UNIVERSITY OF CALIFORNIA
RIVERSIDE

Design and Implementation of Connected Vehicle Dataset & Future Control System Based on Wireless Communication and Sensor Synchronization

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Engineering

by

Zhaoze Sun

June 2024

Acknowledgement

Firstly, thanks my parents and my all families who are supporting me for my two years master study. They sponsor me whatever the fees or the encouragement and give me a lot of power to continue my study at here. And also I want to thank my advisor Dr. Hang Qiu, he provides me a good chance and platform to learn a lot at lab and also give me a lot of supports on finishing the project and thesis writing. Eventually, I want to thank all my colleagues who is leading me and helping me when I was in the most tough time.

Contents

1. **Introduction** 1

1.1 Motivation 1

1.2 Literature Review 2

1.3 Contribution 3

1.4 Outline of Chapter 4

2. **Connected vehicles and future control system based on wireless communication and sensor synchronization** 6

2.1 Problem Definition 6

2.2 Wireless Communication 8

2.2.1 On Board Unit(OBU) 8

2.3 Synchronization 10

2.3.1 Timestamp Synchronization 10

2.3.2 Sensor Calibration 14

2.4 Wire Control(CANbus message interaction mechanism) 17

2.4.1 OpenPilot 19

3. **Hardware & Software** 22

3.1 Cohda Wireless MK6 OBU 22

3.1.1 Configuration 22

3.2 Ouster LiDAR 23

3.2.1 Configuration 23

3.2.2 Highlights 24

3.3 Camera 24

3.3.1 Lucid Triton 5.4 MP Model (IMX490) 24

3.3.2 Configuration 25

3.4 Operating System 25

3.5 Software 25

3.5.1 Robot Operating System 25

3.5.2 Ouster Driver 25

3.5.3 Arena Camera Driver 26

3.5.4 Openpilot 26

3.6 Bench Tower & Testing Vehicle 27

4. **Tests and Results** 28

4.1 Environment Setup 28

4.2 Time Synchronization Processes 31

4.3 Real-World Scenario Data Recording 33

5. **Conclusion and Future Work** 38

5.1 Conclusion 38

5.2 Future Work 38

List of Figures

Figure 2.1. Three kinds of layer structure of time synchronizat- 11

ion system 11

Figure 2.2 NTP Process Decomposition 12

Figure 2.3 Time synchronization between the client and the server 14

Figure 2.4 Proposed Calibration method MDPCalib 17

Figure 2.5 CANbus Single Connection between Component 18

Figure 2.6 CANbus standard 19

Figure 2.7 Software Architecture 21

Figure 3.1 Cohda Wireless MK6 OBU 23

Figure 3.2 80/20 Framing Tower 27

Figure 3.3 2020 Kia Niro EV 27

Figure 4.1 On Board Tower 29

Figure 4.2 Tower mounting 30

Figure 4.3 Chrony Check 31

Figure 4.4 Status of NTP client 32

Figure 4.5 Before Synchronization 32

Figure 4.6 After Synchronization 32

Figure 4.7 Data recording 34

Figure 4.8 ROS db3 Bag 34

Figure 4.10 Other Towers 37

ABSTRACT OF THE THESIS

Design and Implementation of Connected Vehicles Dataset & Future Control System
Based on Wireless Communication and Sensor Synchronization

by

Zhaoze Sun

Master of Science, Graduate Program in Computer Engineering
University of California, Riverside, June 2024
Dr. Hang Qiu, Chairperson

The thesis presents the design and implementation of connected vehicles using vehicle-to-vehicle(V2V) and vehicle-to-infrastructure(V2I) and control system integrating wireless communication and sensor synchronization for the enhanced performance and functionality. Nowadays the repaid development in sensor technology and wireless communication protocols expand new possibilities for developing the intelligent transportation systems capable of improving the road safety and traffic efficiency. The proposed the system uses multi-sensor setup including the camera, LiDAR to gather real-time environmental data around the vehicle and infrastructure. These sensor data are trained by advanced model and algorithm to create a comprehensive perception of vehicle's surrounding. Meanwhile the key to the system's effectiveness is the synchronization of sensor data through the wireless communication. The synchronization mechanism ensure temporal alignment and coherence across sensors minimizing latency and improving the accuracy of perception. Overall the developed system offers the relevant recorded dataset from sensors and theoretical basis for model training, localization stuff meanwhile contributing how to control the vehicle ECU accordingly based on sensor data to improve traffic efficiency, and overall driving experience in the era of autonomous and connected vehicles.

Chapter 1

Introduction

1.1 Motivation

As urbanization continues to grow and road networks become increasingly congested, there is a critical demand for innovative solution that significantly enhance the road safety, traffic efficiency and driving experience. We have to face different challenges associated with the modern transportation system, especially the advanced autonomous and connected vehicle. The traditional driver-dominated approaches to transportation are proving insufficient in the situation of gradually rising number of the road accident. Furthermore with the emergence of new technologies such as artificial intelligence, sensor fusion and wireless communication, there exists a special chance to revolutionize the way vehicles interact with the environment. Hence, this paper aims to find better ways to improve the existing traffic conditions and enhance the interoperability between V2V and V2I based on the synchronization and communication of vehicle-mounted sensors to improve driving safety.

1.2 Literature Review

In Vehicle to Vehicle(V2V) Communication it has emerged as a critical technology for enhancing the safety and enabling the cooperative driving strategies. V2V communication system allow for proactive hazard detection and avoidance, as well as coordination in traffic flow.

The Global Navigation Satellite System(GNSS) and C-V2X are two prominent technologies used in V2V communication systems. The role of multi- sensor synchronization is crucial. It create a coherent and accurate perception of the surrounding environment. Synchronized sensor data enables precise object detection, tracking, and scene understanding, enhancing the vehicle's ability to perceive and interpret its environment accurately. Hardware synchronization using GNSS-supplied clocks is frequently employed for sensors equipped with interfaces for accessing time from GNSS receivers. However, when GNSS signals are weak or unavailable in areas such as urban canyons or GNSS-denied zones, both localization and synchronization accuracy suffer.[5] Accurate calibration of cameras and LiDAR sensors is essential for sensor fusion and perception tasks. Kalibr is a popular tool for camera and LiDAR calibration, offering robust and accurate calibration results. It was generally divided into two parts Zhang et al ' s pioneering work in Target-Based Calibration and Target-Less Calibration.[7] Based on algorithms, researchers can accurately align sensor data in the 3D world coordinate system, enabling seamless integration and fusion of information from different sensor modalities.

In wire control part, there are some control algorithms in coverage, such as PID control and model predictive control (MPC)[13], utilize perception data and CANbus feedback to adjust vehicle dynamics and ensure safe and efficient operation. Thus our experimental purpose is to implement the above-mentioned method and optimize the time synchronization and calibration processes to obtain the better performance during the vehicle-to-vehicle(V2V) and vehicle-to-infrastructure(V2I) testing.

1.3 Contribution

This thesis mainly studies the relevant methodologies applied in the field of communication in connected vehicle and the way to do the data interaction in advanced autonomous driving. It can be recognized as one of the resources allocation problems in hardness of time error and sensor synchronization. In this study I am contributing to achieve wireless communication between multiple vehicles, calibrate sensors for precise data collection, and enable intercommunication of onboard data to support perception and localization.

This, in turn, facilitates wire control of the vehicles.In the realm of wireless communication, our objective is to synchronize timestamps across various components of the vehicle, be it communication between vehicles (V2V) or between vehicles and infrastructure (V2I). Inevitably, some degree of error exists in these communications, underscoring the necessity to harmonize timestamps throughout the entire vehicle. To achieve this, we leverage the MK6 On Board Unit as the vehicle's signal transmitter and receiver, utilizing the Network Time Protocol (NTP) to synchronize timestamps between MK6 OBU and laptop. Then we introduce a concept is Peer-to-Peer Communication. Vehicles communicate with each other to exchange time information. This can be achieved using V2V communication protocols, where vehicles broadcast their time information and receive updates from neighboring vehicles. Across the vehicle and thereby mitigate errors in data transmission. Furthermore, between the camera and LiDAR, a relative positional relationship exists. Thus, calibration becomes imperative to unify them under a common coordinate system, thereby augmenting accuracy, reliability, and consistency.

Concurrently, I also engage in automotive control-related tasks, involving the understanding and customization of certain ECU control Python scripts based on openpilot. This includes the utilization of controllers to manage steering, pedals, and other vehicle aspects via CANbus signals. Overall, these endeavors contribute significantly to establishing a robust theoretical foundation for intelligent control within Advanced Driver Assistance Systems (ADAS).

1.4 Outline of Chapter

Chapter 2 covers the theoretical knowledge of main methods of time synchronization, calibration, localization and car controls.

Chapter 3 cover the software and hardware used by experiment. It includes each part from the actual part to the methods we used.

Chapter 4 present the environment setup includes the experimental intersection and onboard tower used to capture the data from vehicle. Experimental data is interpreted as well in this chapter.

Chapter 5 goes over the issues we met during the experience and make conclusion for the whole part then extending for the future works.

Chapter 2

Connected vehicles and future control system based on wireless communication and sensor synchronization

2.1 Problem Definition

In today's era of automotive intelligence, enhancing driving safety and information accuracy is paramount. Vehicle connectivity, particularly real-time data synchronization, plays a crucial role in achieving this goal. In the process of wireless communication, time synchronization is especially critical.

Specifically, each sensor in a vehicle has its own preferred timestamp during initialization.[1] Therefore, it is essential to find a method to unify the clock of various platforms and sensors onto a single time reference. To accomplish this, we introduce

Network Time Protocol (NTP) and Precision Time Protocol (PTP) for timestamp synchronization.

In our vehicle testing section, we utilize the On-Board Unit (OBU) as the onboard communication platform. Simultaneously, we integrate LiDAR and high-definition cameras with the ROS platform for data collection. Thus, synchronization entails ensuring that the system, OBU, LiDAR, and cameras are all synchronized to the same timestamp.

Additionally, data from LiDAR and high-definition cameras is typically used for target detection and environmental perception. To enable effective fusion and processing of this data at the algorithmic level, it is crucial to unify them under the same coordinate system. Calibration ensures the consistency of the data provided by both sensors, thereby enhancing system stability and accuracy. In this work we need to derived a method for calibrating the multiple IMU based on the open source camera/IMU calibration toolbox . The upcoming approach is to incorporate multi- IMUs into a same estimator. It should employs a rigorous mathematical frame work to estimate the intrinsic and extrinsic parameters of all sensors that you use. And allowing for the accurate alignment in common coordinate system. After the data collection work on the data side and achieving data synchronization, from the perspective of autonomous driving, the data needs to be transmitted to the vehicle's system in some form. The challenge is to design and implement a reliable and efficient CANbus communication system and our dataset to control multiple ECUs in a vehicle. The system must ensure accurate data transmission, synchronization among various ECUs (for tasks such as acceleration, braking, and steering wheel control), fault tolerance, and real-time performance under diverse operational conditions.

2.2 Wireless Communication

To achieve autonomy, vehicles need to communicate with each other and the surrounding environment. GNSS is a standard term for satellite navigation systems that provide positioning. This term includes global constellation such as GPS, GLONASS etc. Having access to multiple satellites is necessary to ensure the accuracy all times. The timing data delivered by GNSS is exceptionally precise and accurate. Within a GNSS system, multiple time scales exist, including GNSS time, satellite time, and standard time such as Coordinated Universal Time(UTC). In the satellite time transfer approach, corrections are conveyed to user receivers to account for the offset between GNSS time and UTC.[2]

2.2.1 On Board Unit(OBU)

In the architecture of connected vehicle technology, OBU is a key part of its actual transportation services. It has a huge market and is highly valued by car manufacturers.[3] The on-board unit(OBU)is a critical component in intelligent vehicles, particularly in the context of ITS and connected vehicle. It design and functionality depend on the specific purpose and requirements.

The OBU features communication interfaces to interaction with various external systems and networks. It include different ways of wireless communication functionality such as Dedicated Short- Range Communication(DSRC), Cellular Vehicle-to Everything(C-

V2X), GNSS, Wi-Fi, as well as wired interface like Ethernet Port. OBU may also gather the integrate sensors to gather data, such as GPS receiver for location information and camera or LiDAR for perception. Processing unit: The central processing unit (CPU) or micro controller is the core of the OBU and is responsible for executing algorithms, processing sensor data, and managing communication with external systems. The processing unit may also contain memory for data storage and software execution.

Power Supply: OBUs require a reliable power supply to continuously operate within the vehicle. This may involve connections to the vehicle's electrical system, backup batteries, or other alternative power sources.

Software and Firmware: The OBU runs software and firmware to control its operations, execute algorithms, manage communications protocols, and interact with other vehicle systems. This software may include embedded operating systems, device drivers, communications protocols, and application-specific software modules.

Enclosure and mounting: OBUs are typically mounted in rugged enclosures designed to withstand the harsh automotive environment, including temperature changes, vibration, and exposure to moisture and dust. Mounting options may include dashboard mounting, roof antenna mounting, or integrated solutions within the vehicle electronics architecture. In our research we choose to use MK6 OBU from Cohda Wireless as the embedded platform as on board units.

2.3 Synchronization

Synchronization ensures that data us processed and transmitted in coordinated manner, preserving its integrity and accuracy. Thus in our research we have to face two critical parts one is timestamp and second is sensor coordinate.

2.3.1 Timestamp Synchronization

Time synchronization methods can be categorized into hardware (HW) and software (SW) approaches. Hardware synchronization, often based on GNSS-supplied clocks, is commonly used for sensors equipped with interfaces for obtaining time from GNSS receivers.[4] However, challenges arise in GNSS-denied zones or urban canyons, where poor GNSS signals not only affect localization but also degrade synchronization accuracy and precision.[5]

In the realm of robotics, many projects rely on the Robot Operating System (ROS) software. ROS typically employs a straightforward software time synchronization method, where sensor measurements are timestamped based on the time of message arrival at a ROS-enabled computer. However, this method falls short in terms of accuracy and precision. The timing fluctuation of arrived messages can often exceed several milliseconds, and delays between the true moment of measurements vary across different interfaces. While significant frequency skew and backward-running local clocks can occur, these occurrences do not inherently guarantee the consistency of a real-time system, particularly one sensitive to time and event sequencing. To maintain consistency in such systems, a time synchronization procedure requires a suitable adjustment model for the local clock. This model helps mitigate inconsistencies arising from clock adjustments while the system is operational. These are the sub-layer systems shown below:

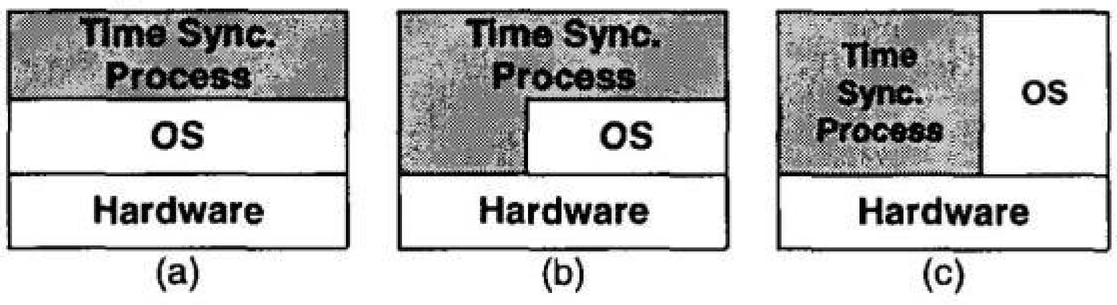


Figure 2.1. Three kinds oflayer structure oftime synchronizat-ion system

The three types of time synchronization layer structures illustrated in Figure 2.1 exhibit different characteristics. Figure2. 1-(a) is relatively easy to implement and more portable, but it poses challenges for optimization. Moreover, its synchronization characteristics rely on OS internals, such as time management and real-time features. If an OS boasts excellent time management functions, Figure2. 1-(a) could serve as a suitable time synchronization method. On the other hand, the time synchronization procedures depicted in Figure2. 1-b) and Figure2. 1-(c) can be optimized with direct access to hardware, yet they are challenging to implement and make portable. [5]

NTP(Network Time Protocol) is a UDP -based application protocol that is widely used to synchronize computer clocks in the global Internet. Using existing general-purpose workstation and Internet technologies, it has been demonstrated that computers can achieve synchronization at the level of several milliseconds in a local area network (LAN) and tens of milliseconds in a wide area network (WAN) environment.

The NTP time synchronization subnet builds a hierarchical tree structure of time servers. At the root of the tree is the master server, which is synchronized to national standards via an external clock. These root servers are called stratum-I servers and can synchronize atomic clocks or GPS (Global Positioning System) receivers. The functionality of the NTP process is broken down in Figure 2.2.

The features of the implementation are summarized as follows: filtering samples from multiple servers through a peer-to-peer process, excluding incorrect servers, and calculating clock corrections based on samples from the selected server and cyclically adjusting the local clock.[5]

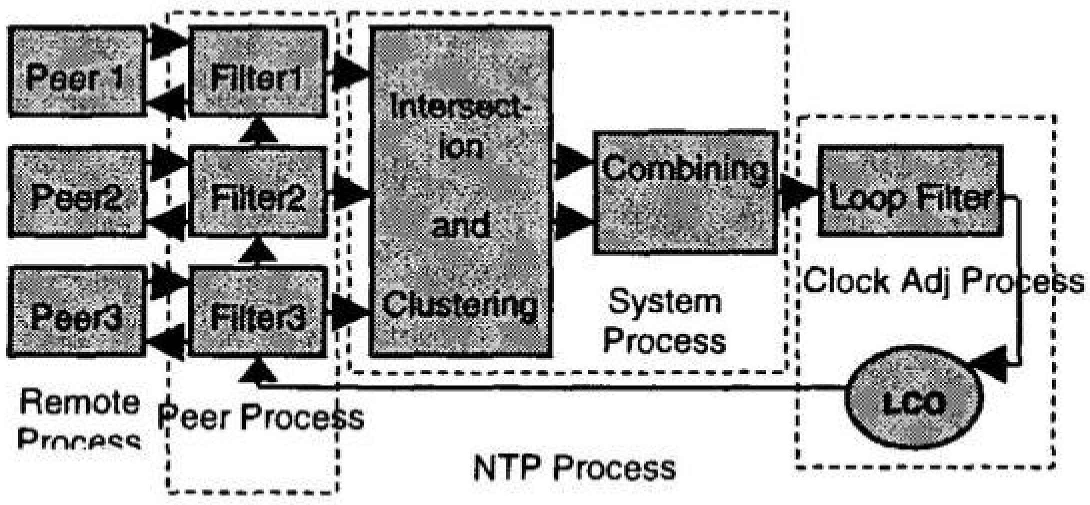


Figure 2.2 NTP Process Decomposition [5]

NTP supports four operating modes: client/server mode, symmetric (peer-to-peer) mode, multicast mode, and broadcast mode. In Client/ Server mode, both client and server maintain a time axis presenting their respective clock times. The process of time synchronization between the client and server shown in Figure 2.3:

1. When the client(Laptop) wants to synchronize itself to the server, it will send an NTP request message to the server(MK6 OBU). This message contains a time when it leaves the client(t1);
2. After receiving the message from client(Laptop), the server(MK6 OBU) will put a record to the message in order to indicate the time in message reached to the server(t2);
3. Afterwards, when server received the message at t2, the message is returned to client. Meanwhile the time it leaves the server is recorded at the time(t3);
4. Finally the client receives the message which contains the all records the time it arrives at the time(t4) [5]

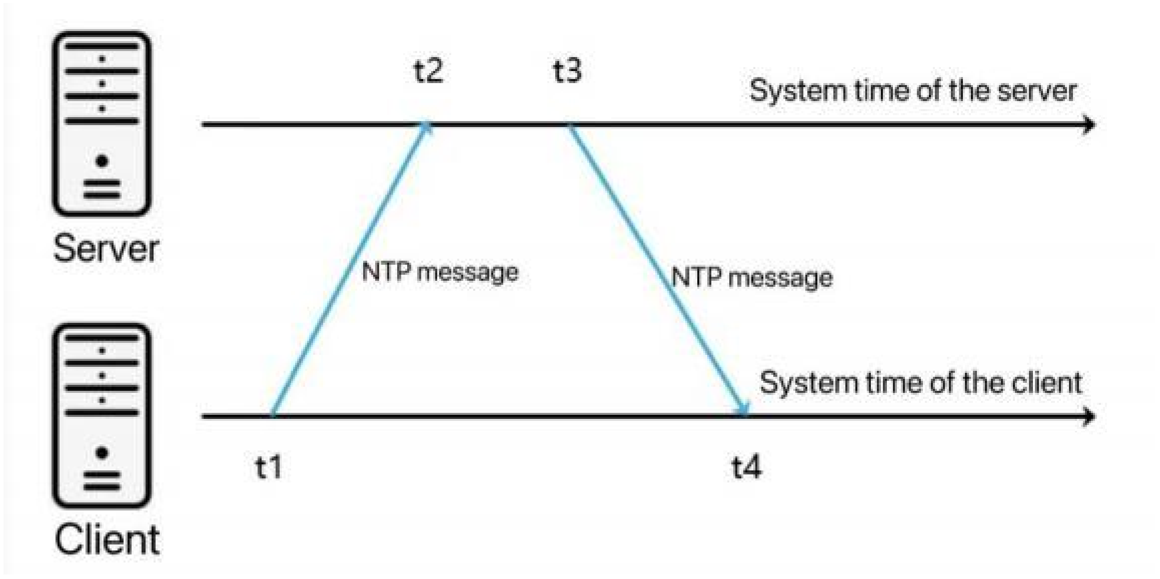


Figure 2.3 Time synchronization between the client and the server[6]

NTP is usually used in scenarios where the clocks of all devices on the network need to be consistent to effectively achieve highly accurate time synchronization. As expected during our experiments, we need to minimize the time error to achieve time synchronization of each unit, so we used NTP.[6]

2.3.2 Sensor Calibration

There may be deviations between different sensors due to manufacturing differences or environmental changes. Calibration allows measurement results to be consistent across sensors, thereby improving system reliability and stability. We employ sensor motion estimates derived from visual and LiDAR odometry, along with deep learning-based 2D-pixel-to-3D-point correspondences obtained without the need for in-domain retraining. The calibration between the camera and LiDAR is formulated as a graph optimization problem, where we minimize the costs arising from constraints imposed by sensor motion and point correspondences.

Method for the camera-LiDAR calibration can literally be divided into target-based and target-less approaches.

Target-Based Calibration: This calibration methods have evolved significantly since Zhang *et al's* pioneering work , which introduced the use of checkerboard pattern for camera-LiDAR extrinsic calibration. Subsequent research has explored diverse pattern styles and techniques to enhance calibration accuracy and applicability·[7] For example, Dhall *et al* utilize ArUco marker with known sizes to accurately estimate pattern corner points in 3D, while Kim *et al* fit camera-detected checkerboard points to corresponding LiDAR plane points.[8] Furthermore, Velas *et al* and Guindel *et al* employed wooden boards with holes to establish points correspondences, demonstrating the diversity of target-based method·[9]

Target-less Calibration: This method aim to perform camera-LiDAR registration without the need for specifically designed target objects, offering flexibility and potentially enabling online calibration for robotics applications. Correspondence-based methods replace artificial targets with patterns perceivable in environments such as urban areas. Tu et al extract features from structure-from-motion(SfM) points and optimize the camera intrinsic poses·[10] Caselitz et al propose a method based on geometric clues fro determining the pose of RGB camera with respect to a 3D point cloud from LiDAR data.

Zhang et al and Yin et al match trajectories from visual and LiDAR odometry and obtain extrinsic parameters through graph optimization, with the latter employing those parameters for initialization of an edge-driven refinement stage·[11]

We view camera-LiDAR calibration as an optimization problem, where the goal is to determine the most likely transformation relationship between camera and LiDAR coordinate systems given a set of sensor measurements. Mathematically, this can be represented by a conditional probability distribution:

$$\mathrm{P(rl\,ro, zo.\,x)}, \tag{1}$$

where x denotes the state vector with an initial x_0 guess and refers to a set of observations, i.e., sensor data. Instead of calculating the exact probability distribution, we perform maximum a posteriori (MAP) estimation assuming Gaussian distributions and independent and identically distributed measurements. This yields the optimal state x^*:

$$\mathrm{ac^{*} \;=arg\,max\,p(ac)zo, x0{:}k)} \tag{2}$$

There is a general approach for camera-LiDAR calibration processtwo inputs of RGB image from camera and 3D point clouds from LiDAR. We comprise a coarse registration based on sensor motion estimated with visual LiDAR odometry·[12] Given the captured initial parameters, a neural network is used to find 2D pixel to 3D point correspondences that result in constrains. The final step is to generate the overall transformation matrix.[7]

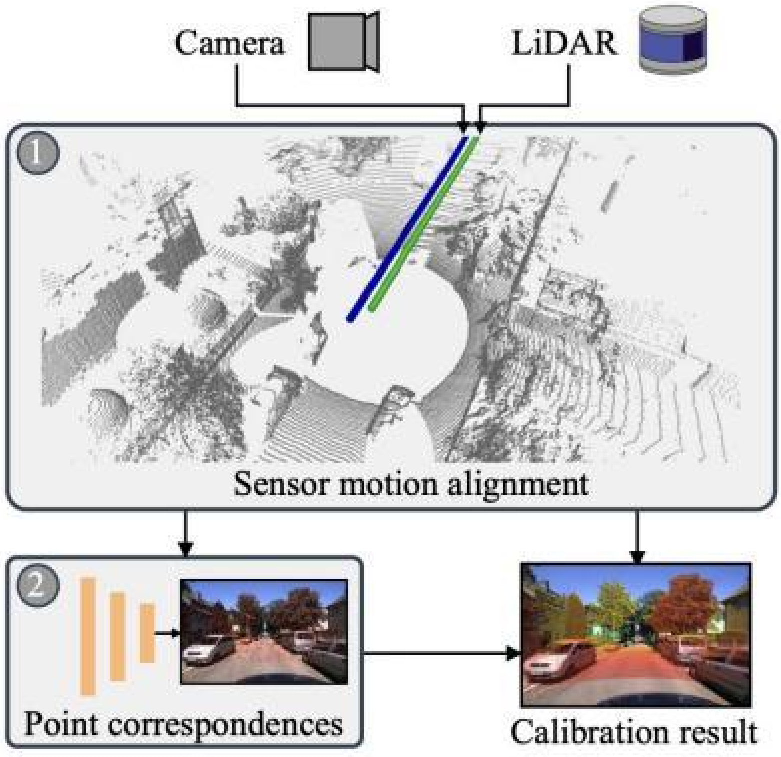


Figure 2.4 Proposed Calibration method MDPCalib[11]

2.4 Wire Control(CANbus message interaction mechanism)

Electrical equipment covers every aspect of a vehicle, from engine control to drivetrain control to driving, braking, steering control, and safety systems and instrument alarm systems. Furthermore, all this work improves the comfort of power management. In addition, with the development of intelligent transportation systems (ITS), the emergence of new electronic communication products (such as 5G) has put forward higher requirements for vehicle control interaction and information sharing.[13]

The modern vehicle contains bunch of different control units like electronically controlled transmission, anti-lock braking system(ABS) and steering wheel control etc.

Each units on requirements of real-time differs from updating rate and control cycle. To facilitate data interaction based on competing model priorities and achieve high communication rates, the CAN bus is specifically designed. Different nodes can be configured in the network based on specific models, forming various CAN network topologies. The diagram below illustrates a typical CAN bus network topology.[13]

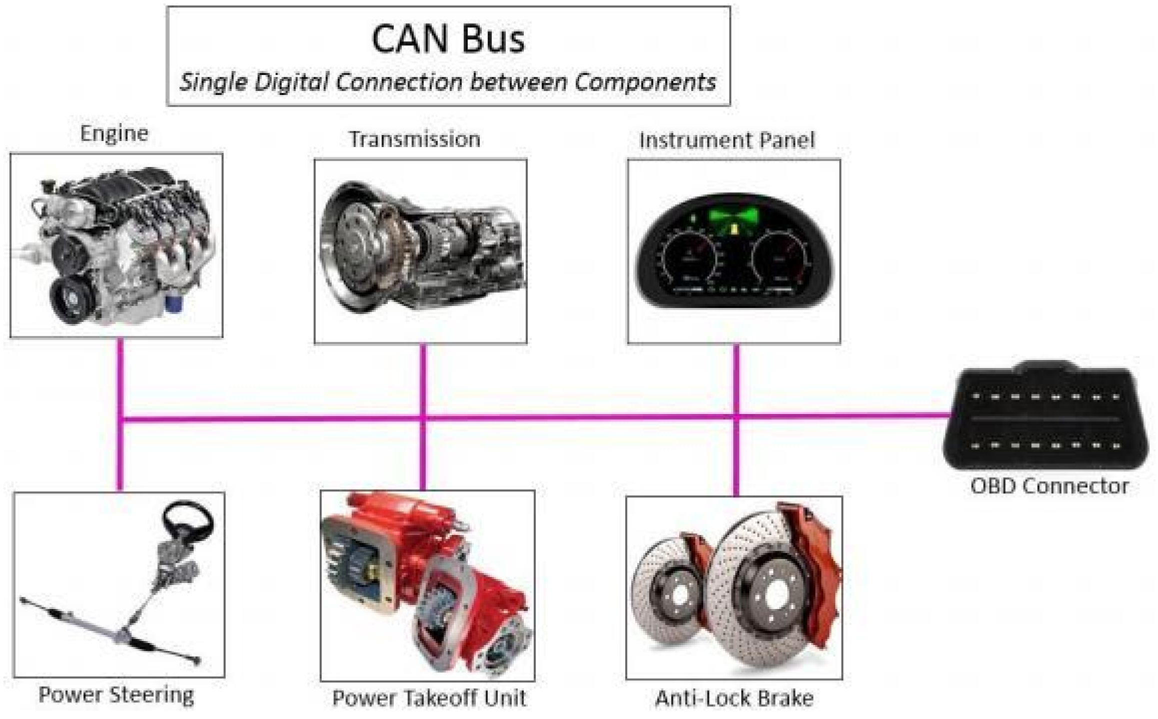


Figure 2.5 CANbus Single Connection between Component

The CANbus system enables ECU to communicate with all other ECUs without complex dedicated wiring. Specifically, an ECU prepare and boradcast information and message (e.g. collecting data from sensor) via the CANbus which include two different wires CAN low and CAN high shown in Figure2.6·[14] The broadcasted data is accepted by all other ECUs on the Can network-and each ECU can then check the data and choose whether to receive or ignore it.

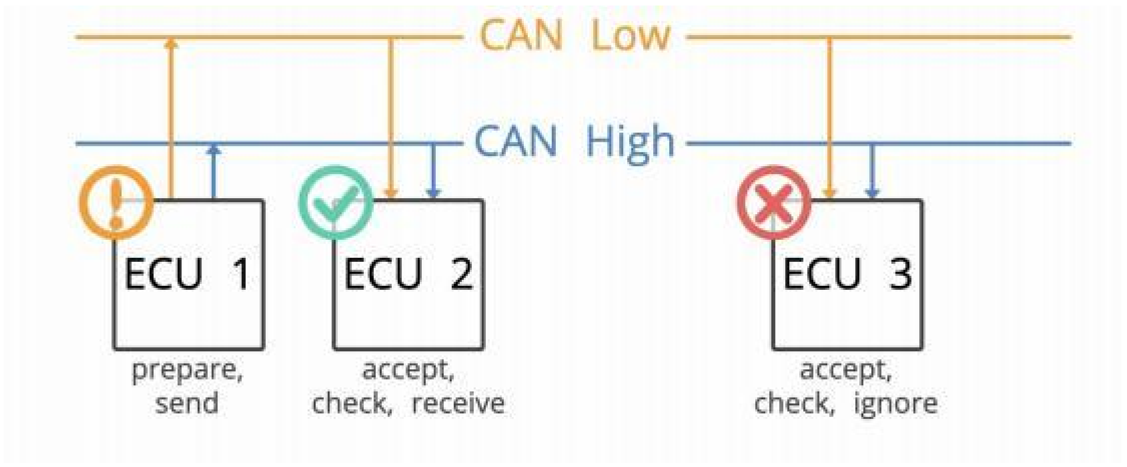


Figure 2.6 CANbus standard[14]

2.4.1 OpenPilot

Openpilot is an open source software platform developed by Comma.ai that provides basic autonomous driving capabilities to compatible vehicles. It allows drivers to experience advanced driving assistance systems such as adaptive cruise control, lane keeping assist and automatic emergency braking, which are implemented with the help of sensors such as cameras, radar and GPS. Due to its open source nature, developers can modify and enhance its code, which makes Openpilot popular in the autonomous driving community because it is cheap, flexible and easy to use.

The platform is a vehicle that utilizes OpenPilot or actuator control and Robot Operating System(ROS) for communication. ROS is used to build and develop robot application that provide the tools, libraries, and conventions. The architecture emphasizes modularity, reusability, and interoperability, helps it comprehensively adopted platform for various application. The integration of Openpilot enables the vehicle to interpret it as the conventional Adaptive Cruise Controil(ACC), Lane Keeping System and Auto Braking System(ABS), simplifying the management of actuators·[15] Additionally, Openpilot furnishes users with the precise data concerning the condition in different parts of vehicle including the acceleration commands, steering torque, vehicle speed, pedals, steering angle and some alert notifications.

In our experiment, our purpose is to customize the open source code of openpilot to obtain the vehicle's speed, corner or some other data, and then convert the data we collect in real time from the OBU into something that can be processed through CANbus. The transmitted information format is then passed to the vehicle's trigger, allowing the vehicle to make some corresponding decisions, whether it is steering or acceleration or deceleration based on changes in the road scenario. The Openpilot code reponsible for sending the acceleration and torque commands was modified to integrate with modules through utilize of ROS. The code was modified to establish two ROS topics. The first topic interfaces with ROS to relay acceleration and torque commands to the vehicle via Openpilot. The second topic disseminates vehicle state data extracted by Openpilot from the CAN network to ROS2.

We source the software architecture from the paper which describe the integration of Openpilot into the vehicle control systems through the ADAS function.

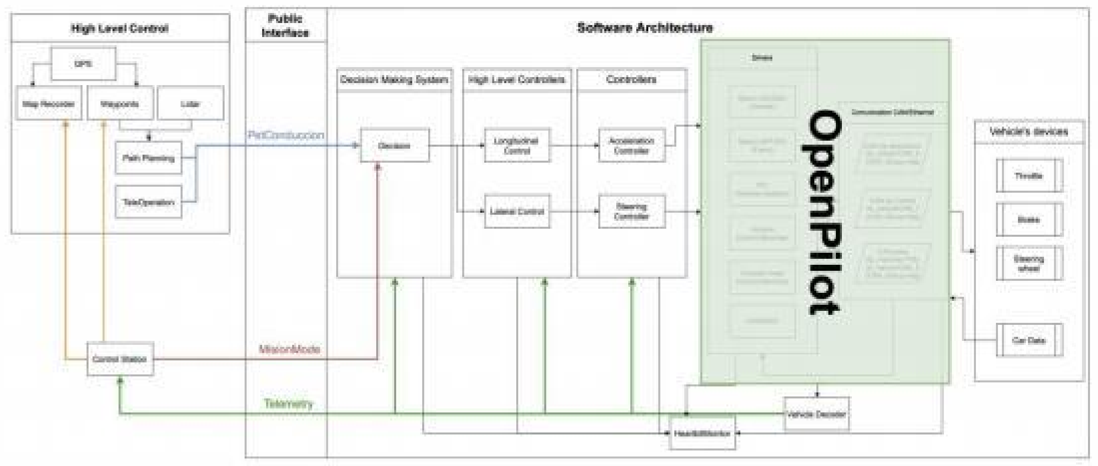


Figure 2. 7 Software Architecture[15]

From viewing the Figure 2.7 it shows the working flow within the integration of Openpilot based on the ADAS. By leveraging ROS to interface with Openpilot, the basic layer of the control architecture have been effectively replaced, yielding a flexible software solution adaptable to diverse vehicles without requiring physical modifications. It provides a developmental framework for signal control software and facilitates integration in the connected vehicles and research of cooperative perception in general.[15]

Chapter 3

Hardware & Software

3.1 Cohda Wireless MK6 OBU

Cohda Wireless 6th-generation On-Board Unit (OBU), is a rugged module that can be retrofitted to vehicles for aftermarket deployment or field trials, and can also serve as a design reference for Automotive production and Smart City deployments. The product is shown in Figure 3.1.

3.1.1 Configuration

Cohda Wireless 6th generation On-Board Unit is the most advanced V2X OBU with dual

- V2X technology using the latest connectivity with:
- DSRC with 2x NXP RoadLink® SAF5400 chipset
- C-V2X with Qualcomm SA515 chipset supporting 3GPP R15, 5G

- Cellular 5G NR with fallback support for 4G (LTE Cat 19)/3G/2G
- Wi-Fi 802.11 a/b/g/n/ac
- Bluetooth v5.1

Cohda Wireless OBU offers a robust, secure foundation for intelligent transport systems of the future no matter the technology choice of C-V2X or DSRC.

Figure 3.1 Cohda Wireless MK6 OBU

3.2 Ouster LiDAR

The mid-range OS1 LiDAR sensor features 90 m range on a dark 10% target, a 45o vertical field of view, and high reliability for the most rugged conditions. The OS1 is designed for all-weather environments and use in industrial automation, autonomous vehicles, mapping, smart infrastructure, and robotics.

3.2.1 Configuration

- Range (80% Lambertian reflectivity, 1024 @ 10 Hz mode) 170 m @ >90% detection probability, 100 klx sunlight
- Rotation Rate: 10Hz
- Field of View: Vertical: 45° (+22.5° to -22.5°) Horizontal: 360°
- Range Resolution: 0. 1cm

3.2.2 Highlights

Calibrated reflectivity

Multi-sensor crosstalk suppression

Open source ROS and C++ drivers

Ouster SDK for software development

3.3 Camera

3.3.1 Lucid Triton 5.4 MP Model (IMX490)

The Triton TRI054S, equipped with the IMX490 (BSI) back-illuminated stacked CMOS sensor, facilitates simultaneous 120 dB HDR imaging and LED flicker mitigation. Boasting a substantial 5.4 MP resolution at 2880 x 1860 pixels, this camera empowers wide-angle shooting in various lighting conditions for Advanced Driver Assistance Systems (ADAS) and Autonomous Driving. Beyond automotive use, it is ideal for applications requiring exceptionally high HDR imaging, including electrical fuse inspection, high-contrast medical imaging, live visible welding analysis, and other Intelligent Transportation Systems (ITS) applications.

3.3.2 Configuration

- Sensor: Sony IMX490 CMOS
- Resolution: 5.4 MP, 2880 x 1860 px
- Frame Rate: 20.8 FPS
- Lens Mount: C-Mount

3.4 Operating System

Ubuntu 20.04 & Ubuntu22.04

3.5 Software

3.5.1 Robot Operating System

ROS(Robot Operating System) facilitates the visualization and recoring of LiDAR(Light Detection and Ranging) data through its ecosystem of package and tools.

ROS Nodes: Responsible for receiving, processing, and transmitting LiDAR data

ROS Messages: Contains each point in the point cloud

RViz: Allow users to visualize sensor data like LiDAR point clouds 3D environment

ROS Bags: Record LiDAR and Camera data stream, it can be played back for analysis

3.5.2 Ouster Driver

This ROS package provide support for all Ouster sensors. Upon launch the driver will configure and connect to the selected sensor device, once connected the driver will handle incoming IMU and LiDAR packets, decode LiDAR frames and publish corresponding ROS messages on the topics of /ouster/imu and /ouster/points. In the case the used sensor supports dual return and it was configured to use this capability, then another topic will published named /ouster/points2 which corresponds to the second point cloud.

3.5.3 Arena Camera Driver

Robot Operating System (ROS) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. This document shows how to install arena_camera, LUCID's ROS driver.

3.5.4 Openpilot

Openpilot is an open source advanced driver assistance system that works on 250+ car models of Toyota, Hyundai, Honda, and many other brands.

Openpilot logs the road-facing cameras, CAN, GPS, IMU, magnetometer, thermal sensors, crashes, and operating system logs. The driver-facing camera is only logged if you explicitly opt-in in settings. The microphone is not recorded.

3.6 Bench Tower & Testing Vehicle

Bench Material: 80/20 Framing T-Slot Aluminum

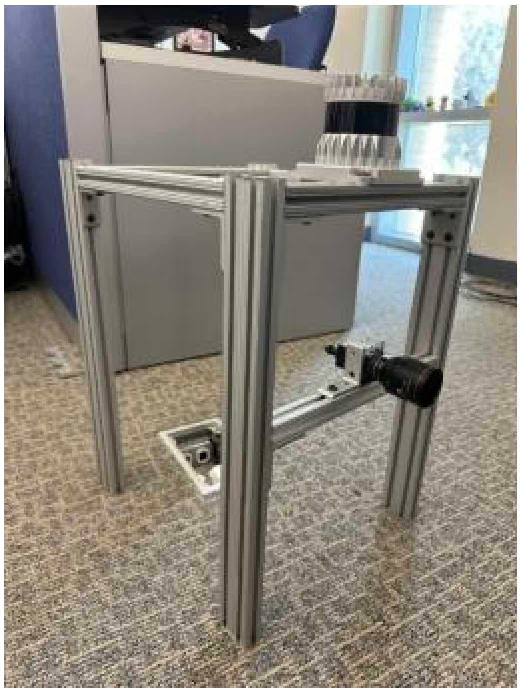


Figure 3.2 80/20 Framing Tower

Testing Vehicle**: 2020 Kia Niro EV**

Figure 3.3 2020 Kia Niro EV

Chapter 4

Tests and Results

In this part we are talking about the testing process, testing result and environment setup. Our test generally divided in to several parts includes how to achieve time synchronization between devices, how to set up the onboard environment of the test vehicle, how to collect actual road data, and how to validate and integrate the data.

4.1 Environment Setup

By using a roof rack and 80/20 T-slot framing, we mounted the cameras, LiDAR, PoE switch, and MK6 OBU on the roof of the Kia Niro. There are two reasons for using the PoE switch. First, the PoE switch can power the cameras, as the Lucid Cameras themselves do not come with a power supply. Second, the PoE switch has a 10G port, which prevents bandwidth contention between sensors during data collection. If we only used a standard Ethernet port (1G), data from the radar and cameras might not be collected completely. Figure 4.1 below shows the completed experimental tower.

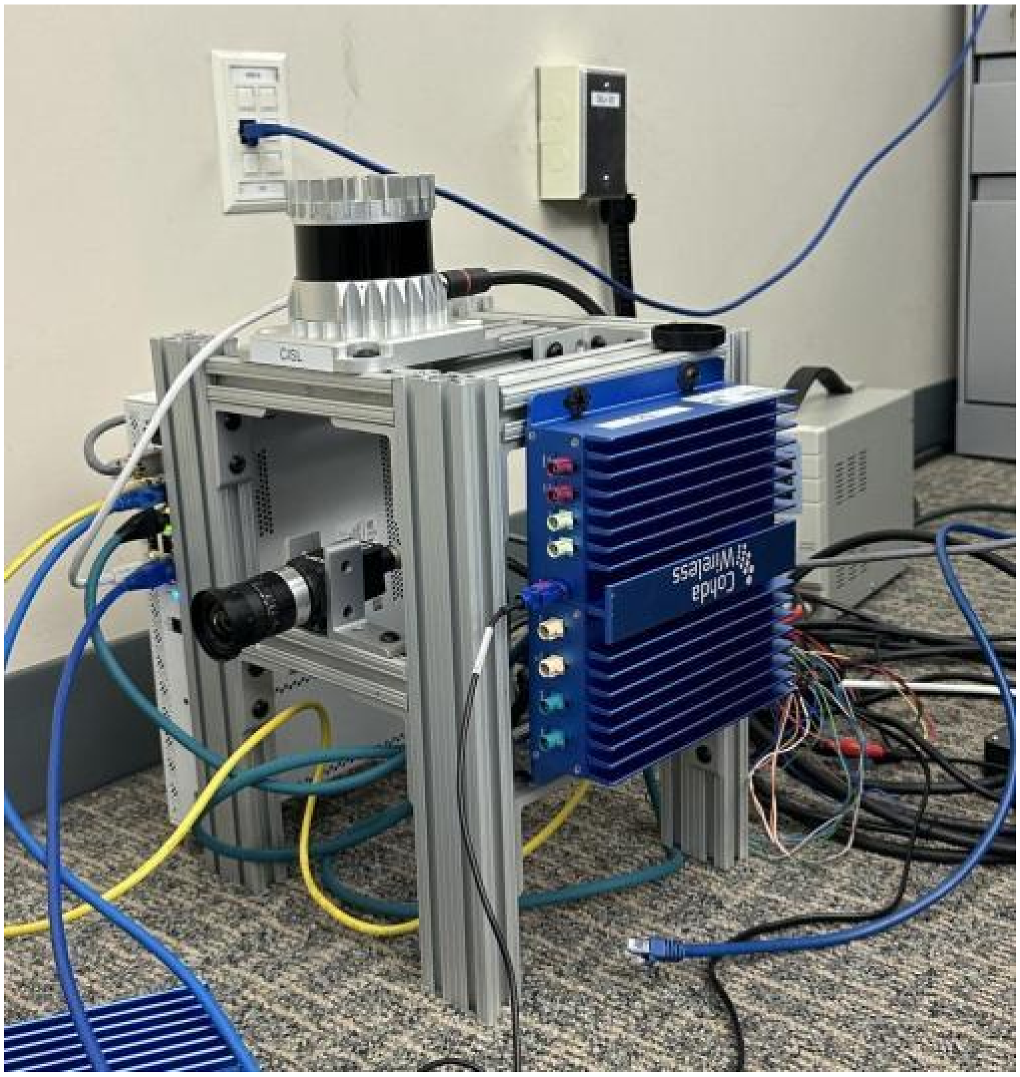


Figure 4.1 On Board Tower

The Figure 4.2 shows the On board tower mounted back to the Kia Niro

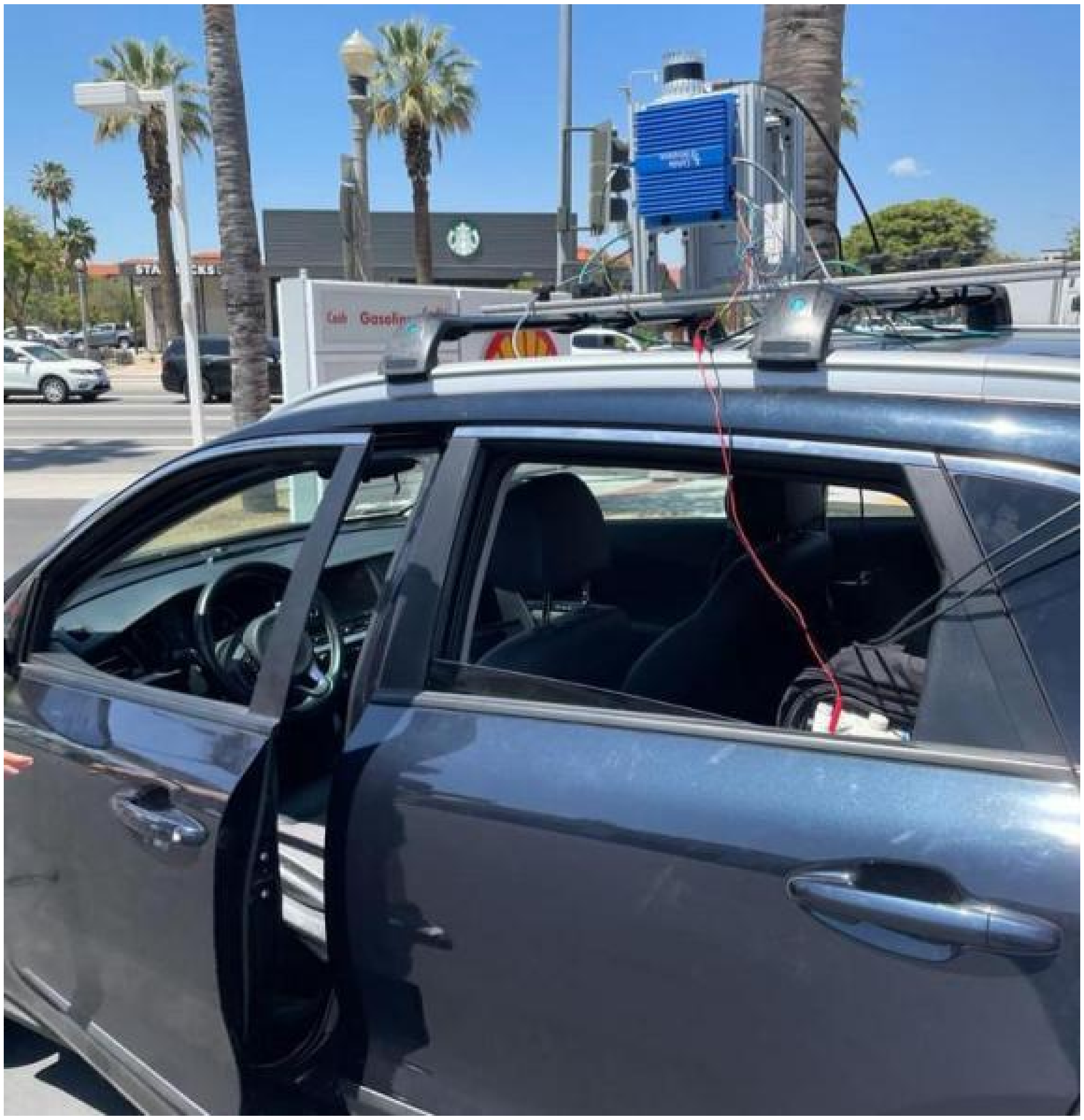


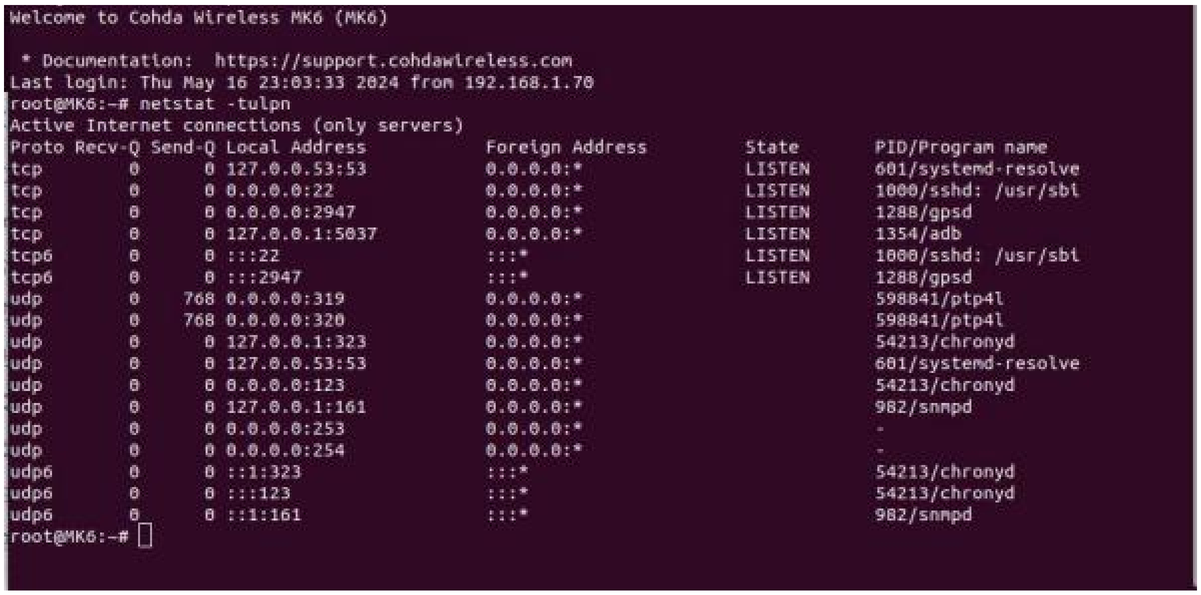
Figure 4.2 Tower mounting

4.2 Time Synchronization Processes

To achieve this, we need to set the MK6 OBU as an NTP Time server and place both the MK6 OBU and the laptops on the same subnet. We set the IP addresses of the two MK6 OBUs to 192.168.1.111 and 192.168.1.80, respectively. The IP addresses of the two laptops are set to 192.168.1.100 and 192.168.1.70. Our goal is to minimize the time discrepancy between the two laptops to reduce the timestamp error between the infrastructure-side and onboard data collection.

Configure GPS synchronization on MK6 OBU and NTP server on MK6

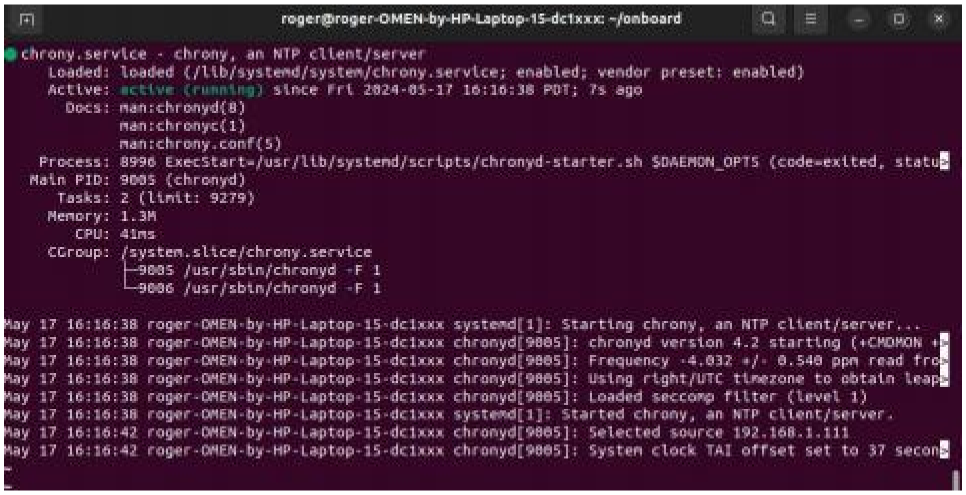
Confirm chrony is listening on udp 123 using



```
Welcome to Cohda Wireless MK6 (MK6)

 * Documentation:  https://support.cohdawireless.com
Last login: Thu May 16 23:03:33 2024 from 192.168.1.70
root@MK6:~# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      601/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1000/sshd: /usr/sbi
tcp        0      0 0.0.0.0:2947            0.0.0.0:*               LISTEN      1288/gpsd
tcp        0      0 127.0.0.1:5037          0.0.0.0:*               LISTEN      1354/adb
tcp6       0      0 :::22                   :::*                    LISTEN      1000/sshd: /usr/sbi
tcp6       0      0 :::2947                 :::*                    LISTEN      1288/gpsd
udp        0    768 0.0.0.0:319             0.0.0.0:*                           598841/ptp4l
udp        0    768 0.0.0.0:320             0.0.0.0:*                           598841/ptp4l
udp        0      0 127.0.0.1:323           0.0.0.0:*                           54213/chronyd
udp        0      0 127.0.0.53:53           0.0.0.0:*                           601/systemd-resolve
udp        0      0 0.0.0.0:123             0.0.0.0:*                           54213/chronyd
udp        0      0 127.0.0.1:161           0.0.0.0:*                           982/snmpd
udp        0      0 0.0.0.0:253             0.0.0.0:*                           -
udp        0      0 0.0.0.0:254             0.0.0.0:*                           -
udp6       0      0 ::1:323                 :::*                                54213/chronyd
udp6       0      0 :::123                  :::*                                54213/chronyd
udp6       0      0 ::1:161                 :::*                                982/snmpd
root@MK6:~# 
```

Figure 4.3 Chrony Check

Configure chrony as an NTP client on Laptop



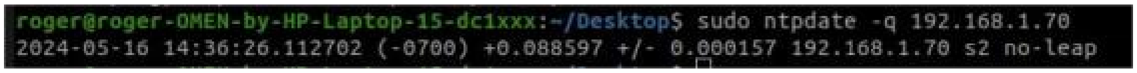
```
roger@roger-OMEN-by-HP-Laptop-15-dc1xxx: ~/onboard
● chrony.service - chrony, an NTP client/server
     Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-05-17 16:16:38 PDT; 7s ago
       Docs: man:chronyd(8)
             man:chronyc(1)
             man:chrony.conf(5)
    Process: 8996 ExecStart=/usr/lib/systemd/scripts/chronyd-starter.sh $DAEMON_OPTS (code=exited, statu>
   Main PID: 9005 (chronyd)
      Tasks: 2 (limit: 9279)
     Memory: 1.3M
        CPU: 41ms
     CGroup: /system.slice/chrony.service
             ├─9005 /usr/sbin/chronyd -F 1
             └─9006 /usr/sbin/chronyd -F 1

May 17 16:16:38 roger-OMEN-by-HP-Laptop-15-dc1xxx systemd[1]: Starting chrony, an NTP client/server...
May 17 16:16:38 roger-OMEN-by-HP-Laptop-15-dc1xxx chronyd[9005]: chronyd version 4.2 starting (+CMDMON +>
May 17 16:16:38 roger-OMEN-by-HP-Laptop-15-dc1xxx chronyd[9005]: Frequency -4.032 +/- 0.540 ppm read fro>
May 17 16:16:38 roger-OMEN-by-HP-Laptop-15-dc1xxx chronyd[9005]: Using right/UTC timezone to obtain leap>
May 17 16:16:38 roger-OMEN-by-HP-Laptop-15-dc1xxx chronyd[9005]: Loaded seccomp filter (level 1)
May 17 16:16:38 roger-OMEN-by-HP-Laptop-15-dc1xxx systemd[1]: Started chrony, an NTP client/server.
May 17 16:16:42 roger-OMEN-by-HP-Laptop-15-dc1xxx chronyd[9005]: Selected source 192.168.1.111
May 17 16:16:42 roger-OMEN-by-HP-Laptop-15-dc1xxx chronyd[9005]: System clock TAI offset set to 37 secon>
```

Figure 4.4 Status ofNTP client

Test the laptop time can be synchronized by NTP server

- Change Laptop time manually to a wrong time
- Restart chrony and run chronyc makestep
- Check if laptop time is corrected by MK6 NTP server

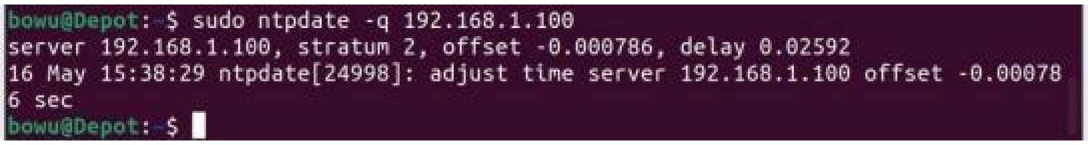
Figure 4. shows the time error before it is synchronized by NTP



```
roger@roger-OMEN-by-HP-Laptop-15-dc1xxx:~/Desktop$ sudo ntpdate -q 192.168.1.70
2024-05-16 14:36:26.112702 (-0700) +0.088597 +/- 0.000157 192.168.1.70 s2 no-leap
```

Figure 4.5 Before Synchronization

Figure 4. shows the time error after using NTP



```
bowu@Depot:~$ sudo ntpdate -q 192.168.1.100
server 192.168.1.100, stratum 2, offset -0.000786, delay 0.02592
16 May 15:38:29 ntpdate[24998]: adjust time server 192.168.1.100 offset -0.00078
6 sec
bowu@Depot:~$ 
```

Figure 4.6 After Synchronization

From the comparison of the two figures, it is evident that the time discrepancy has been significantly reduced from 0.08 seconds to 0.000786 seconds, which is a remarkable improvement and meets our expectations.

4.3 Real World Scenario Data Recording

At the experimental intersection of University Ave and Iowa Ave, we conducted our first data collection involving a test vehicle and LiDAR at the intersection. We equipped the intersection with an Ouster LiDAR and the onboard setup with both an Ouster LiDAR and a Lucid camera.

Using the ROS 2 platform, we collected data separately for the infrastructure (infra) and onboard setups, as shown in the figures. On the infra side, we recorded topics including ouster/imu/points. On the vehicle side, we recorded ouster/points/imu/arena/image_raw. After data collection, ROS bag files in db3 format are generated. We need to ensure that the timestamps of the two bags are synchronized and use Rviz for data visualization. The data visualization includes the camera's image display and the LiDAR's point cloud display, as shown in Figures 4.9(a), 4.9(b), and 4.9(c).

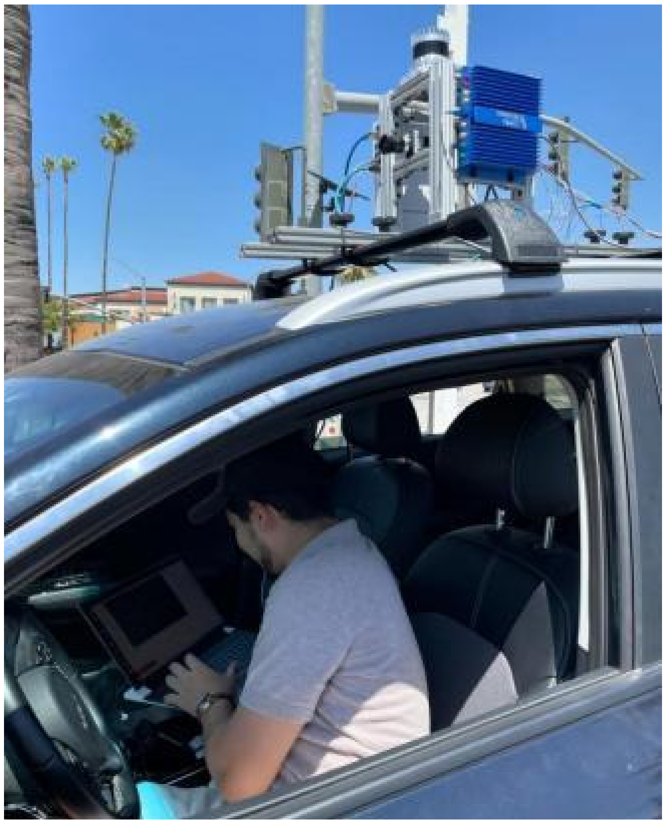


Figure 4. 7 Data recording

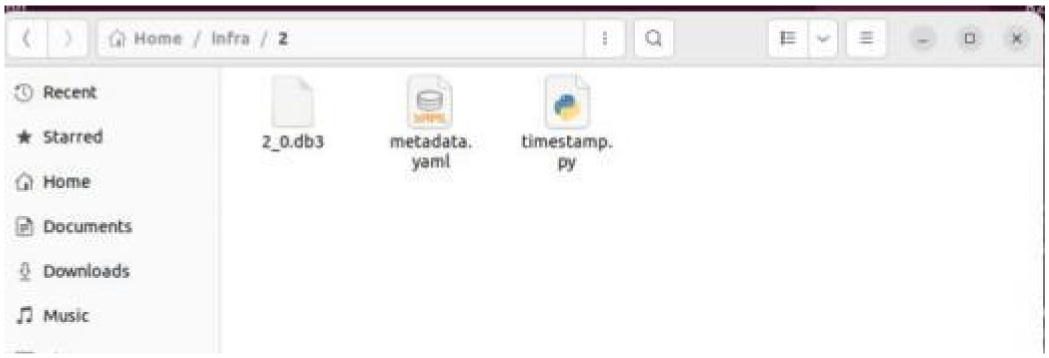
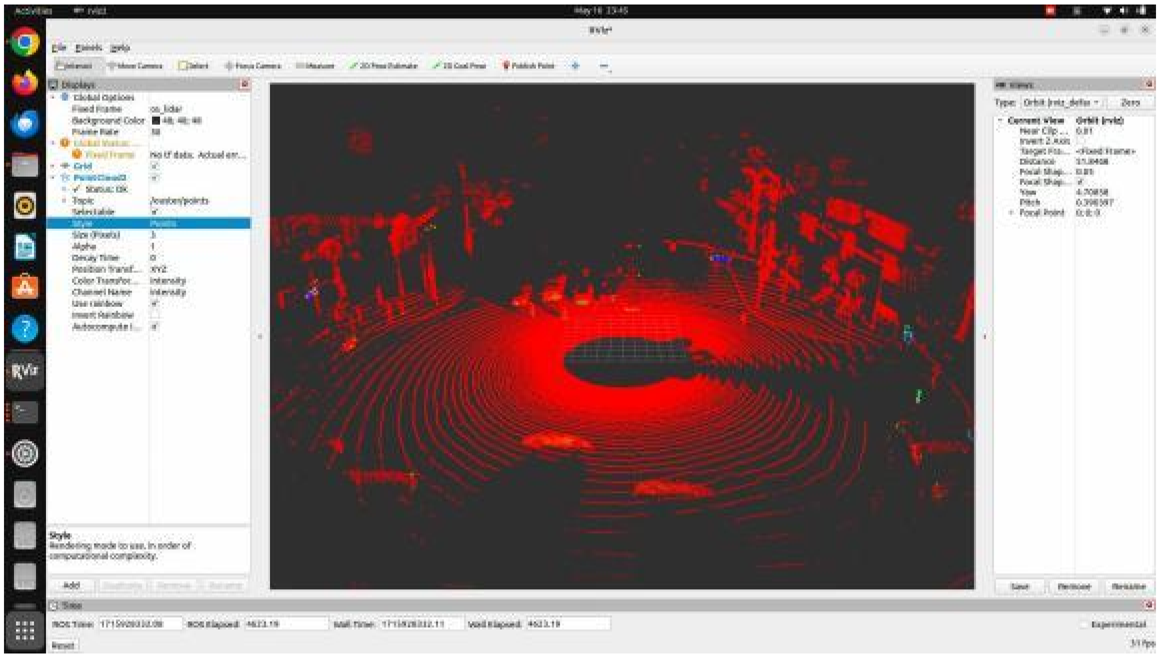


Figure 4.8 ROS db3 Bag



```
Displays
  Global Options
    Fixed Frame            os_lidar
    Background Color       48; 48; 48
    Frame Rate             30
  Global Status: ...
    Fixed Frame            No tf data. Actual err...
  Grid                     ✓
  PointCloud2              ✓
    ✓ Status: Ok
    Topic                  /ouster/points
    Selectable             ✓
    Style                  Points
    Size (Pixels)          3
    Alpha                  1
    Decay Time             0
    Position Transf...     XYZ
    Color Transfor...      Intensity
    Channel Name           intensity
    Use rainbow            ✓
    Invert Rainbow
    Autocompute I...       ✓

Style
Rendering mode to use, in order of computational complexity.

Views
Type: Orbit (rviz_defau
Current View   Orbit (rviz)
  Near Clip ...    0.01
  Invert Z Axis
  Target Fra...    <Fixed Frame>
  Distance         11.8408
  Focal Shap...    0.05
  Focal Shap...    ✓
  Yaw              4.70858
  Pitch            0.398597
  Focal Point      0; 0; 0

ROS Time: 1715926032.08   ROS Elapsed: 4623.19   Wall Time: 1715926032.11   Wall Elapsed: 4623.19
```

Figure 4.9(a) On Board LiDAR Visualization

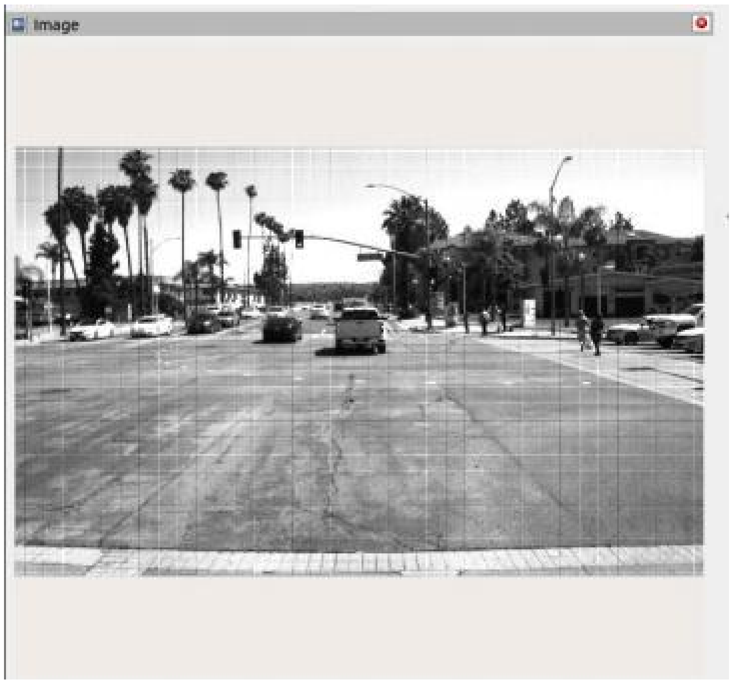


Figure 4.9(b) On Borad Camera Visualization

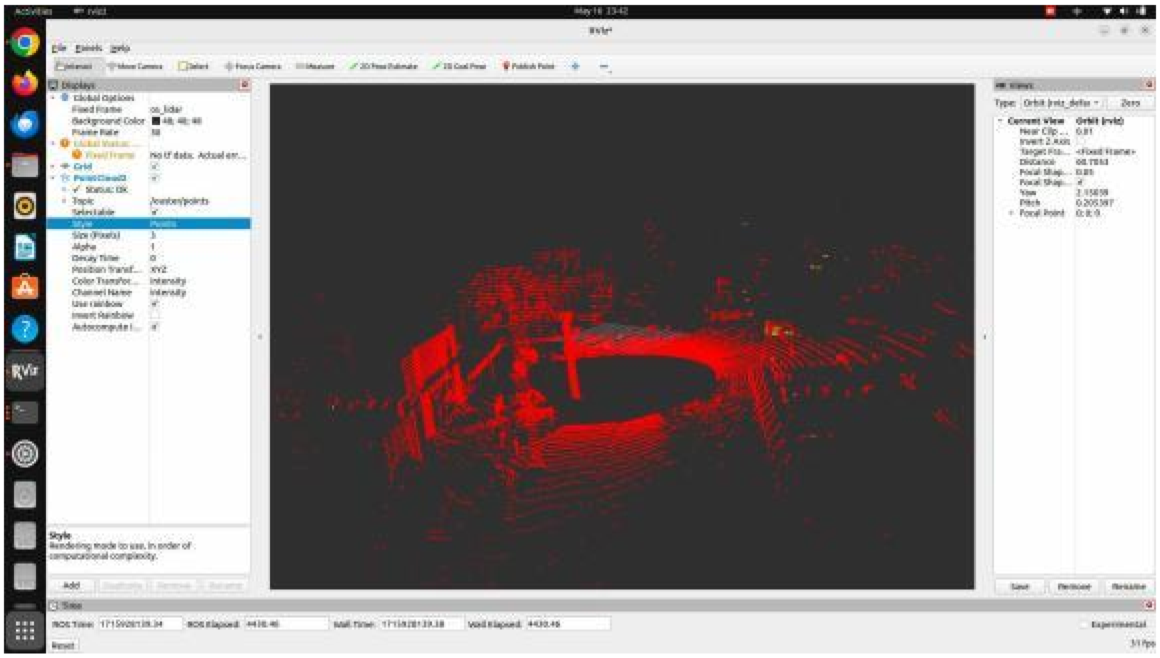

```
Global Options
  Fixed Frame          os_lidar
  Background Color     48; 48; 48
  Frame Rate           30
Global Status: ...
  Fixed Frame          No tf data. Actual err...
Grid                   ✓
PointCloud2            ✓
  Status: Ok
  Topic                /ouster/points
  Selectable           ✓
  Style                Points
  Size (Pixels)        3
  Alpha                1
  Decay Time           0
  Position Transf...   XYZ
  Color Transfor...    Intensity
  Channel Name         intensity
  Use rainbow          ✓
  Invert Rainbow
  Autocompute I...     ✓

Style
Rendering mode to use, in order of computational complexity.

Type: Orbit (rviz_defau     Zero
Current View           Orbit (rviz)
  Near Clip ...        0.01
  Invert Z Axis
  Target Fra...        <Fixed Frame>
  Distance             68.7853
  Focal Shap...        0.05
  Focal Shap...        ✓
  Yaw                  2.15859
  Pitch                0.205397
  Focal Point          0; 0; 0
```

Figure 4.9(c) Infra LiDAR Visualization

Multi- Vehicle Overview

We plan to conduct data collection with two or even three vehicles in the future. We have already installed two additional mounting towers and will follow the same steps mentioned above to install these towers on the other two vehicles, as shown in Figure-4.10. Data Processing and Analysis: Process and analyze the collected data to extract useful information for research and development, such as traffic flow analysis and autonomous driving algorithm training and model training. Meanwhile, The data collection from interconnected vehicles provides a large amount of training data for autonomous driving technology, helping to optimize autonomous driving algorithms, improve the reliability and accuracy of autonomous driving systems, and accelerate the development and adoption of autonomous driving technology.

Data Integration: Integrate and synchronize the data from multiple vehicles to create a unified dataset for subsequent analysis and application like localization and cooperative perception.

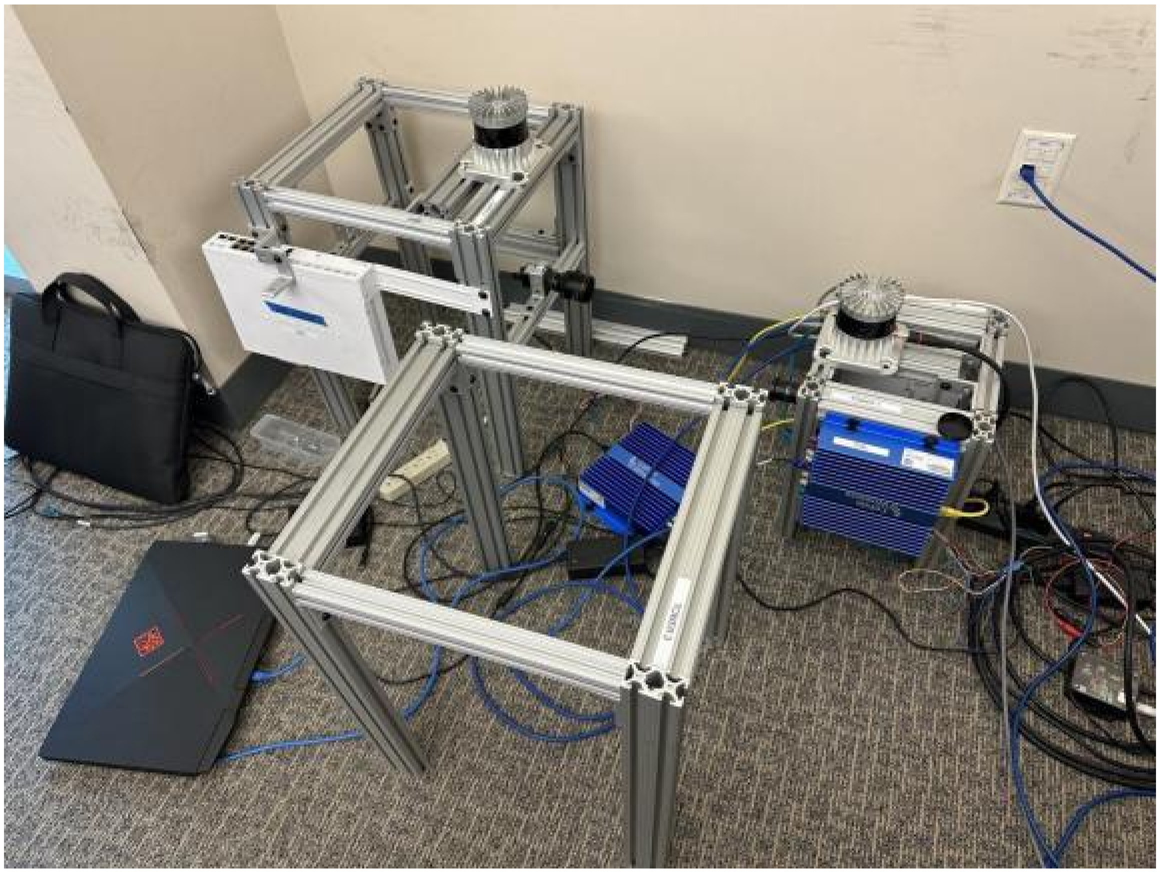


Figure 4.10 Other Towers

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This paper primarily describes the main theories and processes involved in collecting data for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) connectivity. It includes how to achieve time synchronization using NTP server and NTP client methods to synchronize time and data. The paper also introduces the methodology for sensor calibration and elaborates on the essential infrastructure and vehicle-side configurations required for successful vehicle connectivity. Preparatory work is detailed, including the steps necessary before data collection begins. The data collection process focuses mainly on gathering data from one vehicle and the infrastructure's sensors. After data integration, the collected data will support future localization and perception tasks. This, in turn, will aid experiments related to V2V and V2I communication, as well as vehicle control modules.

5.2 Future Work

In future experiments, multi-vehicle data collection will be incorporated to make the dataset more comprehensive. Additionally, in the area of wire control, we will further explore how to control vehicle modules based on the existing dataset. This will aid in achieving more precise cooperative perception.

References

[1] Hasan, Khondokar Fida, Charles Wang, Yanming Feng and Yu-Chu Tian. "*Time synchronization in vehicular ad-hoc networks: A survey ontheory andpractice*." ArXiv abs/1811.04580 (2018): n. pag.

[2] L. Conde, R. Chelim and U. Nunes, "*Collaborative Vehicle Self-Localization Using Multi-GNSS Receivers and V2V/V2I Communications*," 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 2015, pp. 2525-2532, doi: 10.1109/ITSC.2015.406.

[3] Xin Chang, 1Haijian Li , 1Jian Rong, 1Zhufei Huang, 1Xiaoxuan Chen,2and Yunlong Zhang3 *Efects ofon-Board Unit on Driving Behavior in ConnectedVehicle Trafic Flow*

[4] L. Conde, R. Chelim and U. Nunes, "*Collaborative Vehicle Self-Localization Using Multi-GNSS Receivers and V2V/V2I Communications,*" 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 2015, pp. 2525-2532, doi: 10.1109/ITSC.2015.406.

[5] M. Faizullin, A. Kornilova and G. Ferrer, "Open-Source LiDAR Time Synchronization System by Mimicking GNSS-clock," 2022 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Vienna, Austria, 2022, pp. 1-5, doi: 10.1109/ISPCS55791.2022.9918446.

[6] https://magewell.com/blog/87/detail

[7] Q. Zhang and R. Pless, "*Extrinsic calibration ofa camera and laser rangefinder (improves camera calibration),*" in IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems, vol. 3, 2004, pp. 2301–2306.

[8] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "*LiDAR-camera calibration using 3D-3D point correspondences,*" arXiv preprint arXiv:1705.09785, 2017.

[9] Petek, Kürsat, Niclas Vodisch, Johannes Meyer, Daniele Cattaneo, Abhinav Valada and Wolfram Burgard. "*Automatic Target-Less Camera-LiDAR Calibration From Motion and Deep Point Correspondences*." (2024).

[10] D. Tu, B. Wang, H. Cui, Y. Liu, and S. Shen, “Multi-camera-LiDAR auto-calibration by joint structure-from-motion,” in IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems, 2022, pp. 2242–2249.

[11] J. Yin, F. Yan, Y. Liu, and Y. Zhuang, “*Automatic and targetless LiDAR–camera extrinsic calibration using edge alignment*,” Sensors, vol. 23, no. 17, pp. 19 871–19 880, 2023.

[12] Y. Liao, J. Li, S. Kang, Q. Li, G. Zhu, S. Yuan, Z. Dong, and B. Yang, “*SE-Calib: Semantic edge-based LiDAR–camera boresight online calibration in urban scenes,*” IEEE Trans. on Geoscience and Remote Sensing, vol. 61, 2023.

[13] S. Guo, "*The application ofCAN-bus technology in the vehicle,*" 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), Jilin, China, 2011, pp. 755-758, doi: 10.1109/MEC.2011.6025574.

[14] https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial

[15] A. V. Barrio, W. M. Alvarez, C. Olaverri-Monreal and J. E. N. Hernández, "*Development and Validation ofan Open Architecturefor Autonomous Vehicle Control,*" 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA, 2023, pp. 1-6, doi: 10.1109/IV55152.2023.10186551.