Intersection-over-Union (IoU):* We use the pre-trained PointRCNN [60] to detect vehicles in front of CAVs. A candidate vehicle having an IoU with ≥ 0.5 the ground-truth vehicle is considered detected.
- *Detection Accuracy:* The fraction of detected vehicles.

We repeat each simulation 10 times and report the average results from a random vehicle. We include 95% confidence intervals whenever possible.

Figure 6.2: The town10 map of CARLA.

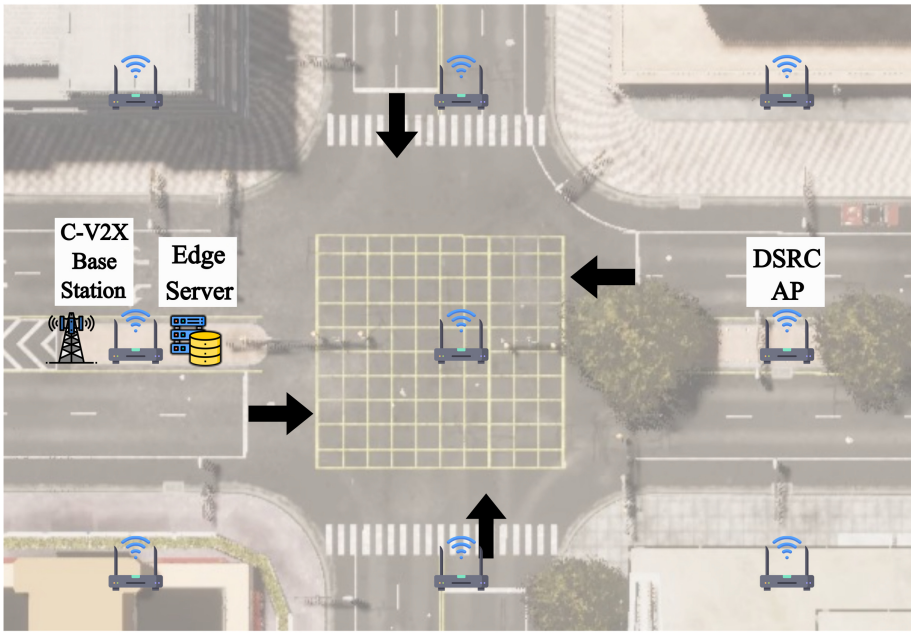


Figure 6.3: The urban map used in our simulations.

6.3 Results

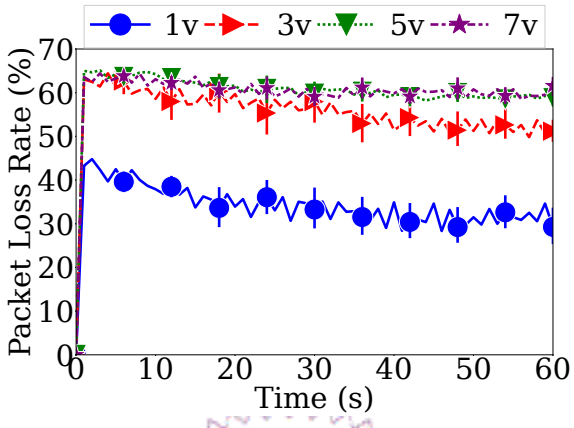
In this thesis, we implemented all of the network protocols (see Chapter 4.4). We focus on the PU messages, which dominate the network traffic. Although we upload the compressed point cloud, PU messages still occupy about 99.98% of the overall network load. In the following experiments, we do not evaluate LU and CR messages, but this has little impact on the overall results.

Table 6.1: Parameters of C-V2X and DSRC Networks.

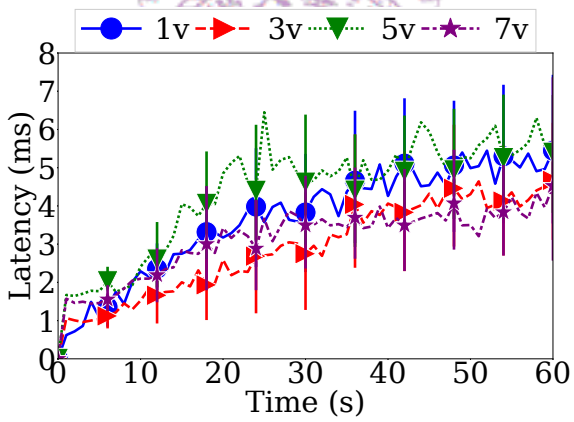
	Bandwidth	**Datarate**	**Frequency**	**Mode**	**Transport Layer**	**Physical Layer**
C-V2X	100 MHz	-	28 GHz	unicast	UDP	5G New Radio
DSRC	10 MHz	27 Mbps	5.9 GHz	broadcast	UDP	IEEE 802.11p (Wave)

Network statuses of C-V2X and DSRC. We use the parameters of networks shown in Table 6.1. The network statuses of C-V2X and DSRC are shown in Figs. 6.4 and 6.5, respectively. Due to randomness such as driving directions and traffic lights, congestion, etc., the results may be unbalanced. As shown in the figure, transmit point cloud may cause 35.47%–65.13% of packet loss rate in the DSRC network. This is serious for classification tasks. In addition, the C-V2X network still has 33.77%–59.87% packet loss, which cannot guarantee stable transmission. Due to the powerful 5G technology, C-V2X has a latency of only 0.2–6.6 ms and greater throughput. On the other hand, DSRC is based on AP for long-distance transmission, which still has a high latency. Due to cellular network limitations, C-V2X does not usually support broadcast modes, which are not

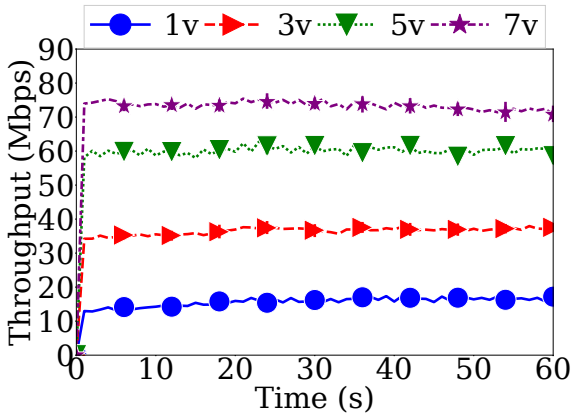
conducive to the spread of messages. DSRC supports ad-hoc network topology, where vehicles and edge servers can broadcast messages to each other, which makes it more suitable for data sharing and regional alerting. *In our system, we recommend using C-V2X to transmit LU and PU messages from CAVs to an edge server, and DSRC to transmit CR messages from an edge server to nearby CAVs.*



(a)

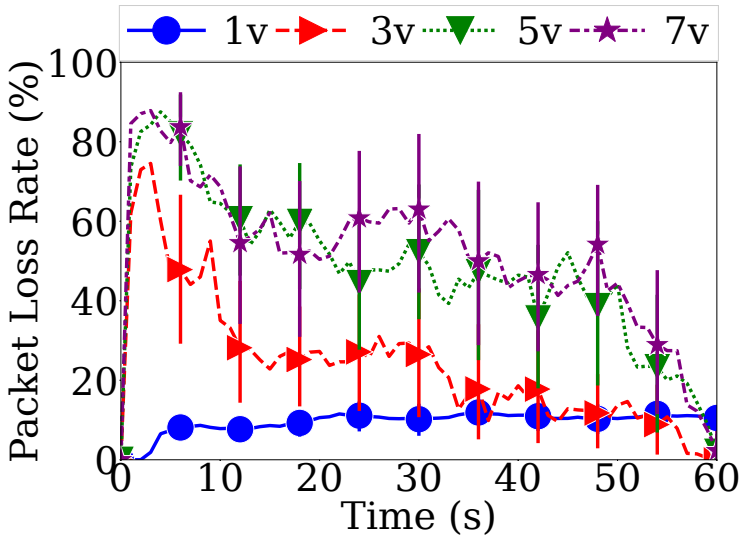


(b)

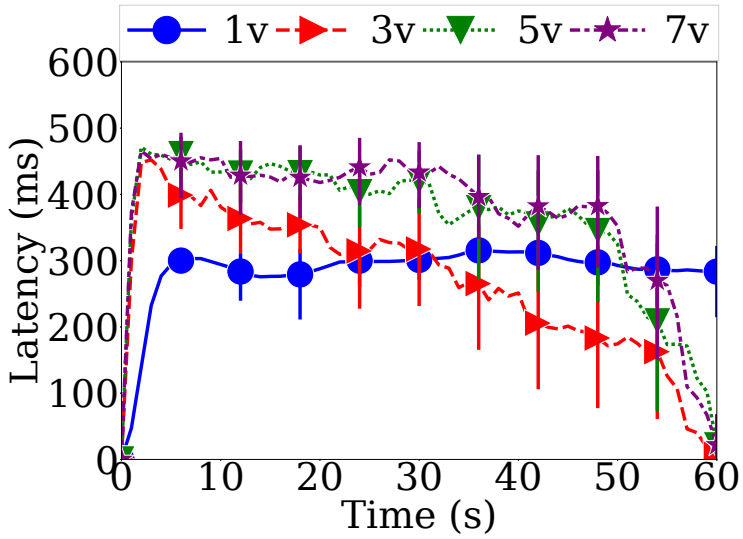


(c)

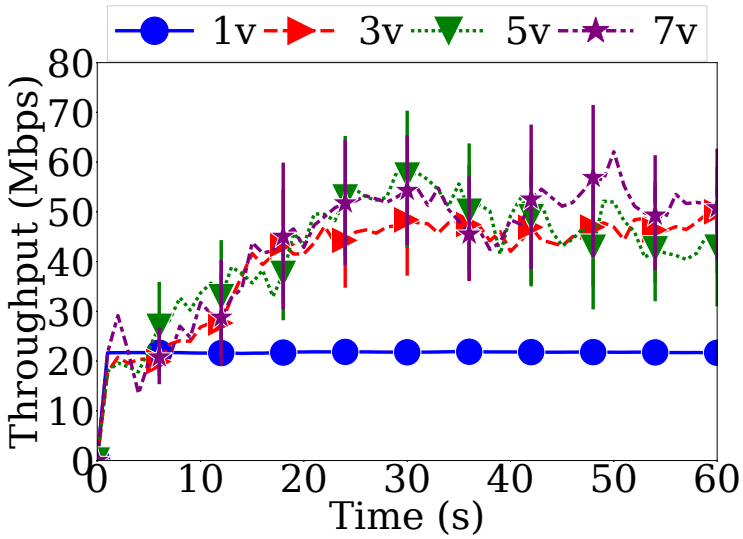
Figure 6.4: Network status in a C-V2X network: (a) packet loss rate, (b) latency, and (c) throughput.



(a)



(b)



(c)

Figure 6.5: Network status in a DSRC network (a) packet loss rate, (b) latency, and (c) throughput.

Table 6.2: Overall Results in LEC Training

	DT	**SVM**	**RF**
Training Acc. (%) ↑	83.12	80.52	84.09
Validation Acc. (%) ↑	76.14	79.21	78.34

Training of the LEC algorithm. We first conduct experiments to find the hyperparameters of the DT, SVM, and RF models of LEC. We use our co-simulator to generate 10,000 consecutive frames and randomly simulate packet loss rates between 0%–100% for each frame. We carry out 5-fold cross-validation to find the optimal hyperparameters, where 80% of the frames are used for training and 20% for validation. We vary the following hyperparameters: (i) maximal tree depth $\in \{1, \underline{5}, 10, 15, 20\}$ in DT, (ii) kernel $\in \{linear, poly, \underline{rbf}, sigmoid, precomputed\}$ and regularization $\in \{1, 5, \underline{10}, 25, 20\}$ in SVM, and (iii) maximal tree depth $\in \{1, 5, \underline{10}, 25, 20\}$ and number of trees $\in \{10, 50, \underline{100}, 150, 200\}$ in RF. The optimal hyperparameters are underlined above and are adopted in the rest of this thesis. We used these frames and parameters to train our LEC model, and the overall results are shown in Table 6.2. Next, we report the overall Chamfer distance and running time from DT, SVM, and RF in Figs. 6.6(a) and 6.6(c), also using 5-fold cross-validation. This figure shows that DT outperforms both SVM and RF by up to 17.17%, and can save 10.53% and 15.00% running time, respectively. We also report the results of Hausdorff distance, average IoU, and detection accuracy in Figs. 6.6(b), 6.6(d), and 6.6(e). Since DT outperforms SVM and RF in Chamfer distance and runs faster, the LEC algorithm adopts DT as the decision model.

Design decision of concealment approaches. To identify the best-performing alternative algorithms in individual approaches (TP, SI, TI), we first give sample results from 3 vehicles with a C-V2X network in Figs. 6.7–6.9. Fig. 6.7(a) shows that MP leads to the Chamfer distance between 0.12–11.87 m. Furthermore, Figs. 6.8(a) and 6.9(a) show that MP results in the Hausdorff distance between 12.27–28.64 m and a per-frame running time of $<$ 1152 ms. Therefore, MP works the best among all TP alternative algorithms. Similarly, Figs. 6.7(b), 6.7(c), 6.8(b), and 6.8(c) reveal that NN and BSF achieve the lowest Chamfer and Hausdorff distances among alternative algorithms of the SI and TI approaches. Figs 6.7(d), 6.8(d), and 6.9(d) further confirmed the superiority of DT compared to SVM and RF. Moreover, we plot the pre-frame results of the LEC algorithm and all alternative algorithms from 1-, 5-, and 7-vehicle simulations in Figs. 6.10–6.18.

Next, we plot the overall Chamfer distance, Hausdorff distance, running time, average IoU, and detection accuracy of all alternative algorithms from 1-, 3-, 5-, and 7-vehicle simulations in Figs. 6.19–6.23 using a co-simulator dataset in a C-V2X network. We observe that MP, NN, and BSF are the best algorithms for the TP, SI, and TI approaches, re-

spectively. Particularly, MP, NN, and BSF reduce the Chamfer distance by up to 73.28%, 98.47%, and 71.48%, compared to other alternative algorithms in TP, SI, and TI respectively. We can say the same for the Hausdorff distance with up to: 32.96%, 73.58%, and 9.84%. Similarly, the average IoU is up to: 0.26%, 2.11%, and -0.14%, and the detection accuracy is up to: 2.74%, -14.25%, and 1.49%. *In summary, we recommend using MP, NN, and BSF in the TP, SI, and TI approaches. For brevity, MP, NN, BSF and TP, SI, TI are used interchangeably in the following discussion.*

Performance of LEC with the co-simulator dataset. We compare the performance of LEC with other algorithms in Fig. 6.24. We observe the superior performance of our LEC algorithm, which: (i) reduces the Chamfer distance by up to 75.77%, (ii) cuts the Hausdorff distance by up to 30.17%, and has a small gap of at most 25.55% compared with OPT, (iii) a small IoU gap up to 0.04% compared with OPT, and (iv) improves the detection accuracy by at most 33.31% with a tiny gap of as small as 0.75% compared with OPT. In addition, our LEC algorithm runs fast: it terminates in 360–570 ms throughput our evaluations.

Performance of LEC in a DSRC network. Also with the co-simulator dataset, we present sample results from 3-vehicle simulations in a DSRC network. We observe an average packet loss rate of 40.18%. Compared to C-V2X, the DSRC network often causes longer inter-packet intervals, which negatively affect the overall performance. Table 6.3 gives the overall results, which shows that our LEC algorithm outperforms TP, SI, and TI in Chamfer and Hausdorff distances by 12.25%–87.43% and 2.46%–66.58%, respectively. Compared to OPT, our LEC algorithm saves on average 74.47% running time, while achieving a small gap of 8.86%, 2.11%, 0.04%, and 0.45% in Chamfer distance, Hausdorff distance, average IoU, and detection accuracy, respectively. We notice that the running time of OPT is underestimated, as selecting the best approach requires multiple Chamfer distance calculations, which is extremely time-consuming.

Performance of LEC in the pre-recorded KITTI dataset. We present sample results from 3-vehicle simulations with the KITTI Odometry dataset in a C-V2X network. We use the same hyperparameters but retrain DT, SVM, and RF with 4071 frames from sequence 8 with 5-fold cross-validation. DT still performs the best and is used in our LEC algorithm model. This KITTI pre-recorded dataset only contains one LiDAR-equipped vehicle. Hence, we duplicate trajectory in sequence 0 three times and time-shift it by 10 and 20 seconds to create a 3-vehicle dataset. Table 6.3 gives the overall results without average IoU and detection accuracy because the KITTI Odometry dataset does not contain labels. We observe an average packet loss rate of 64.93%. Our LEC algorithm outperforms TP, SI, and TI in Chamfer and Hausdorff distances by 2.56%–92.49% and 0.58%–62.48%, respectively. Moreover, our LEC algorithm saves 70.72% of the running

time compared to OPT, with small gaps of 5.56% and 2.59% in Chamfer and Hausdorff distances.

Table 6.3: Performance Comparison in: (Top) DSRC, (Bottom) C-V2X Networks

	TP	**SI**	**TI**	**LEC**	**OPT**
Co-simulator Dataset in a DSRC Network					
Chamfer D. (m) ↓	0.98	6.84	3.23	0.86	0.79
Hausdorff D. (m) ↓	8.95	26.12	20.92	8.73	8.55
Run. Time (ms) ↓	589	1130	570	589	2307
IoU (%) ↑	66.59	66.34	66.38	66.75	66.79
Accuracy (%) ↑	52.52	45.96	52.91	53.91	54.36
Pre-recorded Dataset in a C-V2X Network					
Chamfer D. (m) ↓	5.06	1.37	0.39	0.38	0.36
Hausdorff D. (m) ↓	31.93	14.08	12.05	11.98	11.67
Run. Time (ms) ↓	50	820	420	380	1298

In summary, our LEC algorithm outperforms the best TP, SI, and TI alternative algorithms in C-V2X and DSRC networks. In addition, our LEC algorithm can run faster and achieves small gaps from OPT.

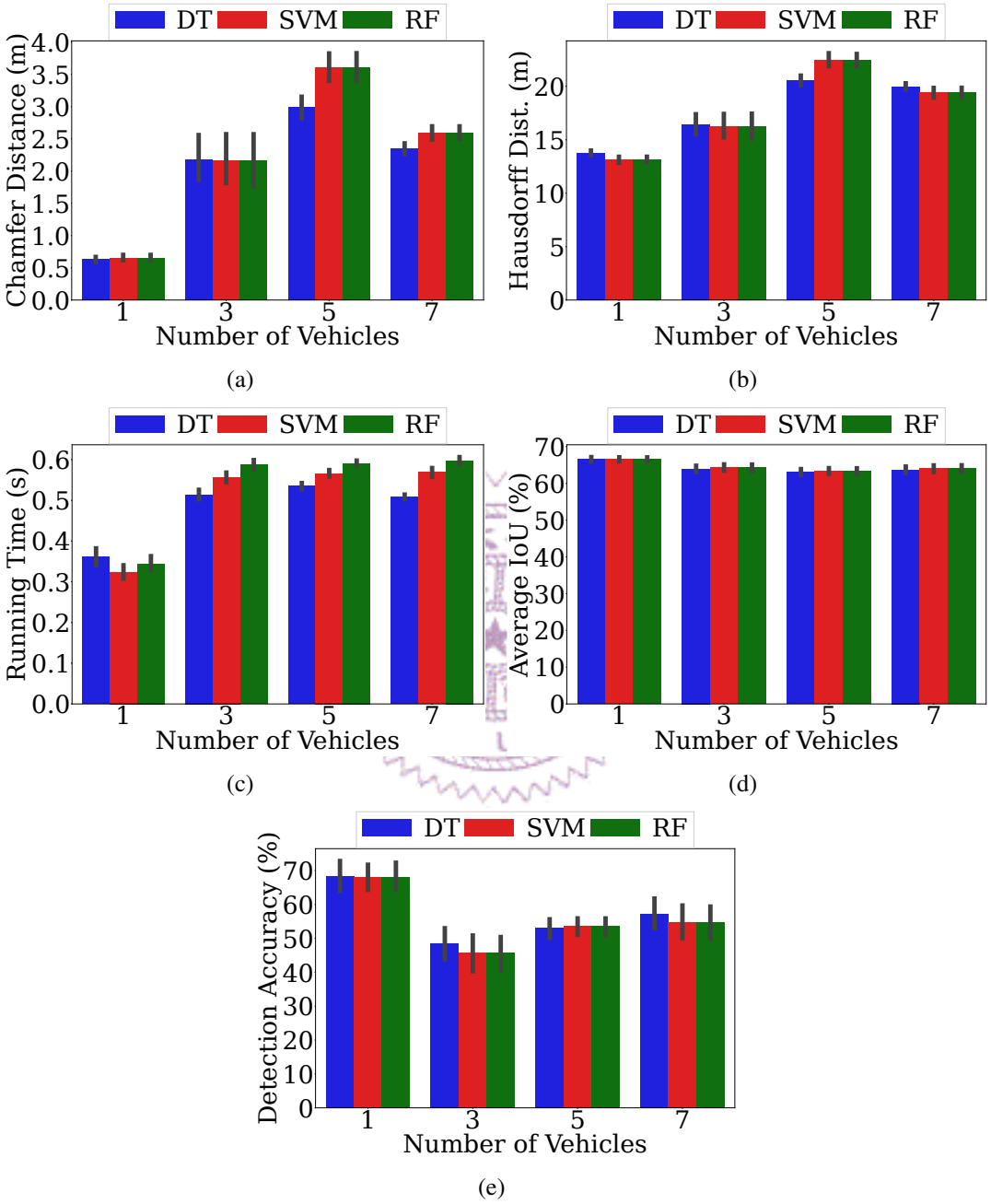


Figure 6.6: Simulation results from our LEC algorithm in a C-V2X network: (a) Chamfer distance, (b) Hausdorff distance, (c) running time, (d) average IoU, and (e) detection accuracy.

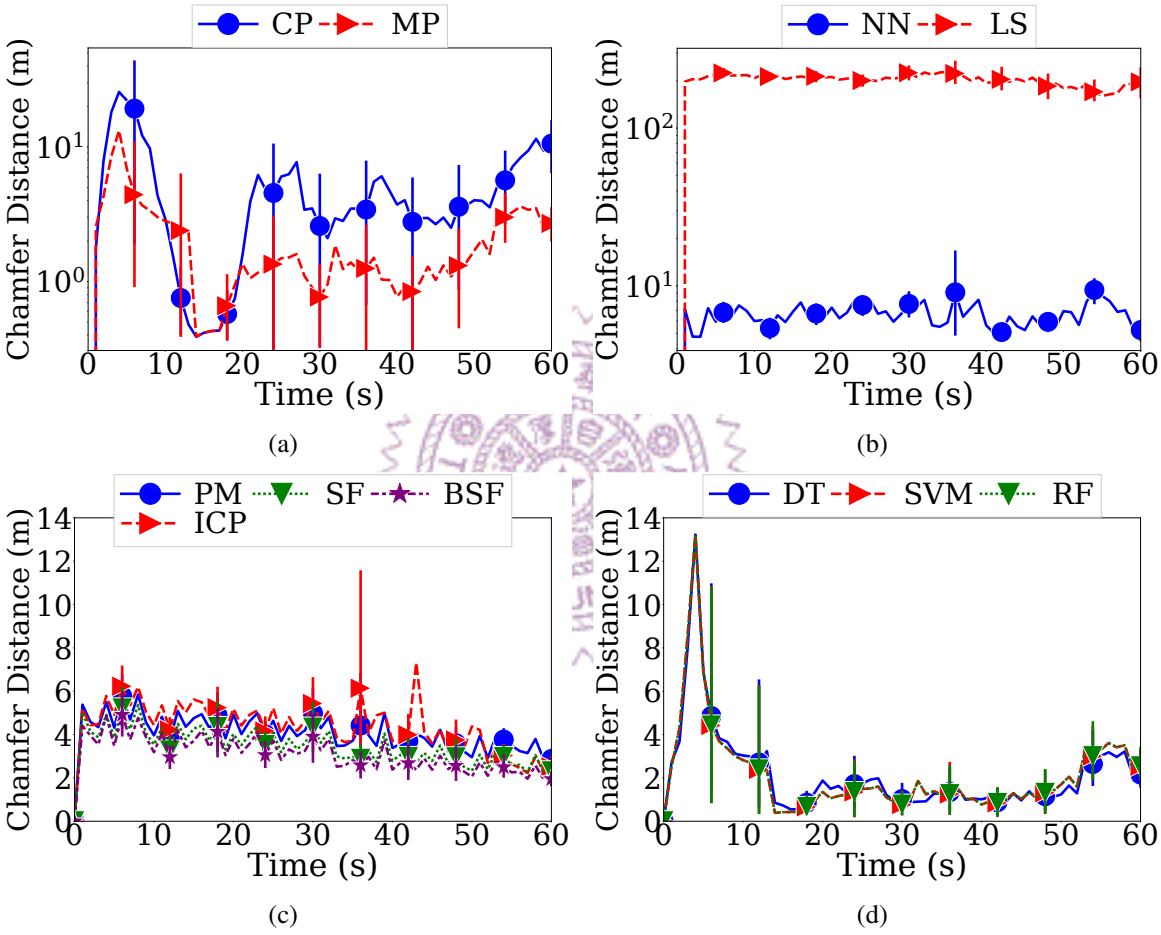


Figure 6.7: Sample performance results from 3 vehicles in a C-V2X network: Chamfer distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

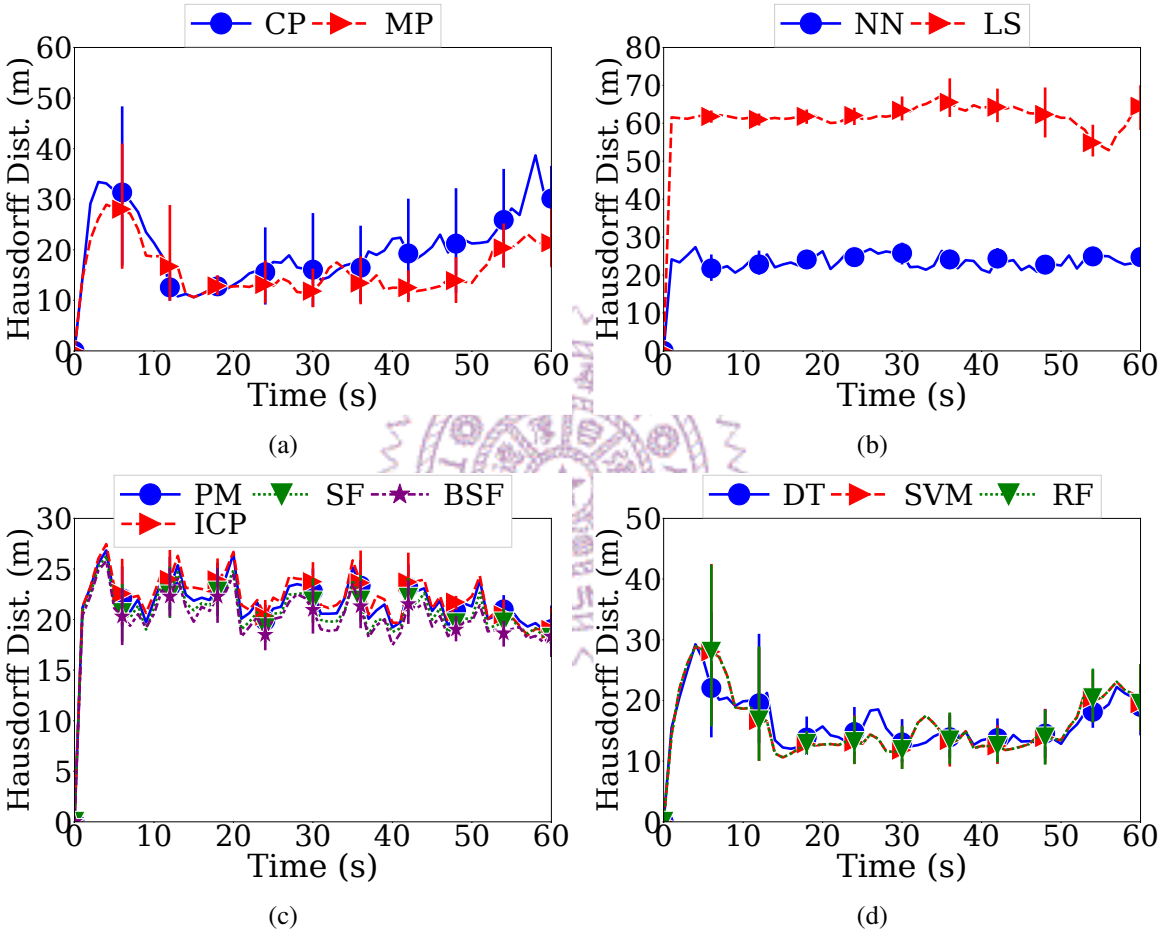


Figure 6.8: Sample performance results from 3 vehicles in a C-V2X network: Hausdorff distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

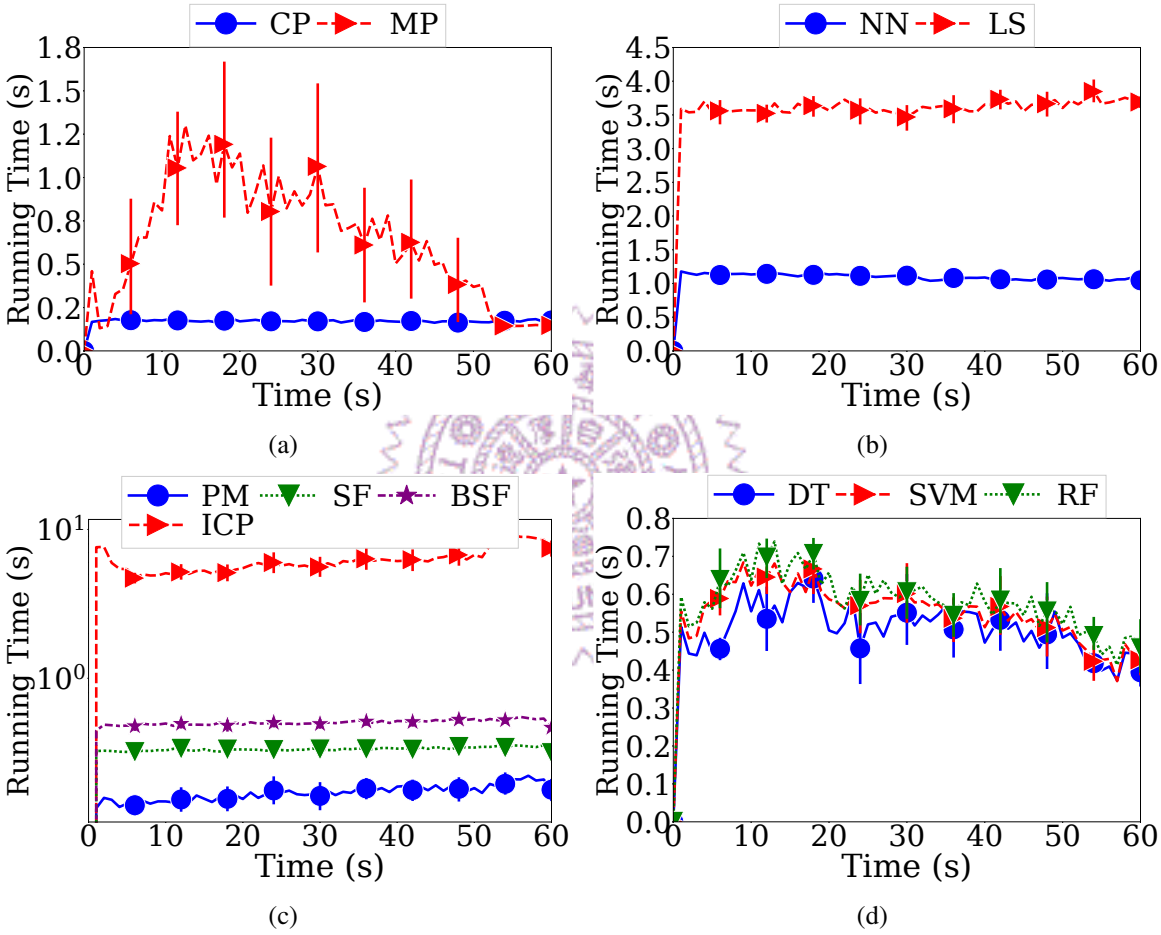


Figure 6.9: Sample performance results from 3 vehicles in a C-V2X network: Running time of (a) TP, (b) SI, (c) TI, and (d) LEC.

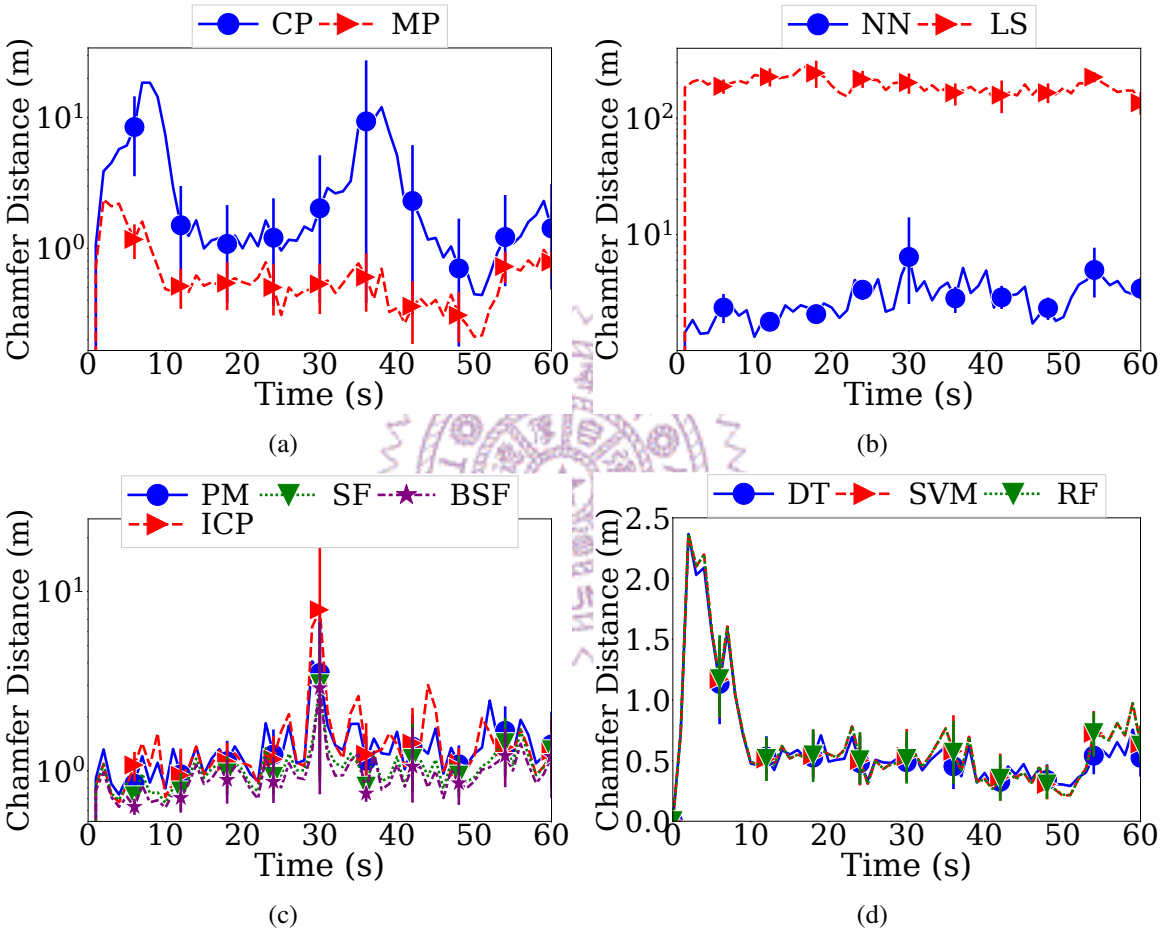


Figure 6.10: Sample performance results from 1 vehicle in a C-V2X network: Chamfer distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

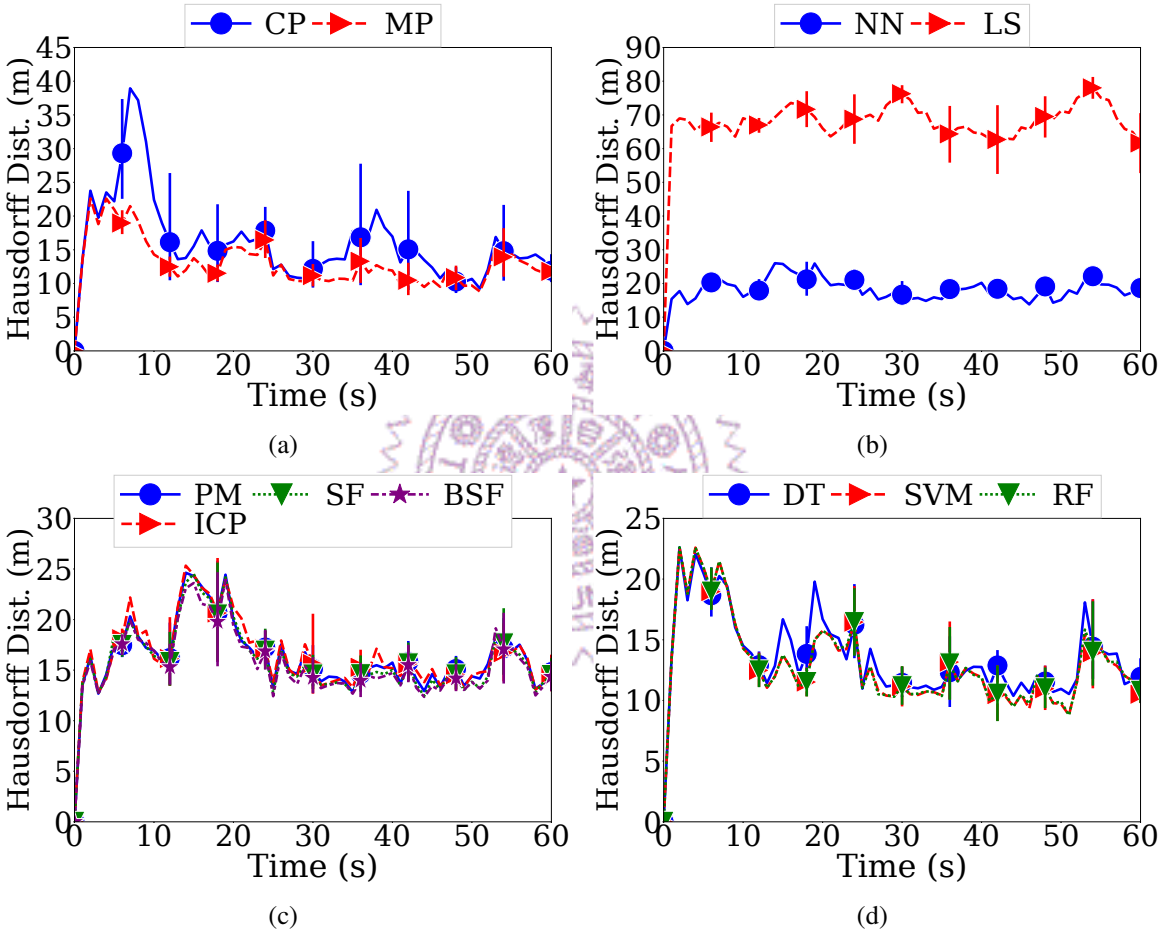


Figure 6.11: Sample performance results from 1 vehicle in a C-V2X network: Hausdorff distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

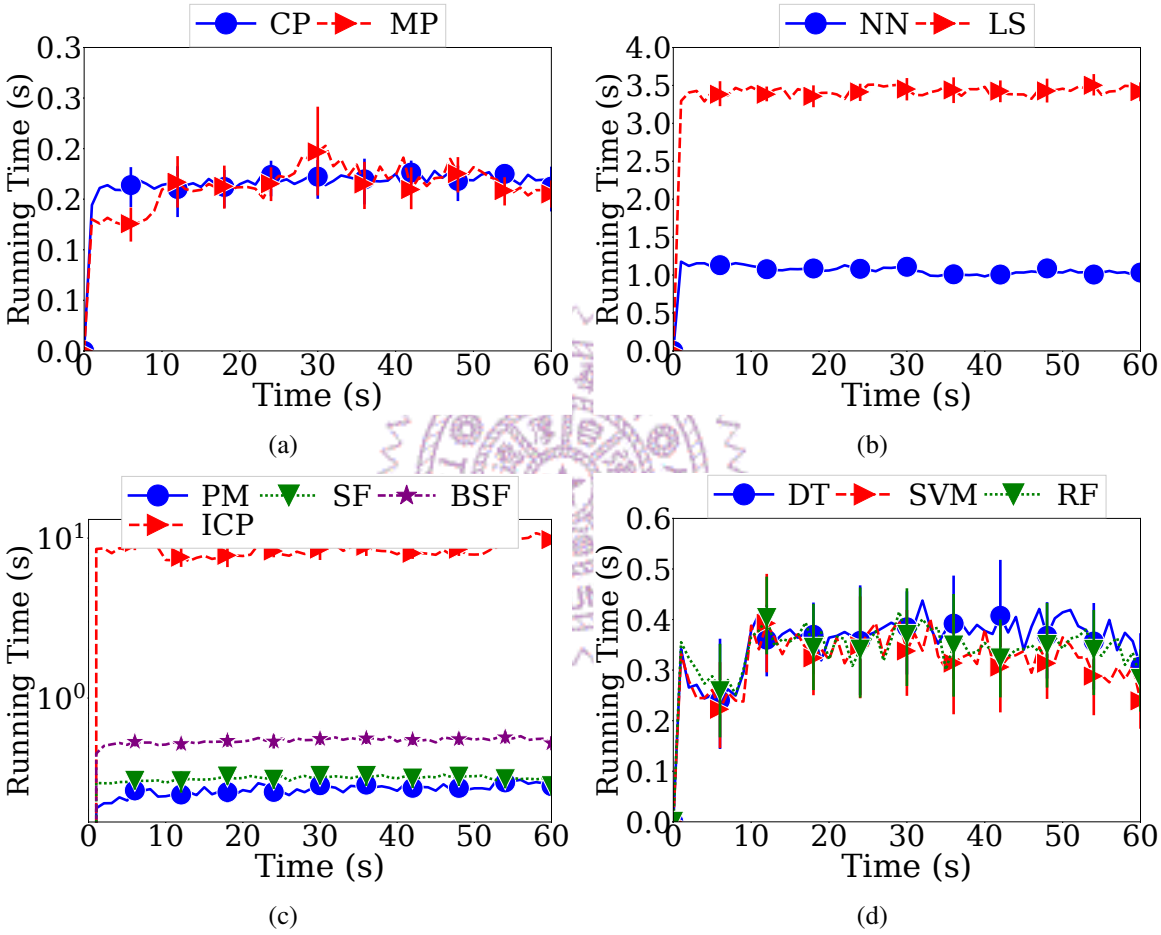


Figure 6.12: Sample performance results from 1 vehicle in a C-V2X network: Running time of (a) TP, (b) SI, (c) TI, and (d) LEC.

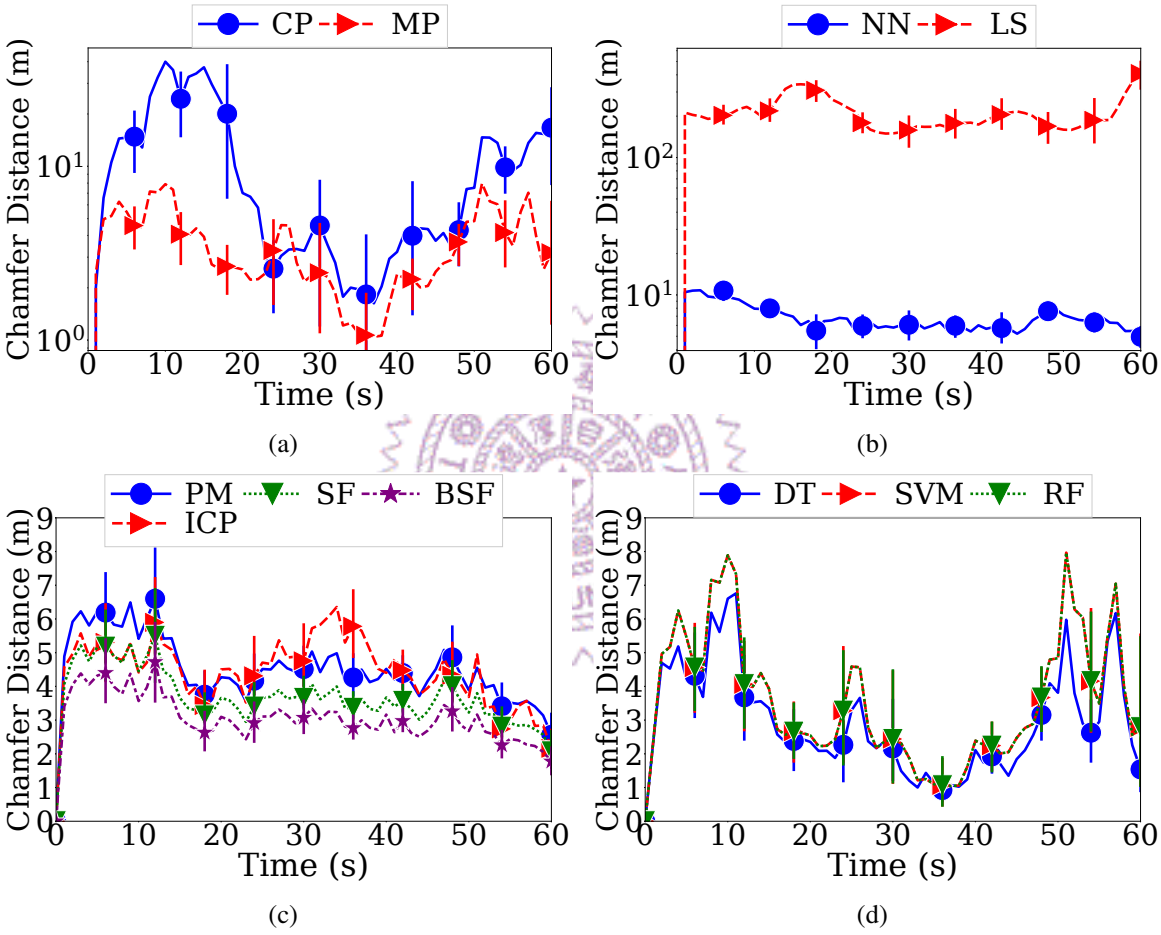


Figure 6.13: Sample performance results from 5 vehicles in a C-V2X network: Chamfer distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

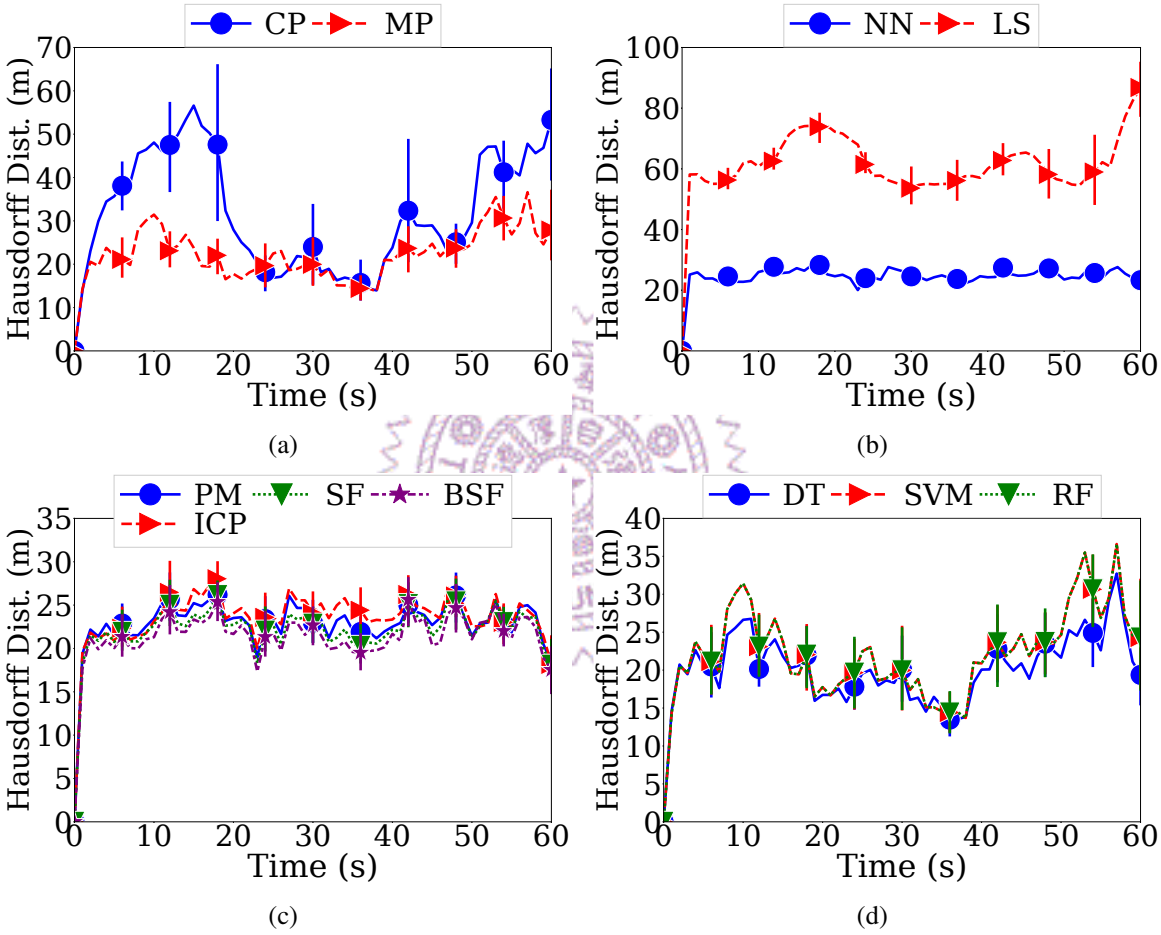


Figure 6.14: Sample performance results from 5 vehicles in a C-V2X network: Hausdorff distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

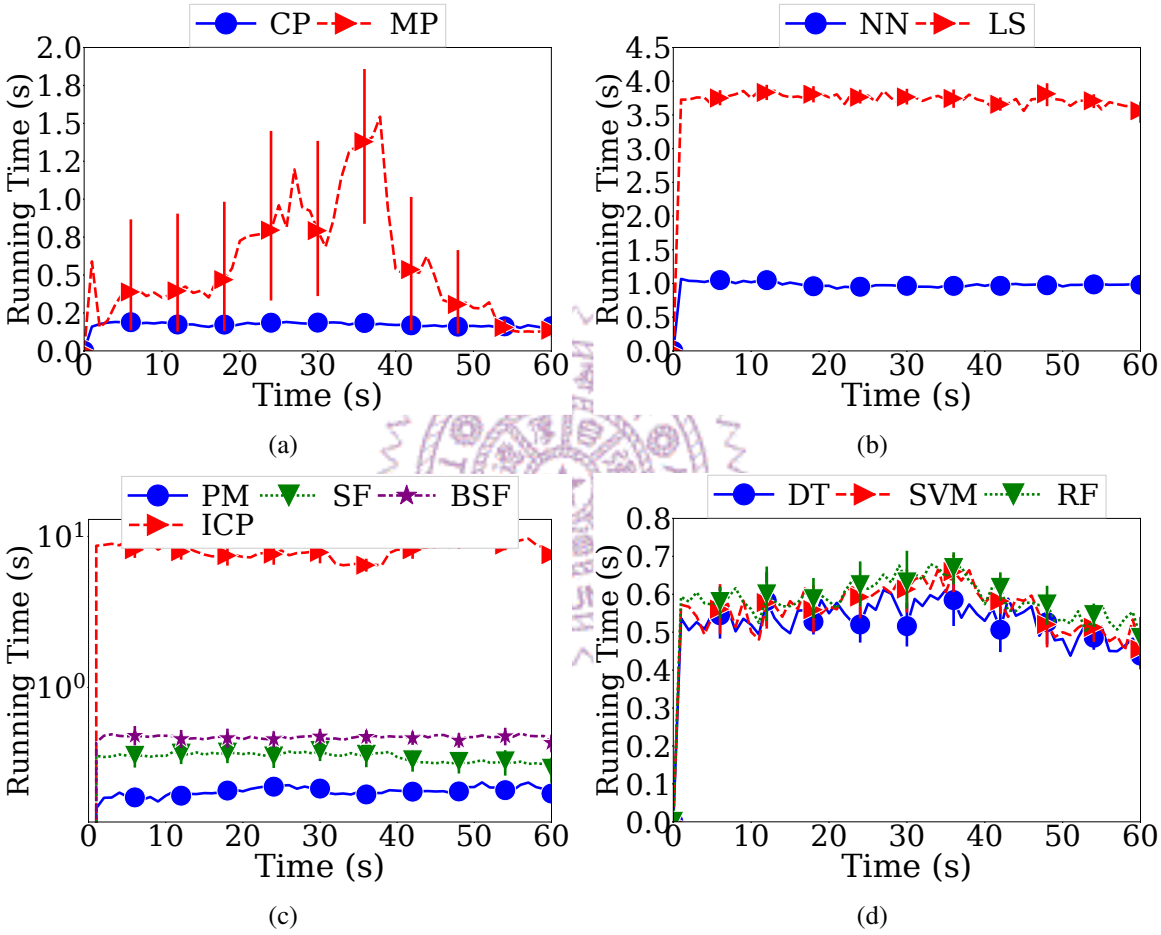


Figure 6.15: Sample performance results from 5 vehicles in a C-V2X network: Running time of (a) TP, (b) SI, (c) TI, and (d) LEC.

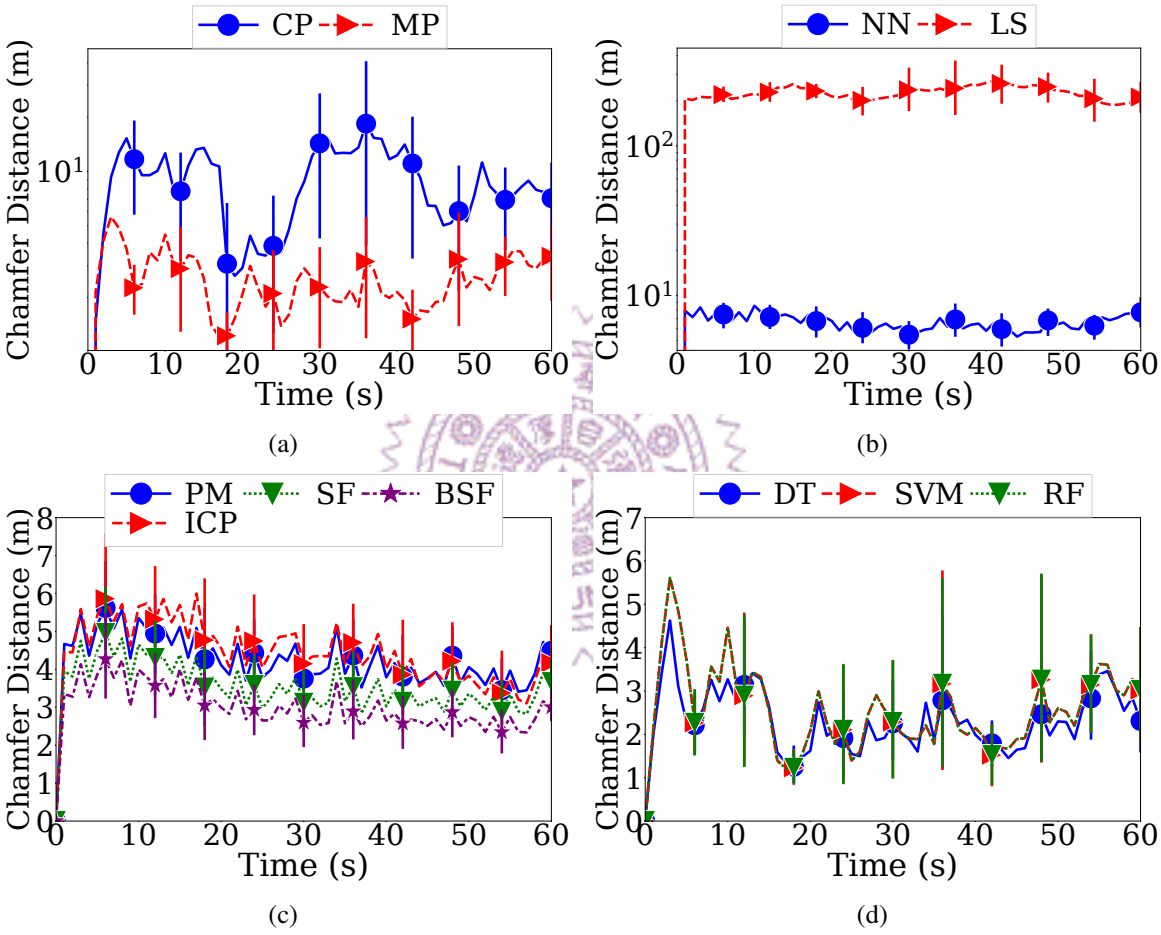


Figure 6.16: Sample performance results from 7 vehicles in a C-V2X network: Chamfer distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

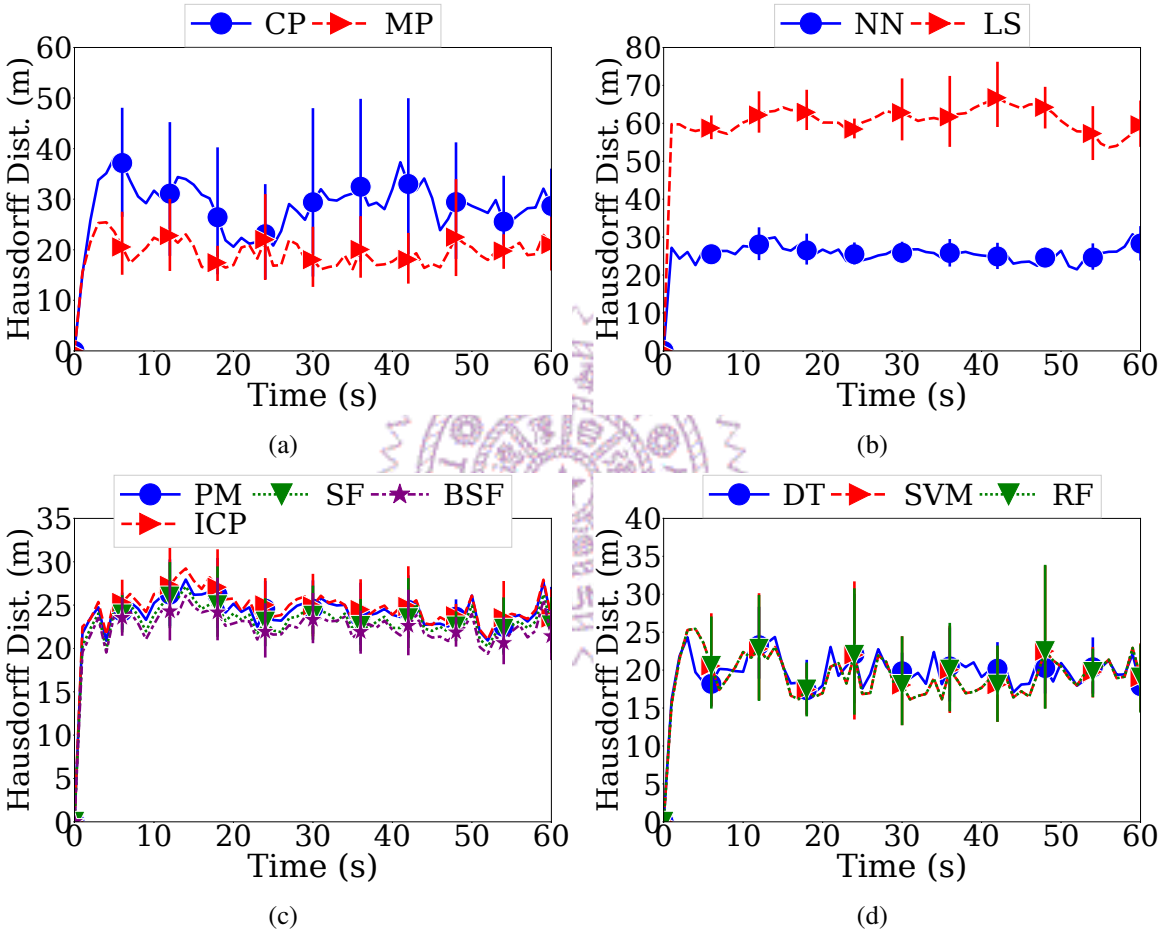


Figure 6.17: Sample performance results from 7 vehicles in a C-V2X network: Hausdorff distance of (a) TP, (b) SI, (c) TI, and (d) LEC.

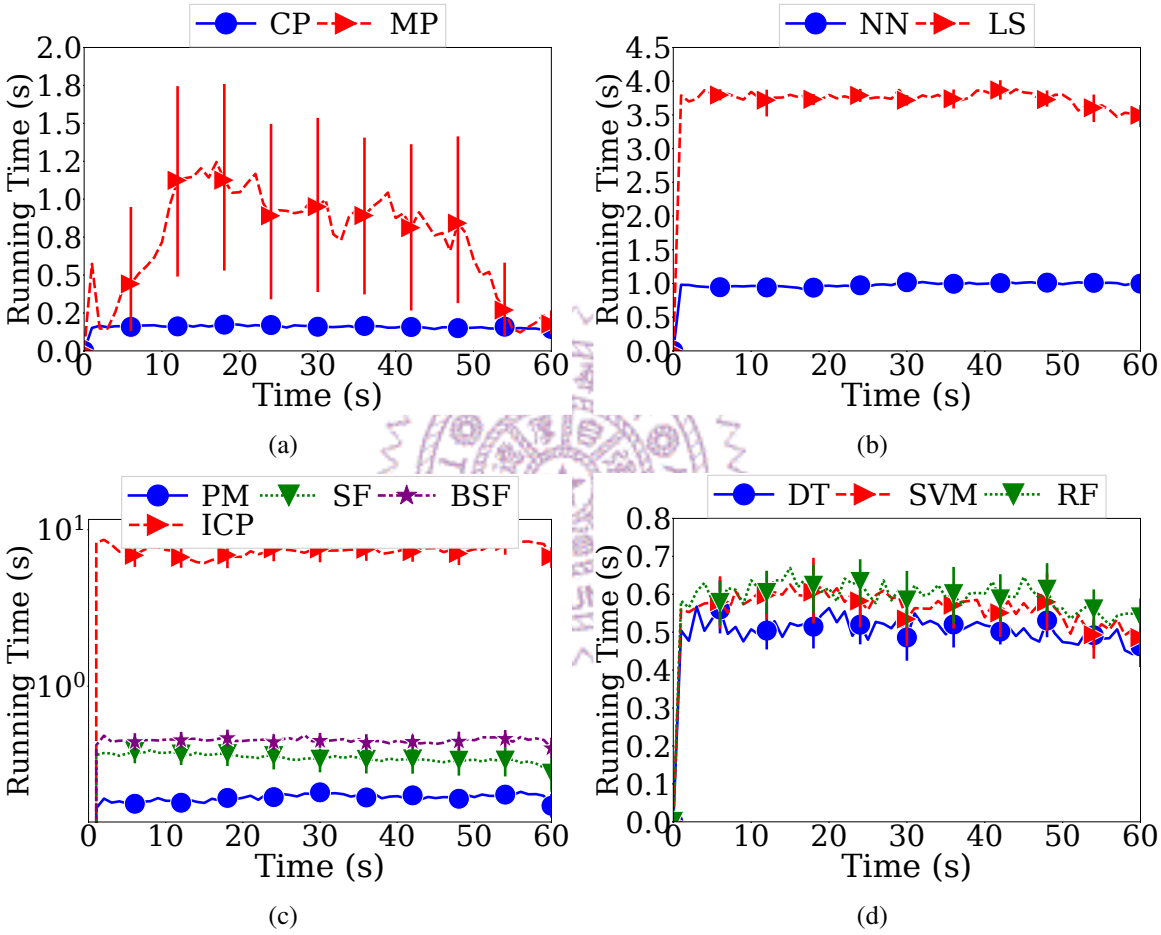


Figure 6.18: Sample performance results from 7 vehicles in a C-V2X network: Running time of (a) TP, (b) SI, (c) TI, and (d) LEC.

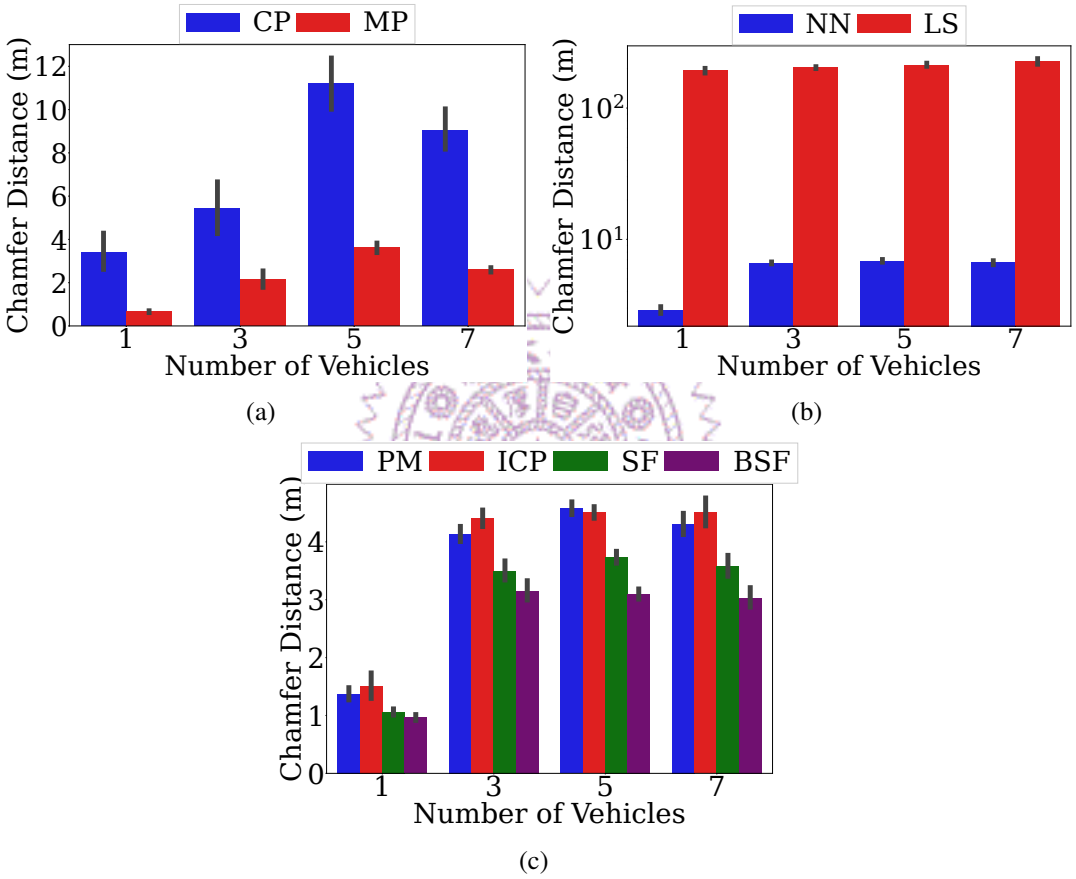


Figure 6.19: Simulation results from different numbers of vehicles in a C-V2X network. Chamfer distance from: (a) TP, (b) SI, and (c) TI.

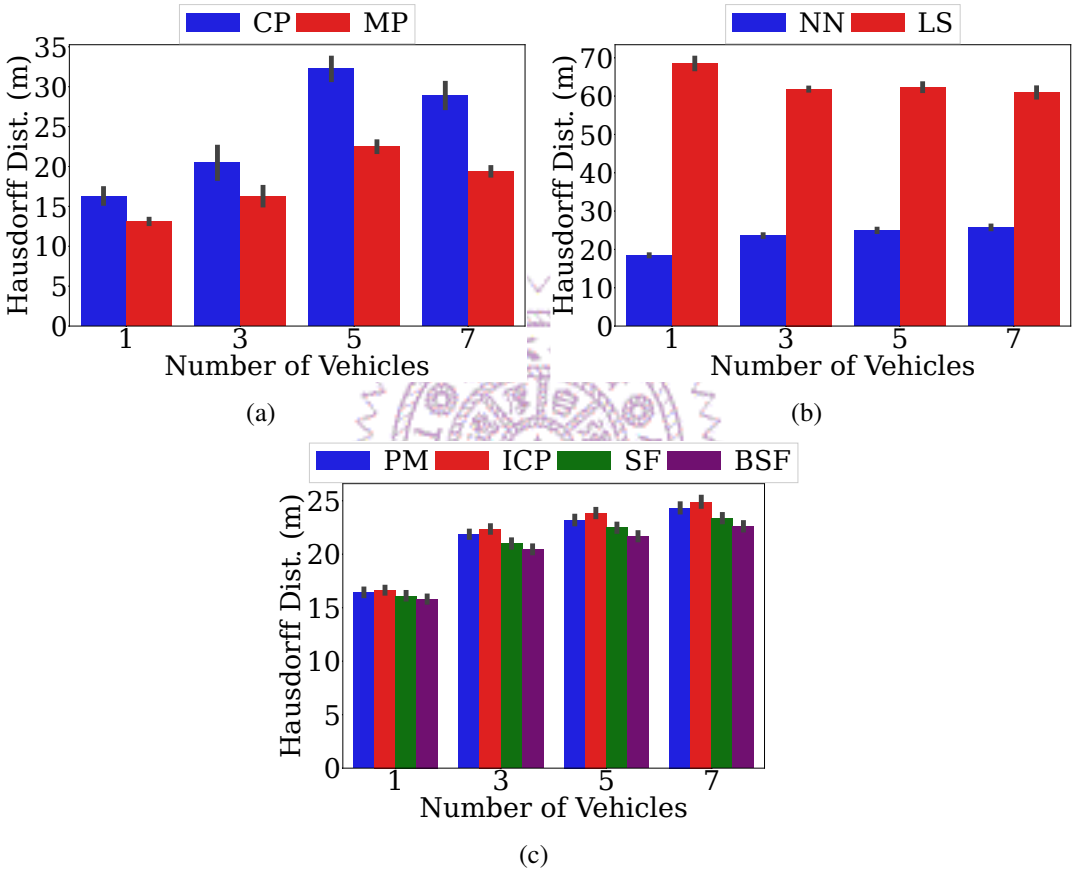


Figure 6.20: Simulation results from different numbers of vehicles in a C-V2X network. Hausdorff distance from: (a) TP, (b) SI, and (c) TI.

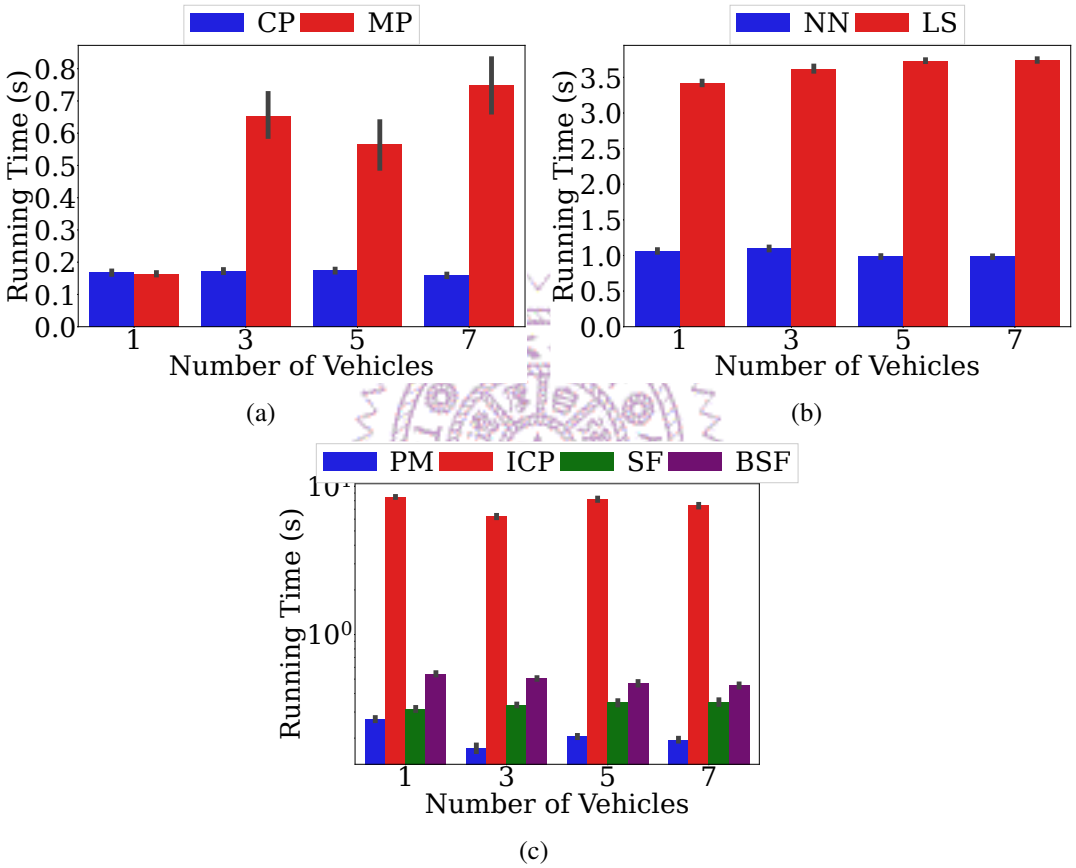


Figure 6.21: Simulation results from different numbers of vehicles in a C-V2X network. Running time from: (a) TP, (b) SI, and (c) TI.

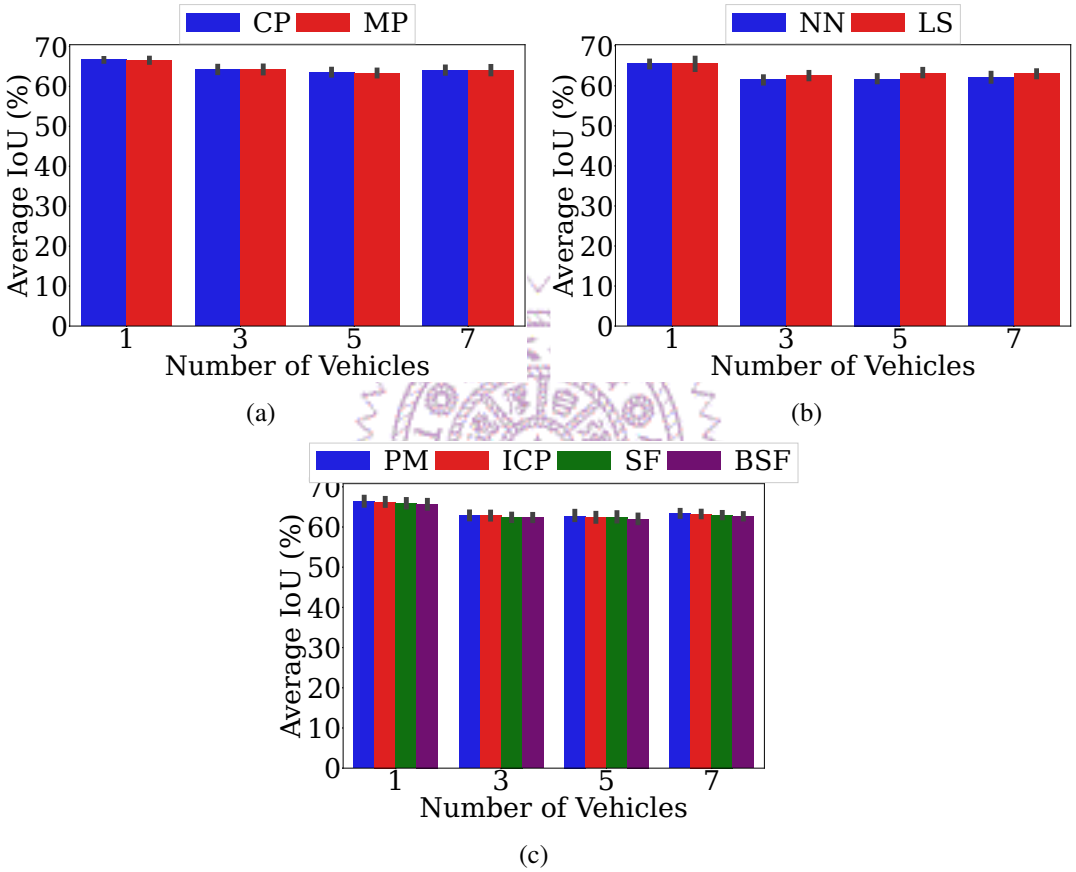


Figure 6.22: Simulation results from different numbers of vehicles in a C-V2X network. Average IoU from: (a) TP, (b) SI, and (c) TI.

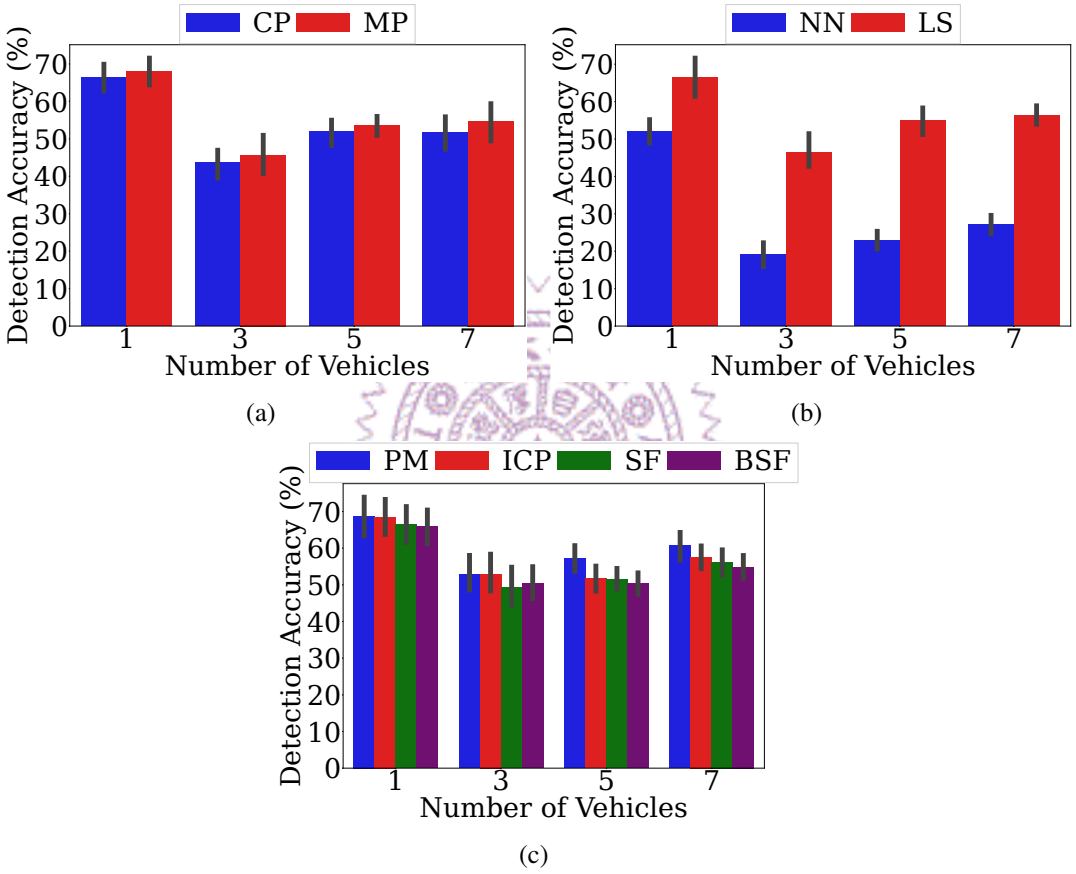


Figure 6.23: Simulation results from different numbers of vehicles in a C-V2X network. Detection accuracy from: (a) TP, (b) SI, and (c) TI.

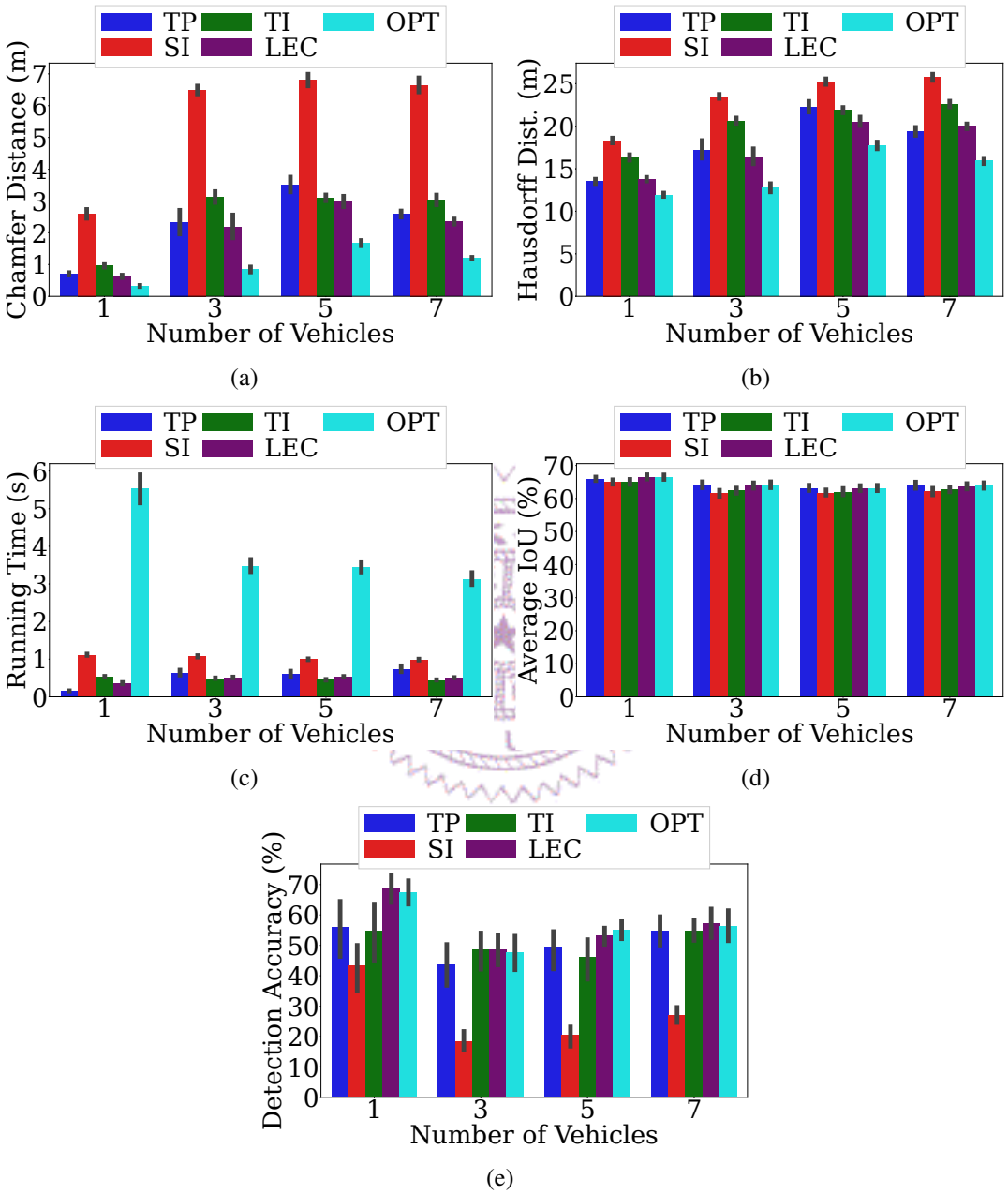


Figure 6.24: Simulation results from our LEC algorithm in a C-V2X network: (a) Chamfer distance, (b) Hausdorff distance, (c) running time, (d) average IoU, and (e) detection accuracy.

Chapter 7

Conclusion and Future Work

In this chapter, we conclude our work and discuss the possibility of our error concealment approaches for real-world CAVs. This is followed by the future work.

7.1 Concluding Remarks

In this thesis, we considered a dynamic point cloud streaming system from a LiDAR-equipped CAV to an edge server for classification tasks. Although streaming point clouds over unreliable wireless networks could lead to incomplete point cloud frames due to packet loss and late packets, the error concealment problem of dynamic LiDAR point clouds has never been studied in the literature. To fill this gap, we designed, developed, and evaluated a LiDAR Error Concealment (LEC) algorithm, which adaptively employs Temporal Prediction (TP), Spatial Interpolation (SI), or Temporal Interpolation (TI) approaches to conceal incomplete point cloud frames for the smallest deviation in Chamfer distance. Such decisions are made by comparing the incomplete ratios of adjacent point cloud frames. For each concealment approach, we proposed and compared multiple alternative algorithms using a comprehensive co-simulator of CARLA and NS-3 implemented by us and the KITTI Odometry dataset. Our simulation results revealed that the LEC algorithm outperforms the alternative algorithms by up to 82.68% and 30.17% in Chamfer and Hausdorff distances, within at most 570 ms per-frame running time in a C-V2X network. Our LEC algorithm also outperforms other algorithms by up to 87.43% and 66.58% in Chamfer and Hausdorff distances in a DSRC network. In the KITTI dataset, our LEC algorithm outperforms the alternative algorithms in Chamfer and Hausdorff distances by up to 92.49% and 62.48%. Moreover, our LEC algorithm saves 70.72% of the running time compared to OPT, with small gaps of 5.56% and 2.59% in Chamfer and Hausdorff distances.

However, some additional work is needed to combine our error concealment algo-

rithm with real-world CAVs. For each CAV, we recommend Velodyne HDL-64E S2 or Livox HAP as onboard-LiDAR, which are typically mounted in front of the vehicle roof. To work with other sensors, we need to align the position of these sensors. We recommend using the C-V2X communication module to upload point clouds and the DSRC communication module to receive messages from the edge server. We recommend using a mini-PC to pre-process point clouds in CAVs and a stronger computer with GPU in the edge server for error concealment and classification tasks.

7.2 Future Work

Our work can be extended in several directions:

- *A smarter model for the LEC algorithm.* In our work, we only consider the DT, SVM, and RF models, which are low-complexity ML models. These models are known to be vulnerable to *overfitting*, which may be attributed to their limited ability to capture the complex relationship between features, especially the nonlinear problems. We may apply Reinforcement Learning (RL) [39] to the LEC algorithm, or use an Incremental Decision Tree algorithm [67] to keep the model up-to-date. RL makes decisions by learning an optimal strategy, and Incremental Decision Tree makes decisions by dividing features. Compared with decision tree, RL is more suitable for complex environments for adaptive adjustment and has advantages for long-term decision problems. However, designing and tuning RL algorithms require expertise and experience to ensure fast convergence and good performance. The Incremental Decision Tree algorithm incorporates new features to avoid concept drift. Hence, the algorithm can adapt to new feature distributions promptly while maintaining the accuracy of historical data. On the other hand, incomplete ratios as input to the decision model may cause higher latency, and we have not considered the spatial relationship between the lost sectors. All these challenges need to be carefully considered.

- *A more complete messaging protocol.* In our work, we focus on designing, developing, and evaluating error concealment algorithms over transmitted PU messages at an intersection. In the future, we can perform larger-scale simulations, which can further highlight the merits of our solution for automatic driving. We need to evaluate some specific scenarios, such as dead-end turns and other common traffic accident scenarios. We need to evaluate some specific metrics, such as safe passage rate and safe distance. Moreover, we currently only detect vehicles and should consider more dynamic objects, such as pedestrians and cyclists. All these enhance-

ments would simulate more future research projects in this exciting direction.

- *Multi-vehicles feature extraction and fusion.* In this thesis, we only consider single-vehicle point clouds for error concealment. We can collect point clouds from nearby vehicles for error concealment in the future. In addition, merging point clouds from multiple LiDAR-equipped CAVs for cooperative perception could reveal more challenging system issues before autonomous driving becomes a reality.

Bibliography

[1] C.-S. Adam. Deep learning with point clouds, 2019. https://news.mit.edu/2019/deep-learning-point-clouds-1021.

[2] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet x.509 public key infrastructure Time-Stamp Protocol (TSP). Technical report, 2001.

[3] A. Akbari, M. Trocan, S. Sanei, and B. AGranado. Joint sparse learning with non-local and local image priors for image error concealment. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(8):2559–2574, 2019.

[4] M. Al-Jarrah, A. Al-Dweik, E. Alsusa, and E. Damiani. RFID reader localization using hard decisions with error concealment. *IEEE Sensors Journal*, 19(17):7534–7542, 2019.

[5] C. Anagnostopoulos, C. Koulamas, A. Lalos, and C. Stylios. Open-source integrated simulation framework for cooperative autonomous vehicles. In *Proc. of IEEE Conference on Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4, 2022.

[6] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee. Redundancy in network traffic: findings and implications. In *Proc. of ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 37–48, 2009.

[7] E. Arnold, M. Dianati, R. Temple, and S. Fallah. Cooperative perception for 3D object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1852–1864, 2020.

[8] V. Banks, K. Plant, and N. Stanton. Driver error or designer error: Using the perceptual cycle model to explore the circumstances surrounding the fatal Tesla crash on 7th May 2016. *Safety Science*, 108:278–285, 2018.

[9] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences.

In *Proc. of IEEE/CVF Conference on Computer Vision (ICCV)*, pages 9296–9306, 2019.

[10] Z. Cai, J. Liang, K. Hou, and S. Liu. LiDAR point cloud image interpolation via separable convolution. In *Proc. of IEEE Conference on Chinese Control Conference (CCC)*, pages 6709–6713, 2022.

[11] L. Cenkeramaddi, J. Bhatia, A. Jha, S. Vishkarma, and J. Soumya. A survey on sensors for autonomous systems. In *Proc. of IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1182–1187, 2020.

[12] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu. F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds. In *Proc. of ACM/IEEE Symposium on Edge Computing (SEC)*, pages 88–100, 2019.

[13] Q. Chen, S. Tang, Q. Yang, and S. Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3D point clouds. In *Proc. of IEEE Conference on Distributed Computing Systems (ICDCS)*, pages 514–524, 2019.

[14] Q. Chen, S. Vora, and O. Beijbom. PolarStream: Streaming LiDAR object detection and segmentation with polar pillars. *arXiv preprint arXiv:2106.HZCY+2007545*, 2021.

[15] S. Chen, J. Hu, Y. Shi, L. Zhao, and W. Li. A vision of C-V2X: Technologies, field testing, and challenges with chinese development. *IEEE Internet of Things Journal*, 7(5):3872–3881, 2020.

[16] X. Chen, S. Shi, B. Zhu, K. Cheung, H. Xu, and H. Li. MPPNet: Multi-frame feature intertwining with proxy points for 3D temporal object detection. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 680–697, 2022.

[17] J. Claybrook and S. Kildare. Autonomous vehicles: No driver...no regulation?, 2018. https://www.science.org/doi/full/10.1126/science.aau2715.

[18] G. Cui, W. Zhang, Y. Xiao, L. Yao, and Z. Fang. Cooperative perception technology of autonomous driving in the internet of vehicles environment: A review. *Sensors*, 22(15):5535, 2022.

[19] J. Cui, H. Qiu, D. Chen, P. Stone, and Y. Zhu. COOPERNAUT: End-to-end driving with cooperative perception for networked vehicles. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17252–17262, 2022.

[20] Q. Delooz and A. Festag. Network load adaptation for collective perception in V2X communications. In *Proc. of IEEE Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–6, 2019.

[21] Q. Delooz, R. Riebl, A. Festag, and A. Vinel. Design and performance of congestion-aware collective perception. In *Proc. of IEEE Conference on Vehicular Networking (VNC)*, pages 1–8, 2020.

[22] S. Demetriou, P. Jain, and K.-H. Kim. CoDrive: Improving automobile positioning via collaborative driving. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, pages 72–80, 2018.

[23] E. Dierikx, A. Wallin, T. Fordell, J. Myyry, P. Koponen, M. Merimaa, T. Pinkert, J. Koelemeij, H. Peek, and R. Smets. White rabbit precision time protocol on long-distance fiber links. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 63(7):945–952, 2016.

[24] C. Domke and Q. Potts. LiDARs for self-driving vehicles: a technological arms race, 2020. https://www.automotiveworld.com/articles/lidars-for-self-driving-vehicles-a-technological-arms-race.

[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. of PMLR Conference on Robot Learning (CoRL)*, pages 1–16, 2017.

[26] M. Drago, T. Zugno, M. Polese, M. Giordani, and M. Zorzi. MilliCar: An ns-3 module for mmWave NR V2X networks. In *Proc. of ACM Workshop on NS-3 (WNS3)*, pages 9–16, 2020.

[27] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[28] D. Frossard, S. Suo, S. Casas, J. Tu, R. Hu, and R. Urtasun. StrObe: Streaming object detection from LiDAR packets. *arXiv preprint arXiv:2011.06425*, 2020.

[29] S. Gao, K. Yang, H. Shi, K. Wang, and J. Bai. Review on panoramic imaging and its applications in scene understanding. *IEEE Transactions on Instrumentation and Measurement*, 71:1–34, 2022.

[30] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[31] C. Glennie. Calibration and kinematic analysis of the velodyne HDL-64E S2 LiDAR sensor. *Photogrammetric Engineering & Remote Sensing*, 78(4):339–347, 2012.

[32] Google. Draco, 2023. https://github.com/google/draco.

[33] T. Hackel, N. Savinov, L. Ladicky, J. Wegner, K. Schindler, and M. Pollefeys. Semantic3D.net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*, 2017.

[34] W. Han, Z. Zhang, B. Caine, B. Yang, C. Sprunk, O. Alsharif, J. Ngiam, V. Vasudevan, J. Shlens, and Z. Chen. Streaming object detection for 3-D point clouds. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 423–441, 2020.

[35] T.-K. Hung, I.-C. Huang, S. Cox, W. Ooi, and C.-H. Hsu. Error concealment of dynamic 3D point cloud streaming. In *Proc. of ACM Conference on Multimedia (MM)*, pages 3134–3142, 2022.

[36] B. Ji, X. Zhang, S. Mumtaz, C. Han, C. Li, H. Wen, and D. Wang. Survey on the internet of vehicles: Network architectures and applications. *IEEE Communications Standards Magazine*, 4(1):34–41, 2020.

[37] Y. Jia, R. Mao, Y. Sun, S. Zhou, and Z. Niu. Online V2X scheduling for raw-level cooperative perception. In *Proc. of IEEE Conference on Communications (ICC)*, pages 309–314, 2022.

[38] Y. Jiang, H. Qiu, M. McCartney, G. Sukhatme, M. Gruteser, F. Bai, D. Grimm, and R. Govindan. CARLOC: Precise positioning of automobiles. In *Proc. of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 253–265, 2015.

[39] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[40] J. Kenney. Dedicated Short-Range Communications (DSRC) standards in the united states. *Proc. of IEEE*, 99(7):1162–1182, 2011.

[41] S.-W. Kim, B. Qin, Z. Chong, X. Shen, W. Liu, M. Ang, E. Frazzoli, and D. Rus. Multivehicle cooperative driving using cooperative perception: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):663–680, 2014.

[42] L. Liu, X. Chen, S. Zhu, and P. Tan. CondLaneNet: a top-to-down lane detection framework based on conditional convolution. In *Proc. of IEEE/CVF Conference on Computer Vision (ICCV)*, pages 3773–3782, 2021.

[43] L. Liu, J. Zhang, R. He, Y. Liu, Y. Wang, Y. Tai, D. Luo, C. Wang, J. Li, and F. Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6489–6498, 2020.

[44] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu. Morphing and sampling network for dense point cloud completion. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 11596–11603, 2020.

[45] X. Liu, C. Qi, and L. Guibas. FlowNet3D: Learning scene flow in 3D point clouds. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019.

[46] P. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using SUMO. In *Proc. of IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.

[47] F. Lu, G. Chen, S. Qu, Z. Li, Y. Liu, and A. Knoll. PointINet: Point cloud frame interpolation network. In *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, pages 2251–2259, 2021.

[48] M. Malinverno, F. Raviglione, C. Casetti, C.-F. Chiasserini, J. Mangues-Bafalluy, and M. Requena-Esteso. A multi-stack simulation framework for vehicular applications testing. In *Proc. of ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet)*, pages 17–24, 2020.

[49] E. Marvasti, A. Raftari, A. Marvasti, and Y. Fallah. Bandwidth-adaptive feature sharing for cooperative LIDAR object detection. In *Proc. of IEEE Conference on Connected and Automated Vehicles Symposium (CAVS)*, pages 1–7, 2020.

[50] D. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[51] Nsnam. Network Simulator 3 (NS-3), 2023. https://www.nsnam.org/.

[52] C. Perkins, O. Hodson, and V. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12(5):40–48, 1998.

[53] L. Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[54] P. Pirri, C. Pahl, N. Ioini, and H. Barzegar. Towards cooperative maneuvering simulation: Tools and architecture. In *Proc. of IEEE Conference on Consumer Communications & Networking Conference (CCNC)*, pages 1–6, 2021.

[55] C. Qi, L. Yi, H. Su, and L. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[56] Y. Qian, L. Yu, W. Liu, and A. Hauptmann. ELECTRICITY: An efficient multi-camera vehicle tracking system for intelligent city. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 588–589, 2020.

[57] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan. AVR: Augmented vehicular reality. In *Proc. of ACM Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 81–95, 2018.

[58] C. Qu, S. Shivakumar, W. Liu, and C. Taylor. LLOL: Low-latency odometry for spinning LiDARs. In *Proc. of IEEE Conference on Robotics and Automation (ICRA)*, pages 4149–4155, 2022.

[59] R. Ravindran, M. Santora, and M. Jamali. Multi-object detection and tracking, based on DNN, for autonomous vehicles: A review. *IEEE Sensors Journal*, 21(5):5668–5677, 2020.

[60] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019.

[61] P. Szikora and N. Madarász. Self-driving cars—the human side. In *Proc. of IEEE Conference on Informatics*, pages 383–387, 2017.

[62] J. Tang, S. Li, and P. Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021.

[63] S.-M. Tang, C.-H. Hsu, Z. Tian, and X. Su. An aerodynamic, computer vision, and network simulator for networked drone applications. In *Proc. of ACM Conference on Mobile Computing and Networking (MobiCom)*, page 831–833, 2021.

[64] TCITS ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. *Draft ETSI TS*, 20(2011):448–51, 2011.

[65] G. Thandavarayan, M. Sepulcre, and J. Gozalvez. Analysis of message generation rules for collective perception in connected and automated driving. In *Proc. of IEEE Symposium on Intelligent Vehicles (IV)*, pages 134–139, 2019.

[66] G. Thandavarayan, M. Sepulcre, and J. Gozalvez. Generation of cooperative perception messages for connected and automated vehicles. *IEEE Transactions on Vehicular Technology*, 69(12):16336–16341, 2020.

[67] N. Vanli, M. Sayin, M. Mohaghegh, H. Ozkan, and S. Kozat. Nonlinear regression via incremental decision trees. *Pattern Recognition*, 86:1–13, 2019.

[68] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *Proc. of ICST Conference on Simulation Tools and Techniques (SimuTools)*, 2010.

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[70] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan. Graph attention convolution for point cloud semantic segmentation. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10296–10305, 2019.

[71] X. Wen, T. Li, Z. Han, and Y.-S. Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1939–1948, 2020.

[72] C.-H. Wu, C.-F. Hsu, T.-K. Hung, C. Griwodz, W. Ooi, and C.-H. Hsu. Quantitative comparison of point cloud compression algorithms with PCC Arena. *IEEE Transactions on Multimedia*, pages 1–16, February 2022. Accepted to Appear.

[73] J. Xu, X. Le, C. Chen, and X. Guan. SPINet: self-supervised point cloud frame interpolation network. *Springer Neural Computing and Applications*, 35(14):9951–9960, 2023.

[74] J. Xu, Z. You, X. Le, C. Chen, and X. Guan. HINet: Hierarchical point cloud frame interpolation network. In *Proc. of IEEE Conference on Intelligent Control and Information Processing (ICICIP)*, pages 334–340, 2021.

[75] R. Xu, Y. Guo, X. Han, X. Xia, H. Xiang, and J. Ma. OpenCDA: an open cooperative driving automation framework integrated with co-simulation. In *Proc. of IEEE Conference on Intelligent Transportation Systems Conference (ITSC)*, pages 1155–1162, 2021.

[76] R. Xu, H. Xiang, X. Han, X. Xia, Z. Meng, C.-J. Chen, C. Correa-Jullian, and J. Ma. The OpenCDA open-source ecosystem for cooperative driving automation research. *IEEE Transactions on Intelligent Vehicles*, 8(4):2698–2711, 2023.

[77] J.-R. Xue, J.-W. Fang, and P. Zhang. A survey of scene understanding by event reasoning in autonomous driving. *International Journal of Automation and Computing*, 15(3):249–266, 2018.

[78] X. Yan, H. Yan, J. Wang, H. Du, Z. Wu, D. Xie, S. Pu, and L. Lu. FBNet: Feedback network for point cloud completion. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 676–693, 2022.

[79] Y. Yang, C. Feng, Y. Shen, and D. Tian. FoldingNet: Interpretable unsupervised learning on 3D point clouds. *arXiv preprint arXiv:1712.07262*, 2(3):5, 2017.

[80] D. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6):2140, 2021.

[81] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou. PoinTr: Diverse point cloud completion with geometry-aware transformers. In *Proc. of IEEE/CVF Conference on Computer Vision (ICCV)*, pages 12498–12507, 2021.

[82] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. Guo, F. Qian, and Z. Mao. EMP: edge-assisted multi-vehicle perception. In *Proc. of ACM Conference on Mobile Computing and Networking (MobiCom)*, pages 545–558, 2021.

[83] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh. PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9601–9610, 2020.

[84] L. Zhao, X. Lin, W. Wang, K.-K. Ma, and J. Chen. Rangeinet: Fast LiDAR point cloud temporal interpolation. In *Proc. of IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2584–2588, 2022.

[85] L. Zhao, Z. Zhu, X. Lin, X. Guo, Q. Yin, W. Wang, and J. Chen. RAI-Net: Range-adaptive LiDAR point cloud frame interpolation network. In *Proc. of IEEE Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6, 2021.