Department of Electrical Engineering

FINAL YEAR PROJECT REPORT

BENGEGU4-ECE-2019/20-Project Code

Measurement and Modelling of Driving Attitudes

Student Name: Cheung Wai Hang
Student ID: 55222411
Supervisor: Dr NEKOUEI, Ehsan
Assessor: Prof CHEN, Jie

Bachelor of Engineering (Honours) in

Electronic and Communication Engineering

Student Final Year Project Declaration

I have read the student handbook and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I declare that the work submitted for the final year project does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Student Handbook.

"Measurement and Modelling of Driving Attitudes"

Student Name : Cheung Wai Hang Student ID: 55222411

Signature: Date : 5th - April - 2022

Abstract

Current driving technology includes only “partial automation”, which the driving system takes over a ‘lateral control’, but driver in vehicle is responsible to give a real time control referring to the on-road conditions. This is what the driving attributes right now behave, the stage 3 (*L03*) of total six levels of vehicle autonomy, as defined by Society of Automotive Engineers (SAE) [1].

Before advancing to the final level L05, fully autonomous driving system, teleoperation topology is introduced thanks to latest 5G development. For remote controlled vehicle, the network requires broad coverage, high data throughput and especially, the low latency [2]. To return a nearly real-time on road condition for remote control drivers, delay time attaches most importance because it directly affects the driver’s attributes and thus the remoted car’s movement on road.

As such, experimenting remote control diving with significant latency will discover more detailed drivers’ behaviours and the performance accordingly. These are also the key figures to build a prediction model for the real position of vehicle on road. With an always ‘delayed’ information collected from the vehicle sensors, LSTM modelling is good solution to estimate the real time position of the car.

Special thanks to

Dr NEKOUEI, Ehsan

Contents

I. Introduction

Due to higher transmission rate in this new telecommunication era, precise remote control is an upcoming trend of driving mode. Drivers are more available to manually control vehicle from long distance away. The technical challenge here is carrying the on-road conditions which were drawn from vehicle sensors, to remote driving terminal. So, the transmission time (or delay time) is the analytical point for the whole teleoperation system, if believing that a delay information will lead different driving attitudes of remote-control driver.

In depth, when the latency is significant in the transmission, remote control drivers are mostly compelled to do more adjustments on their manual control, for ensuring vehicle is in normal driving. Moreover, all their figures/graphics received by the vehicle sensors are postponed (due to long distance transmission), drivers might be affected to maintain stable driving performance.

Brief of Delay Time (RTT)

In fact, the whole transmission delay (named Round-Trip Delay) is composed as two-way delay, because of bidirectional feature of remote driving system. The Round-Trip Delay (RTT) in remote driving will be separated as,

1. Transmission time from driver's panel (Td) to vehicle, and
2. Transmission time from vehicle sensors (Tv) signalling back to driver's terminal

Td, is usually describing the time delay of transmitting driver's operation code to vehicle when remote driving. The operation commands conclude all the driver's actions, e.g., the rotation of steering wheel, a fast break from pedal. Or simply, this type of time delay is an operation/command delay in practice. And, later responses from car will be the result of enlarging Td.

Tv, is considered as the transmission time for sending on road image/scene, or any other information from vehicle sensors to the driver's terminal. This delay issues a lag for all collected information. A significant phenomenon is the pictures displayed on driver's monitor, will barely be coherent with the on-road condition, due to Tv. This makes a non-real-time display problem in actual remote driving. Tv can be deemed as a display delay, and it influences the perception of drivers.

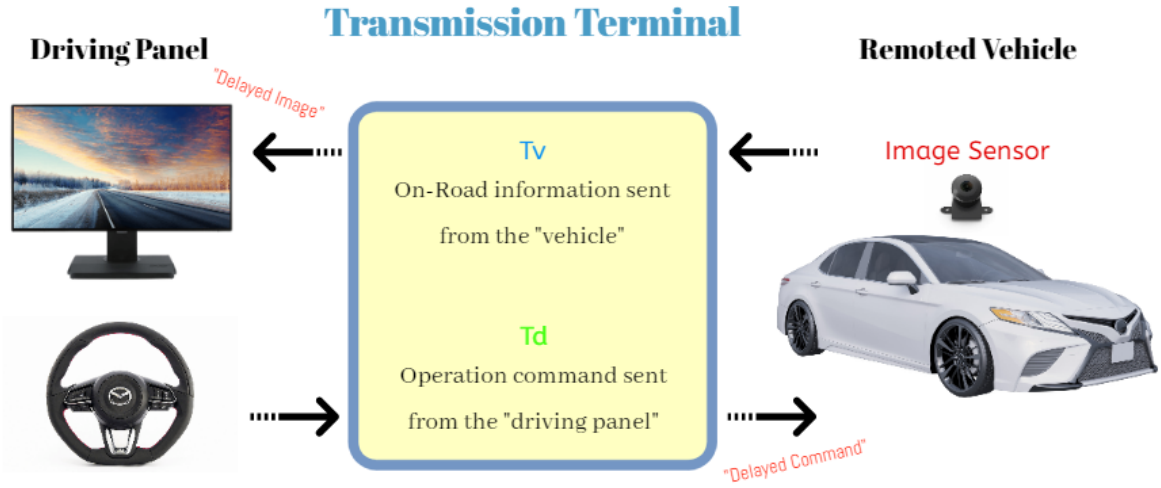

Fig 1.1 Information Transmission Flow of Remote Driving System

Project Aims

Although these two delays might cause problematic control options, it is still uncertain how the drivers behaves and their corresponding reacts. The measurements are needed to analysis drivers' attitudes, for understanding what adjustments people will take under delayed remote driving. It is available to conclude a relatively acceptable range of delay time (it might be negligible, but always exists) that drivers will perform safer driving.

Thus, this session is to conduct an experiment for drivers, with a non-real time remote driving simulation. The experiment setup will insert RTT on purpose, and accordingly, by measuring bunch of driving attributes and data, providing analysis of delayed remote driving behaviour, or performance.

Paper here will also discuss a solution when the information exists latency. The information we received, e.g., geometric location, is uncertain. It is dangerous if the real-time position is doubtful and delayed. Modelling data is workable to estimate the exact real-time location of vehicle. It assists remote control drivers resolve the latency problem of collected information.

<u>Conclusively, two experimental procedures this paper will focus on,</u>

Inviting participants to remote driving and…

1. Study remote control driver's measurements when the RTT is stable but two-way adjustable
2. Build vehicle position predictive model to estimate the exact (real-time) position.

II. Background

Beyond 5G network, LTE transmission network was already investigated the availability to be the teleoperation topology. A research from United State made a similar setup, but replacing with a relatively high and random generated RTT (56ms – 358ms) [3]. The researchers recruited 11 participants to drive a remoted truck (1:10 as actual size) along a fully curved routine (400 meters, as figure 2.1), and thus sampling their attitudes, cross lane errors and duration time.

Fig 2.1 Track of LTE Remote Driving Experiment [3]

Driver's Performance in LTE Network

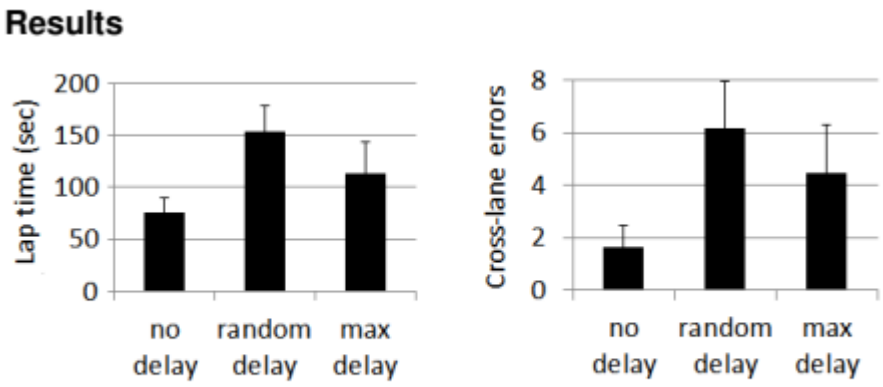

Figure 2. Mean and standard deviation of lap time.

Figure 3. Mean and standard deviation of cross-lane errors.

Fig 2.2 & Fig 2.3 Driver's Results from the Experiment [3]

The random-delay setup seemingly shows worst performance for participants. Drivers averagely spent doubled time, but made tripled errors for going each lap, comparing to no delay driving scenario.

Interestingly, with inserting a stable but largest latency to transmission system, it performs better than random-delay setup (minimizes around 30% of cross lane error). Researchers tends to have a perspective that people perceive uncertainty "whether their driving commands have taken effect" and "where the car really is compared to what is seen through the video" [3]. These two psychological impacts are rooted from "Command and Display Delay" respectively, which have been revealed in advance.

Perspective & Extension

As this research paper prospects progressive telecommunication will achieve lower and stabler delay, to sooth the degradation from RTT. However, here is a temporarily methodology can be suggested to further ensure the "information timing validity".

By implementing multiple cross validation, prediction model can be achieved to generate a best fit for "real-time information". This will be discussed, in using Long Short-Term Memory (LSTM) model, a real-time vehicle position model can be established. Predictive model has its feasibility to tackle with imperfect information.

III. Methodology

i. Experiment Hardware

Desktop OS	Ubuntu 20.04
CPU	Intel® Core™ i7-8700
Graphic Card	GeForce RTX™ 2060 OC 6G
Block Ram Memory	DDR4 3200Mhz * 16GB
External Device	Logitech G29 Driving Force Steering Wheel

List 3.1.1 Desktop Configuration

ii. Carla: Driving Simulation Platform

Simulator Software	Carla v0.9.13
Scripting Language	Python 3.8.10

List 3.2.1 Simulation Platform Configuration

Carla is an open-source autonomous driving simulator, it democratizes autonomous driving R&D, serving as a tool that can be easily accessed and customized by users. CARLA is grounded on Unreal Engine to run the simulation and uses the Open DRIVE standard (1.4 as today) to define roads and urban settings. Control over the simulation is granted through an API handled in Python and C++ that is constantly growing as the project does [4], but here the setups are mainly scripted in Python.

Fig 3.2.1 Schematic View of Carla

A. Sensors

Sensors sample first-hand (real-time) on-road information and parameters of remoted vehicle.

Carla provides several sensors to record various vehicle measurements. Those sensors will be installed with the vehicle but here will just select advantageous sensors to take the measurements.

Below table listed what sensors will be in use.

Camera Sensor[1]	➢ Sensing on-road images ➢ Capturing real-time view of driving perspective
Collision Sensor	➢ Detecting any collisions to external objects
Lane Invasion Sensor	➢ Sensing whether vehicle has changed to a new lane
GNSS Sensor	➢ Reporting current GNSS[2] position, ➢ And speed, rotation axes of vehicle

List 3.2.2 Selected Sensors Description

B. Open Resources

The open resources in Carla allow free usages in map, car model and detailed world settings.

CARLA facilitates different maps for urban settings with control over weather conditions and a blueprint library with a wide set of vehicles to be used. These elements can be customized and new can be generated freely [4].

Below table shows the selected layout for experiment.

Car Model	Tesla Model 3[3]
Weather	Cloudy Noon (Default)
Selected Maps	Town 01 Town 02

[1] Camera Sensor is compulsory for displaying the driving perspective to remote control drivers. The remote driving monitor displays all graphics received by camera.

[2] GNSS Position: Global Navigation Satellite System (GNSS) refers to a constellation of satellites providing signals from space that transmit positioning and timing data to GNSS receivers. The receivers then use this data to determine location [10]**.**

[3] Tesla Model 3 is selected, due to its "Autopilot" system fits the topic of the experiment. The vehicle in simulation will bind to the real physical features of Tesla Model 3, e.g., the Wheel Physics, Engine Parameters, Gear.

	Town 03 Town 06 Town 07
Physical Attributes	All by default

List 3.2.3 Carla Simulation Layout

The characteristics of maps are different, to include variety when collecting samples. All captures of selected maps are in appendix, here is a summary of different maps,

Town 01	A basic town layout consisting of "T junctions".
Town 02	Like Town01, but smaller.
Town 03	The most complex town, with a 5-lane junction, a roundabout, unevenness, a tunnel, and more.
Town 06	Long highways with many highway entrances and exits. It also has a Michigan left.
Town 07[4]	A rural environment with narrow roads, barns and hardly any traffic lights.

List 3.2.4 Selected Maps Description

iii. Scenario Runner

Scenario runner is scriptable, and of two role functions,

1. Define and activate the simulation environment, before conducting experiment,
2. Record wanted measurements, during the whole experiment process.

In other words, setting up a wanted testing environment, scripts to apply those "Open Resources" from Carla is compulsory.

Recall that, two main purposes for this experiment,

…

Inviting participants to remote driving and…

1. *Study remote control driver's measurements when the RTT is stable but two-way adjustable*
2. *Build vehicle position predictive model to estimate the exact (real-time) position.*

Here, due to two different analytical points, the experiment will be divided as two part, and thus two scenario runners to handle with each individual task[5]. Simply speaking, for the

[4] **Town 07 is specially selected for both experimenting driver's performance and position dataset. Will be detailly discussed.**

[5] **However, the general features will always bind to List 3.2.3**

first part of experiment (point 1 of project aims), it is run by ‘Scenario I’; with the second part (point 2 as well), namely ‘Scenario II’.

A. Scenario I

“Study remote control driver’s measurements when the RTT is stable but two-way adjustable”

In this part, the pinpoint is measuring how drivers perform when varying RTT. One main essential criterion should be set as showing the driving qualities of drivers, and to compare their results by referencing this.

a) Δ: Measuring Driving Quality

As such, the experiment is to ask participants (remote control drivers) drive a along a designated route. The parameter for measuring driving quality, is calculating the relative distance between the remoted car and the waypoints of the route. In other words, through observing how far is the car deviated from the waypoints, it shows how precise they close to the route. This difference of position (waypoint and the car[6]) will be the main standard for comparing with each sample, and named as symbol Δ in below.

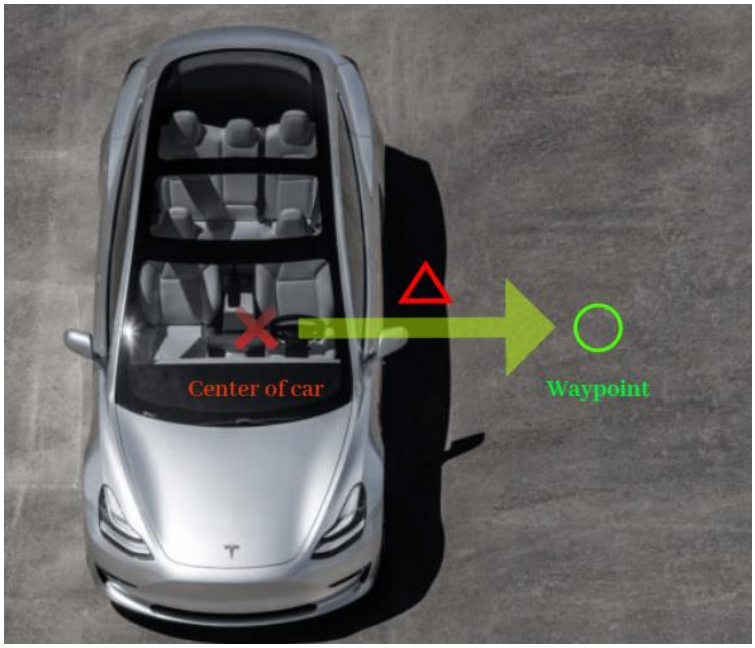

Fig. 3.3a.1 Concept of Δ

[6] **The position of car will be taken at the centre, as Fig. 3.3a.1**

Accordingly, the script defined the algorithm for acquiring all the corresponding Δ when the car passed through each waypoint. Hence, the sampling condition is completely depending on "when the remoted vehicle has passed the waypoint". However, there is no inner-built function from Carla to detect such condition, the scenario runner will loopy run forever until experiment done, and sample the data at a certain interval. But this sampling by time is unwanted, so below algorithm will make conditional sampling effective.

Logic flow:

When the car moves towards the front two points i and i+1, the scenario runner is always taking the measurement of Δ(i), and the Δ(i+1)[7] at each shot time interval[8].

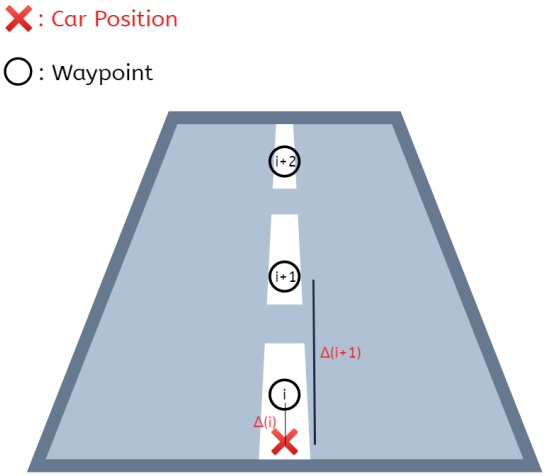

Fig. 3.3a.2 Initiation

Simultaneously, when the remoted car is moving to point i+1, computer system will save all the now and previous Δ(i) into a temporary storage.

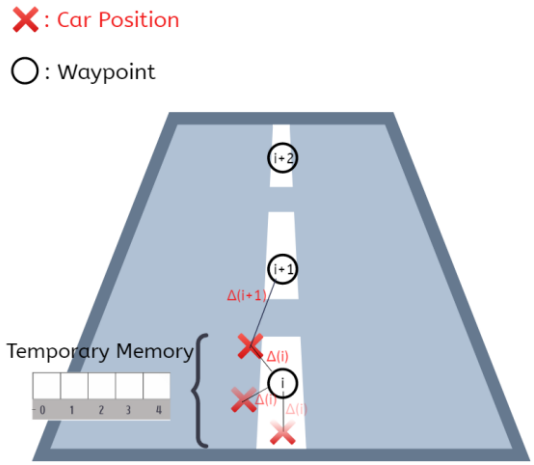

Fig. 3.3a.3 When Moving to Next Point

[7] **The route is designated, so all the waypoint locations are known.**

[8] **The time interval is depending on the clock frequency of computer system, desktop here (See List 3.1.1).**

If system calculates a result of Δ(i+1) is smaller than Δ(i), which means the car has left the point i. The finalized Δ(i) will be known, which is the minimum Δ(i) saved in the temporary storage.

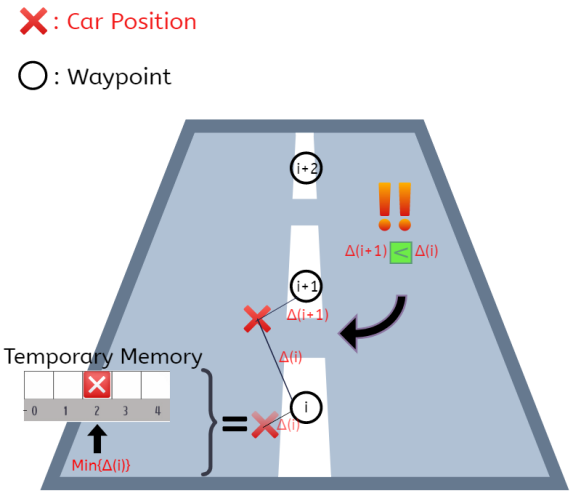

Fig. 3.3a.4 Left the Previous Point

And hence, keep looping the same process for the rest of the points. The algorithm assigns two consecutive points as a pair, to make loopily comparison.

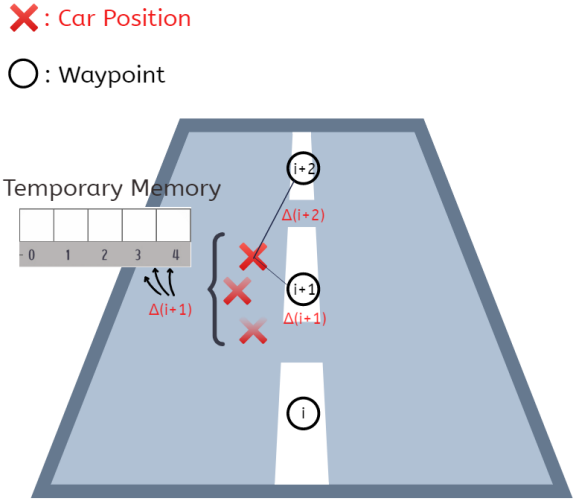

Fig. 3.3a.5 Moving to the Next Point

The above cycle will be end until all waypoints has passed, so actually it is a realization of converting ‘frequency sampling’ to ‘conditional sampling’ in Carla. And note that here is, passing through waypoint is the only condition to facilitate ‘valid’ sampling. But indeed, the temporary storage is sampling at each execute cycle. To be simply, ‘conditional sampling’ is achieved by regularly checking, and finalizes the wanted measurement once the condition meets.

This is the whole logic of sampling Δ in scenario I.

Part of the code below demonstrates the previous process,

```
TempList = []    # a buffer saving temporary Δ
DeltaSet = []    # List saving all the finalized Δ at each point

for i in range(len(routine)):  # routine has 705 waypoints
    while True:   # Loop until waypoints were passed
        if i == len(routine) - 1:
            break
        else:
            # Get the locations of point i and point i+1
            i_point = routine[i][0].transform.location
            ii_point = routine[i + 1][0].transform.location

            # Calculate the Δ(i) and Δ(i+1)
            d_i = get_player_car_distance(i_point.x, i_point.y)
            d_ii = get_player_car_distance(ii_point.x, ii_point.y)

            if d_ii >= d_i:   # Not yet leaving point i
                TempList.append(d_i)    # Save the Δ(i) at now
            elif d_ii < d_i:  # Left point i
                if TempList:
                    # Get back the minimum in the buffer
                    DeltaSet.append(min(TempList))
                    # Jotting down measurements
                    f.write(str(min(TempList)) + " ")
                    f.write(str(time.time() - start_time) + " ")
                    f.write(str(get_player_car_yaw()) + " ")
                    f.write(str(get_player_car_speed()) + "\n")

                    TempList.clear()  # clear the buffer for next
                    break
```

Table 3.3a.1 Code of Sampling Δ

b) Experimental Environment: Selection of Route (Town 07)

Town 07 is chosen to be the driving map in Scenario I. To have more detailed figures from drivers, the driving route should contain complexity and variety.

The route selected as below is a country road, where a rural environment with narrow roads, barns and hardly any traffic lights[9]. It is relatively challengeable to drivers, such that participant is supposed to behave more detailed operations during driving.

Fig. 3.3b.2 Overview of Route

Further, this route has numerous curves and long distance to test the samplers. It totally ends up with 705 waypoints, which the waypoints are allocated with 1 metre distance. The above graph, Fig.3.3b.1 shows the route with all waypoints connected. In a closer view to see the waypoint arrangement, see below,

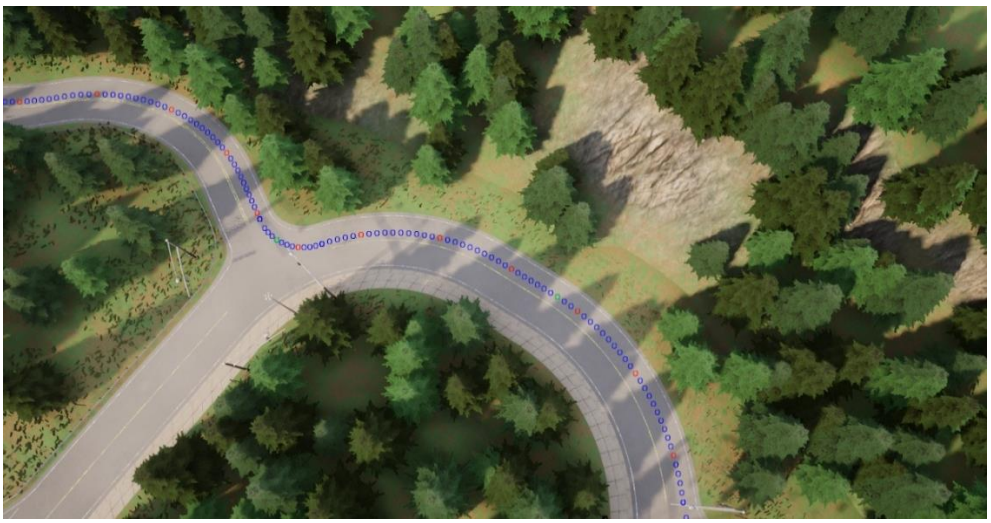

Fig. 3.3b.2 Closer View of Route

[9] **See https://youtu.be/-2Ob0f-k1zQ, with some detailed views in appendix.**

B. Scenario II

"Build vehicle position predictive model to estimate the exact (real-time) position"

Unlike Scenario I, the experiment here is to collect as much vehicle measurements, to estimate the real-time information (car position in this project). In other words, a fixed route is not necessary. Instead, the predictive model acquires huge database to achieve

a) Position Measurements

Here, collecting coordinates measurement of vehicle is the main content of scenario II.

Recall that, GNSS sensor is appended with the remoted car. Sampling geometric location is workable in Carla. However, merely position of car is not informative enough to achieve a reliable predictive model. In scenario II, the yaw and speed of car will be also sampled, though the model is not going to have any estimation on both. More input features make more effective prediction.

Conditional sampling is no longer workable in such case, since no condition is bounding the sample's validity. Instead, sampling information by time is good enough. The recorder set 100ms as the sampling period, sampling that car information when every 100ms passed.

Here is a part of code for better comprehension,

```
def get_player_car_attributes():
    for target in world.get_actors():
        if target.attributes.get('role_name') == 'hero':
            player = target
            velocity = player.get_velocity()
            transform = player.get_transform()
            # Get car Locations
            location_x = player.get_location().x
            location_y = player.get_location().y
            # Get car Speed
            vel_np = np.array([velocity.x, velocity.y, velocity.z])
            pitch = np.deg2rad(transform.rotation.pitch)
            # Get car Yaw
            yaw = np.deg2rad(transform.rotation.yaw)
            orientation = np.array([np.cos(pitch) * np.cos(yaw),
np.cos(pitch) * np.sin(yaw), np.sin(pitch)])
            speed = np.dot(vel_np, orientation)
            # Jotting Down Measurements
            f.write(str(location_x) + " ")
            f.write(str(location_y) + " ")
            f.write(str(round(transform.rotation.yaw, 3)) + " " +
str(round((3.6*speed), 2)) + " ")
```

```
            f.write(str(round(time.time()-start_time, 3)) + "\n")
            break
    return

while True:
    time.sleep(0.1) # Sampling at each 0.1s(100ms)
    get_player_car_attributes()

f.close()
```

Table 3.3b.1 Code of Sampling Car information

b) Experimental Environment: Multiple Maps and Random Route

This part is to measure the car position data under all situations. Thus, there are no boundaries for choosing the driving environment.

Here selecting 5 different maps to do driving experiment, including various roadway characteristics (See List 3.2.4). In addition, the spawn point of vehicle (start point) of the car is randomly assigned, the drivers can drive to any location to be the end point. Termination of sample is decided by drivers.

In order words, it is a free driving simulation conducting on different RTT setups, to record all natural results from remote vehicle.

iv. RTT Setup

In practical 5G network, the RTT is approximately stayed under 50ms [5].

Even though the latency is unstable, 50ms is barely being perceived by human. To have a that low latency, the measurements will be difficult to analysis. By contrast, a relatively larger latency will disclose more human driving attitudes. So, the project here decided,

0ms (Reference Set)	200ms	500ms

List 3.4.1 RTT Selection

Simulation of RTT in python is using below topology.

Logic flow:

Timer is available in Python. By using internal time system, timer interrupt can be implemented.

Assume RTT is 200ms in below case, and delayed 100ms on each terminal

(i.e. Operation Delay = 100ms, Display Delay = 100ms).

First inserting a storage buffer inside, to make First-In-First-Out architecture,

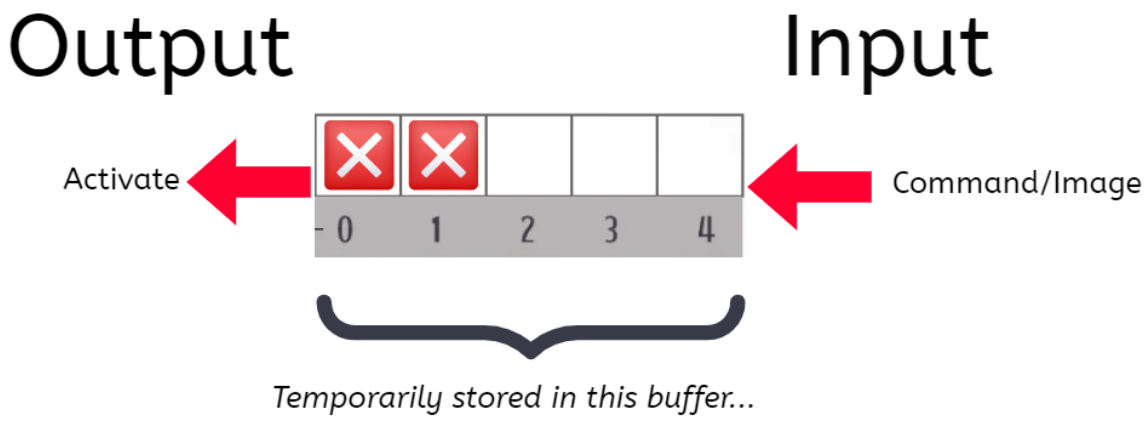

Fig. 3.4.1 Real-Time FIFO Buffer

To insert a latency inside, a time checking system is appended though the output,

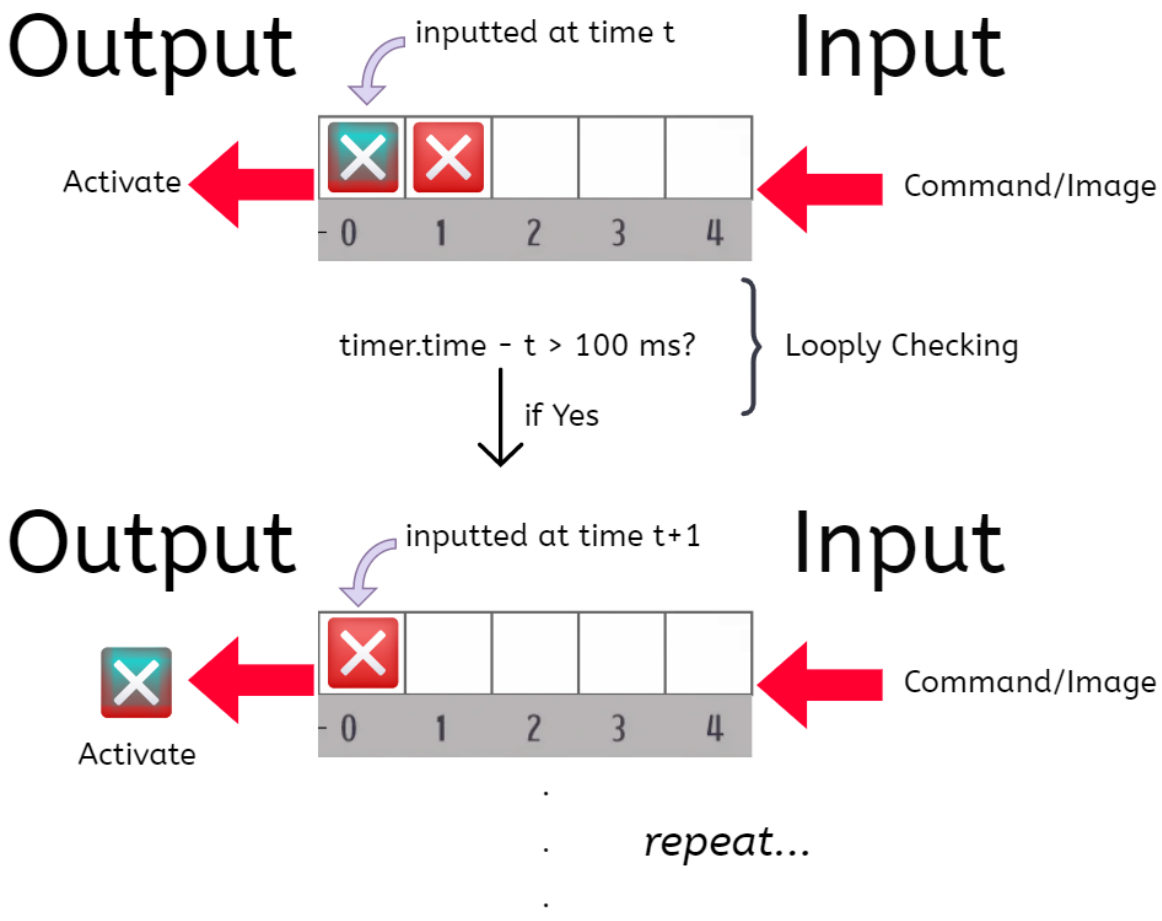

Fig. 3.4.2 Non-Real-Time FIFO Buffer

Saving the coming time of each input and checking whether 100ms was passed or not. If passed, pop the rest elements in buffer, and activate the first element.

This makes a delayed FIFO Buffer.

Part of codes demonstrating RTT setup,

Display Delay

```
pic_buffer = []
with mss() as sct:
    start_time = time.time()
    last_time = time.time()
    while "Screen capturing":

        # Get raw pixels from the screen, save it to a Numpy array
        img = np.array(sct.grab(screen))
        pic_buffer.append(img)

        # 0.01 for 0.1s
        # 0.05 for 0.25s
        if time.time() - start_time > 0.01:
            cv2.imshow("OpenCV/Numpy normal", pic_buffer[0])
            pic_buffer.pop(0)  # Pop out the activated screen from list
        # print("fps: {}".format(1 / (time.time() - last_time)))
        break
```

Operation Delay

```
def parse_events(self, world, clock):
    # Find out what key was pressed
    for e in pygame.event.get():
        # Set a list to save the new coming cammands
        self.event_list.append(e)
        # Set a list to save their new coming time
        self.event_time_list.append(pygame.time.get_ticks())
        pygame.event.clear()

    if self.event_list:
        for event in self.event_list:
            if self.event_time_list:
                # 100 ms Command Delay
                if pygame.time.get_ticks() - self.event_time_list[0] >
100:
                    # Remove the first element and push
                    self.event_time_list.pop(0)

# then Activate
.
.
.
```

Table. 3.4.1 Code of RTT Setup

v. Driver's Panel

The driver's panel connecting Carla's server will realize practical remote driving environment.

A. Driver's UI

Recall that, the camera sensor accompanies with the remoted vehicle (See List 3.2.2) assign the on-road graphics, which is first person perspective. In addition, the steering wheel and pedal (See List 3.1.1) are provided for driver's manual control.

To integrate the graphic processing and steering control, Python Pygame is the applied module[10]. Pygame coordinates the events (steering command) and information flow from Carla (sensors feedback), to provide User Interface for remote control drivers.

Here are screen captures of driver's UI by using Pygame,

Fig. 3.5A.1 Third Person Perspective of Player Car (Dashboard on left)

Dashboard shows important information provided by sensors, e.g., current speed, geometric location, throttle.

Certainly, FFP is the realistic driving screen for remote control driver,

[10] **Pygame is a set of Python modules designed for writing video games. See https://www.pygame.org/news for more.**

Fig. 3.5A.2 First Person Perspective of Player Car (Dashboard hided)

B. Real Experiment Process

Practically, participants will be required to drive along the route in Scenario I (See Fig. 3.3b.1). Until they get familiarity to the mechanics of steering wheel and pedal, and acknowledge to the targeted route.

Experimenters are not allowed to stop the car or drive backward. These two behaviours destroy the continuity of the whole journey, and negatively affects the validity of samples.

Also, they are not notified that what changes were added in each experiment.

a) Scenario I

Sampling Order	1st	2nd	3rd	4th	5th
Display Delay	0ms	0ms	250ms	100ms	250ms
Command Delay	0ms	250ms	0ms	100ms	250ms

List 3.5Ba.1 Sampling Arrangement of Scenario I

The sampling procedures as the above list.

1st : It records the reference set as no delay time inserted.

2nd to 3rd : These are isolation tests on Display/Command delay.

4th to 5th : These are full RTT tests.

b) Scenario II

Drivers will be required to drive through 5 maps, as mentioned. Their given time is set to one and half minute for each map. As such, sampling on five maps as one dataset, and following below sampling order.

Sampling Order	1st	2nd	3rd
Display Delay	0ms	100ms	250ms
Command Delay	0ms	100ms	250ms

List 3.5Ba.2 Sampling Arrangement of Scenario II

For each RTT setup, approximately 5000 (~1000 from each map) samples were received to build the predictive model.

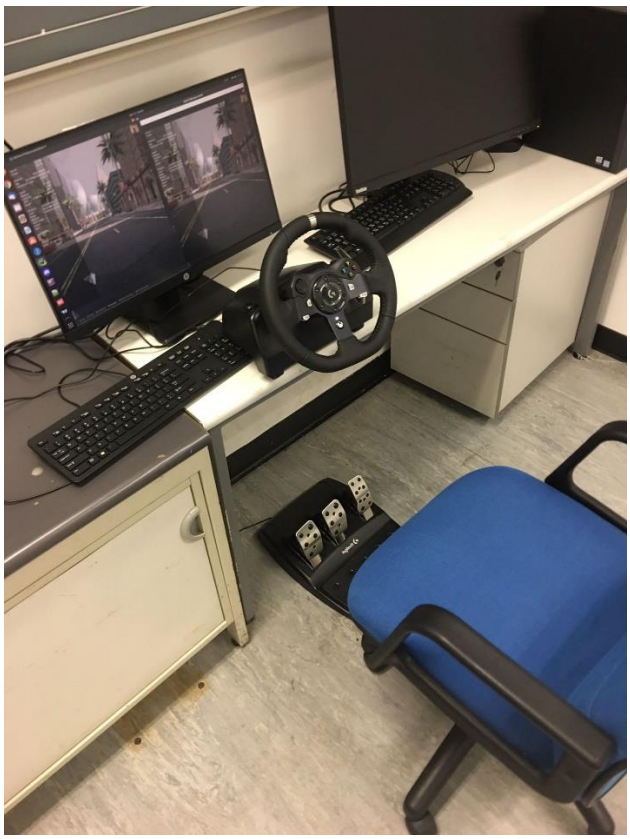

Fig. 3.5B.1 Driver's Setup in Practice

vi. Predictive Modelling: Long Short-Term Memory (LSTM)

In recent years there are many researches put forward various predictive modelling topologies on real vehicle position, mainly the model includes two purposes,

1. Estimate on the future position
2. Filtering out the noise

Long Short-Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. [6] This network is capable of learning long-term dependencies if the training data is sequential and behaves "chain-like" property. And thus, providing prediction on future datum on the input dataset.

The merit of LSTM is the sequence of data will also be taken in learning process. Note that the data of vehicle position is sequential, and specifically for each datum, it relies on the previous state of datum. LSTM model appropriately good at training for this "chain-like" characteristic dataset. It is what LSTM outer performs among other learning models.

Input Features	X (t -1)
	Y (t -1)
	Yaw (t -1)
	Speed (t -1)
Output Features	X (t)
	Y (t)

List 3.5.1 I/O Features of LSTM Model

In this experiment, the "future position" is actually referring to the real-time position of vehicle. All the information received is 'delayed' (at time t-1) due to existence of RTT, by inputting a past position data to predict the real-time location (at time t) of vehicle. Yaw represents the turning direction; speed reflects the changes of vehicle movement. These two features will be also inputted to enhance the predictive analytics.

Yaw and speed are "actively changeable", because these two features are manually manipulated though steering wheel and gas pedal. Both data directly imply the driving attitudes of remote-control drivers. Consider, the geometric position is "passively changeable", because it is affected only by the previous position of car, and the further changes from yaw and speed. This indicates that the changes of position are also reflecting the driver's attitudes. For a dataset of vehicle position, it also carries the driver's attitudes/behaviours inside.

Therefore, LSTM predicts the real time location of vehicle, and most importantly, the long-term dependency learnt by the model (simply the model fitting) is describing the driver's attitudes/behaviours. LSTM analytically learns the tendency from a dataset (including yaw and speed). In fact, this tendency describes the driving pattern of drivers. That is the reason why this LSTM will also model the driving attitudes from drivers, not only the position data.

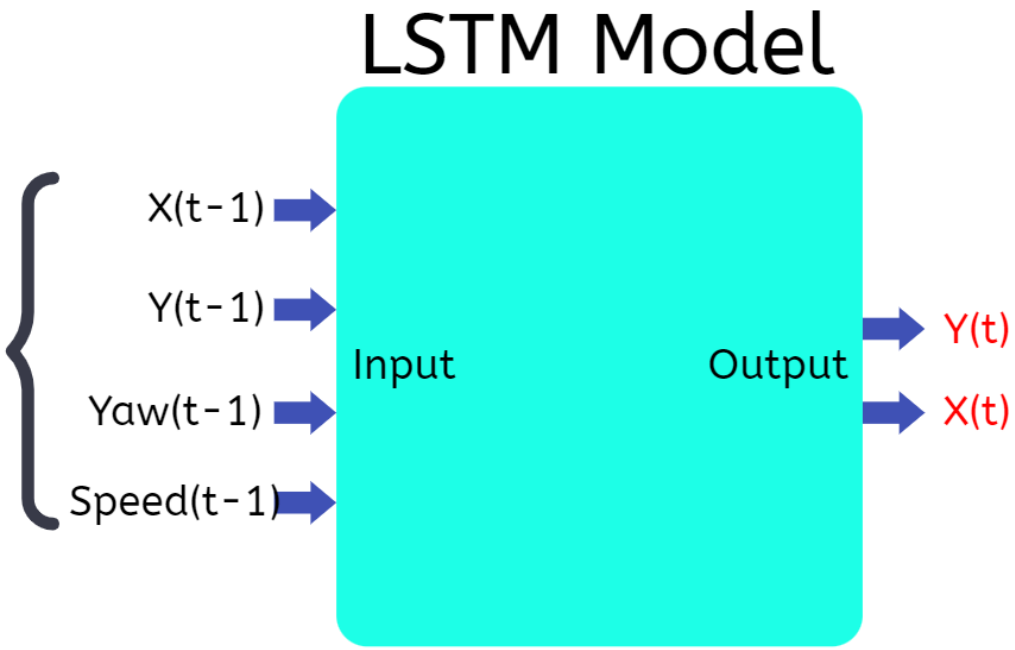

Fig. 3.5.1 Architecture View of the Predictive Model

Detailed settings for this LSTM model,

Number of Neurons	50
Number of Training Epochs	50
Batch Size	64
Learning Rate	0.01
Activation Function	Hyperbolic Tangent (tanh)
LSTM Algorithm Library	Keras

List 3.5.2 Main Parameters Setting of the LSTM

IV. Results

2 participants were invited to drive along the designated route in Scenario I, namely, participant A and participant B.

For each RTT setup in Scenario II, approximately 50000 (~10000 from each map) samples were received to build the predictive model.

i. Driver's Performance varies RTT

Overall, from results, the total error distance (Δ) is directly proportional to the RTT.

RTT	Total Sampled Error Distance Δ	
	Participant A	Participant B
0 ms	254 m	214 m
200 ms	266 m (+ 4.72%)	237 m (+ 10.7%)
500 ms	394 m (+ 55.1%)	281 m (+ 31.3%)

List 4.1.1 Total Δ of Participants A&B

For both experimenters, their percentage rises are slightly when RTT is 200ms, but significant as RTT is 500ms. Although their percentage increases are different, minimum 30% growth of Error Distance Δ will be observed if RTT is 500ms. Similarly, at least 4% of error distance is increased in 200ms RTT setup.

From deeper observation, increasing RTT will also enlarge the coincidence of making large errors. If taking both participants' personal average error as standard value in 0ms case respectively, number of samples exceeding such standard is also proportional to RTT.

Participant	Personal Average Δ (PA) when RTT = 0ms
A	0.362 m
B	0.305 m

List 4.1.2 Average Δ of Participants A&B when RTT is 0ms

RTT	# of Samples Exceed Their PA	
	Participant A	Participant B
0 ms	263	306
200 ms	292 (+ 11.0%)	333 (+ 8.82%)
500 ms	377 (+ 43.3%)	355 (+ 16.0%)

List 4.1.3 Number of Samples exceed PA in all RTT cases

These figures illustrate higher RTT will result more deviated samples. By referencing their 0ms datasets, both participants made more erroneous driving samples. Participant A has minimum 11% increase in 200 ms case, and it generated over 43% increase in 500 ms RTT experiment. Participant B has less increases, added 8.82% in 200 ms and 16% in 500ms of RTT separately.

The maximum error distance behaves differently in various RTT setups.

RTT	Maximum Δ among All Waypoints	
	Participant A	Participant B
0 ms	1.247 m	1.360 m
200 ms	1.217 m (- 0.03)	1.078 m (- 0.282)
500 ms	2.344 m (+ 1.097)	1.747 m (+ 0.389)

List 4.1.4 Maximum Δ for all RTT

Interestingly, when RTT is 200ms, the maximum Δs vary relatively little comparing to the case of 0 ms, but both were enlarged significantly under 500ms RTT. Especially driver A increased 88% of maximum Δ in 500 ms RTT circumstance.

Conclusively, participant B has relatively lower errors under the above standards. However, the adverse effect from increasing RTT is still observable.

More detailed graphs will be summed up in appendix.

ii. Isolation Test on Two Delay Types

Command Delay influences more to drivers seemingly from result.

The isolation test was conducted by setting another delay as 0ms, for observing driving attributes specifically when only one delay type exists. Here, only a larger delay time 250 milliseconds was selected to test, with the aim of enlarging the experimental results/effects.

Delay Type	Participant	Total Δ	# of samples > PA
Display Delay (250 ms)	A	268	306
	B	262	353
Command Delay (250 ms)	A	322 (+ 20.1%)	313 (+ 2.29%)
	B	277 (+ 5.73%)	417 (+ 18.1%)

List 4.2.2 250ms Isolation Test on Each Delay Type

Overall, both participants reflected more total error distance Δ when attempting isolation test of Command Delay. Participant A was sampled 20% increase; B's result also has 5.7% increase of total Δ, comparing to their results of isolation test on Display Delay. Concurrently, their sampled Δ more frequently hits above their PAs (illustrated in List 4.1.2). Participant A has 2.3% jumping and B has 18% also. This reveals that both participants tend to yield more large Δs under 250ms Command Delay isolation test.

iii. Performance of LSTM Predictive Model

The predictive models' Root Mean Square Errors (RMSE) are summarized below,

RTT	RMSE of x coordinate	RMSE of y coordinate
0 ms	2.382	2.522
200 ms	6.378	5.370
500 ms	6.672	5.898
200ms (Trained by 0ms dataset)	8.912	7.706
200ms (Trained by 0ms dataset)	8.711	6.768

List 4.3.1 RMSE of All LSTM Models

Main characteristic here is the LSTM models accuracy of "zero RTT" is higher than "non-zero RTT". For non-zero RTT prediction, the LSTM model here attaches a minimally doubled root mean square error than zero RTT one.

Interestingly, when applying 0ms training set to predict the non-zero RTT data, the RMSE will be even higher. As the red figures highlighted, there are at least 15% increased errors if applying zero RTT dataset as training set. The performance will be worse if the training set and testing set are of different RTTs.

Here is a comparison chart to see the performance of the model,

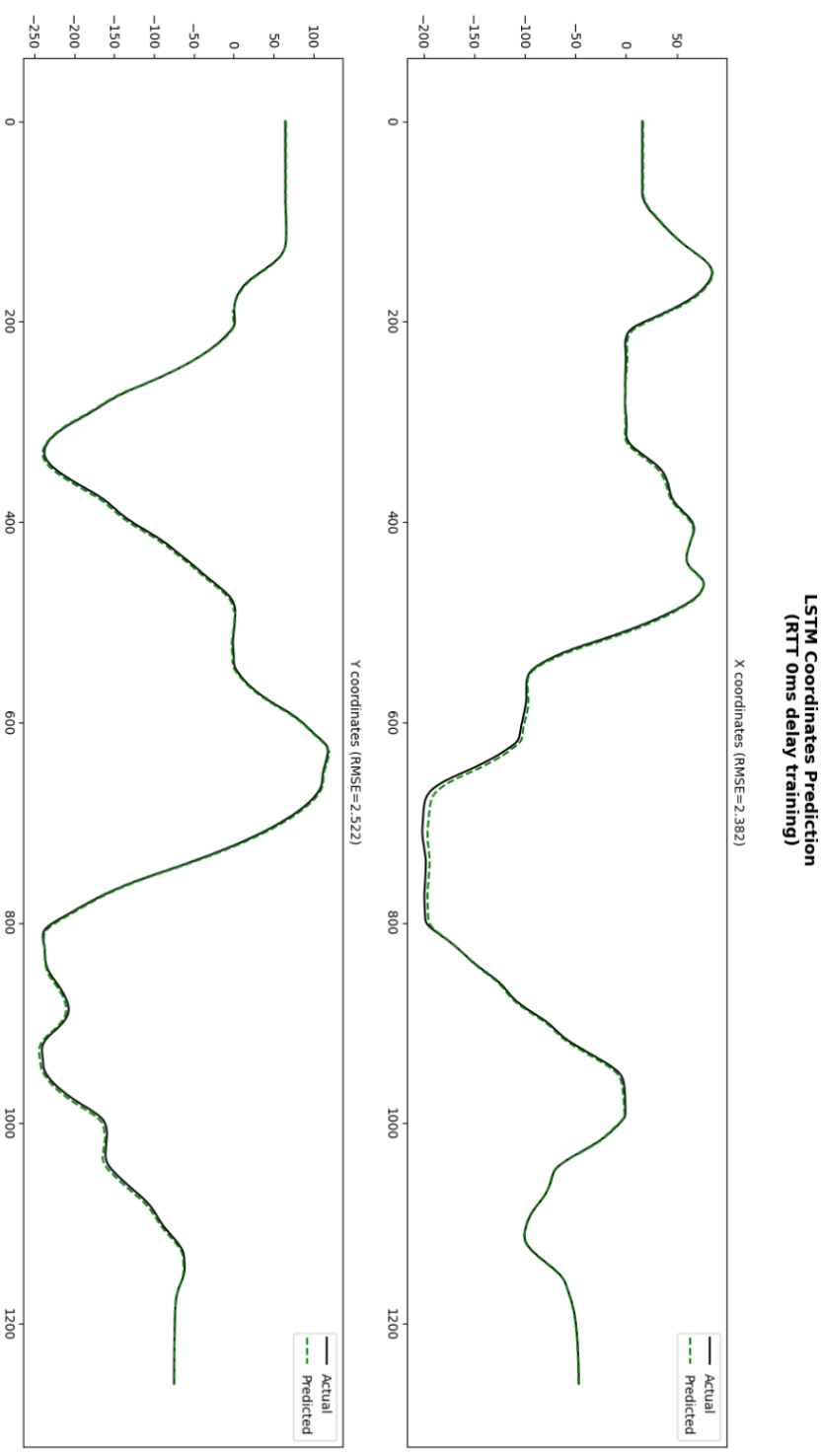

Fig 4.3.1 Actual-Predicted Comparison Chart on 0ms Test Set

The appendix includes all the actual-predicted comparison charts of the above testing sets.

V. Discussion

i. Comments on Driver's Performance under Full RTT

Evidently, RTT negatively impacts the driving quality. Both remote control drivers have not alike driving attitudes, which the figures about their Δs show the difference. Driver B is better in overall as the error distance measurements from it are relatively lower, even A and B drove on the same route. However, their driving qualities simultaneously dropped when RTT had enlarged, their total Δs reflected this feature.

The mentioned figures gave out two characteristics, to reflect how total Δ was increased with RTT,

1) They tend to make more error operations (List 4.1.3)
2) Their consequences of mistakes will also be enlarged (List 4.1.4)

Not the driving skills are degraded, RTT affects more to driver's judgement and control. Recall that RTT is composed by Command and Display Delay. Command Delay makes detention of sending driver's operation commands, in other words, driver loses the ability to make instant control. Display Delay influences more to driver's comprehension, as driver perceives on-road conditions only from display. Display Delay induces uncertainty to drivers, and correspondingly they will try to 'guess' the real position of vehicle. Command Delay distorted the control, and Display Delay affects judgement.

Back to 1) and 2), they can be further illustrated.

Command Delay limits the ability of driver to make instant false correction. Consider when vehicle is deviating from the exact waypoints, driver will logically operate corrective adjustments, relocating the vehicle position closer to the waypoints. However, the corrective operations will be somewhat deferred to vehicle in practice due to Command Delay. The corrective commands always come late to adjust the remote car to a better position. This gap time also allows remote car goes further from the waypoint, and thus the maximum deviation was enlarged. In fact, the remote car might be not relatively large at some instants, if the adjustments can be activated on time. Pre-adjustment is finally turned to be post-adjustment in practice. Command Delay limits the corrective ability of remote-control driver. Below graph illustrated how Command Delay influences during driving,

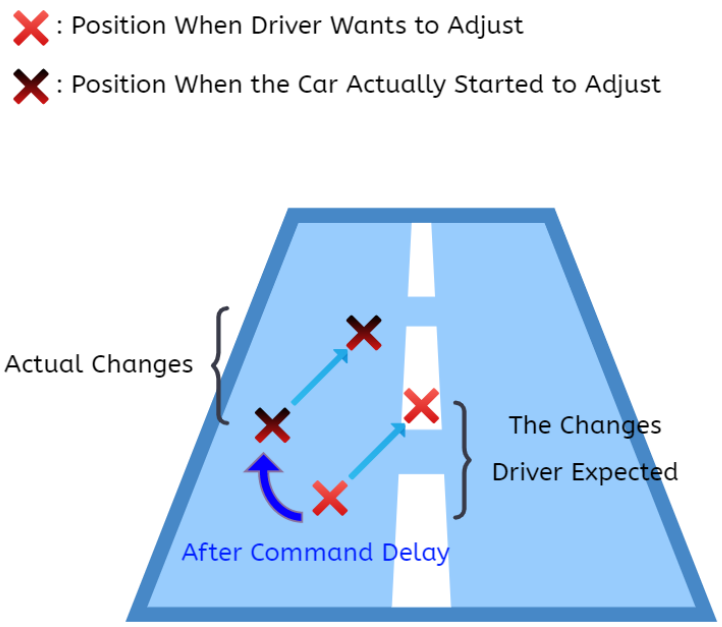

Fig. 5.1.1 Command Delay Influences

Display Delay distorts the driver's perception. Graphics sent from camera sensors dominate the source of information which a remote-control driver will rely on. Consider, the screen is showing the remote car have been driven to a position where the driver will decide to make adjustment. However, the point is quite far away in reality at that moment since the frame was not showing the real-time position. In the whole process, driver will always react lately because the information it received is delayed.

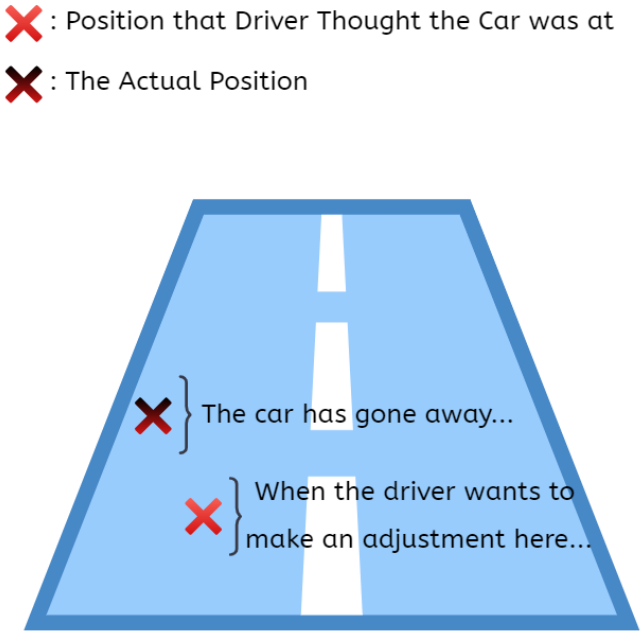

Fig. 5.1.2 Display Delay Influences

The disparity is, Display Delay will let driver takes a late response to the actual on-road condition; Command Delay will slow down the activation of driver's command.

If taking deeper look in the distribution of Δ among all waypoints, interestingly, large Δ frequently exists when the inclination is vitally changing.

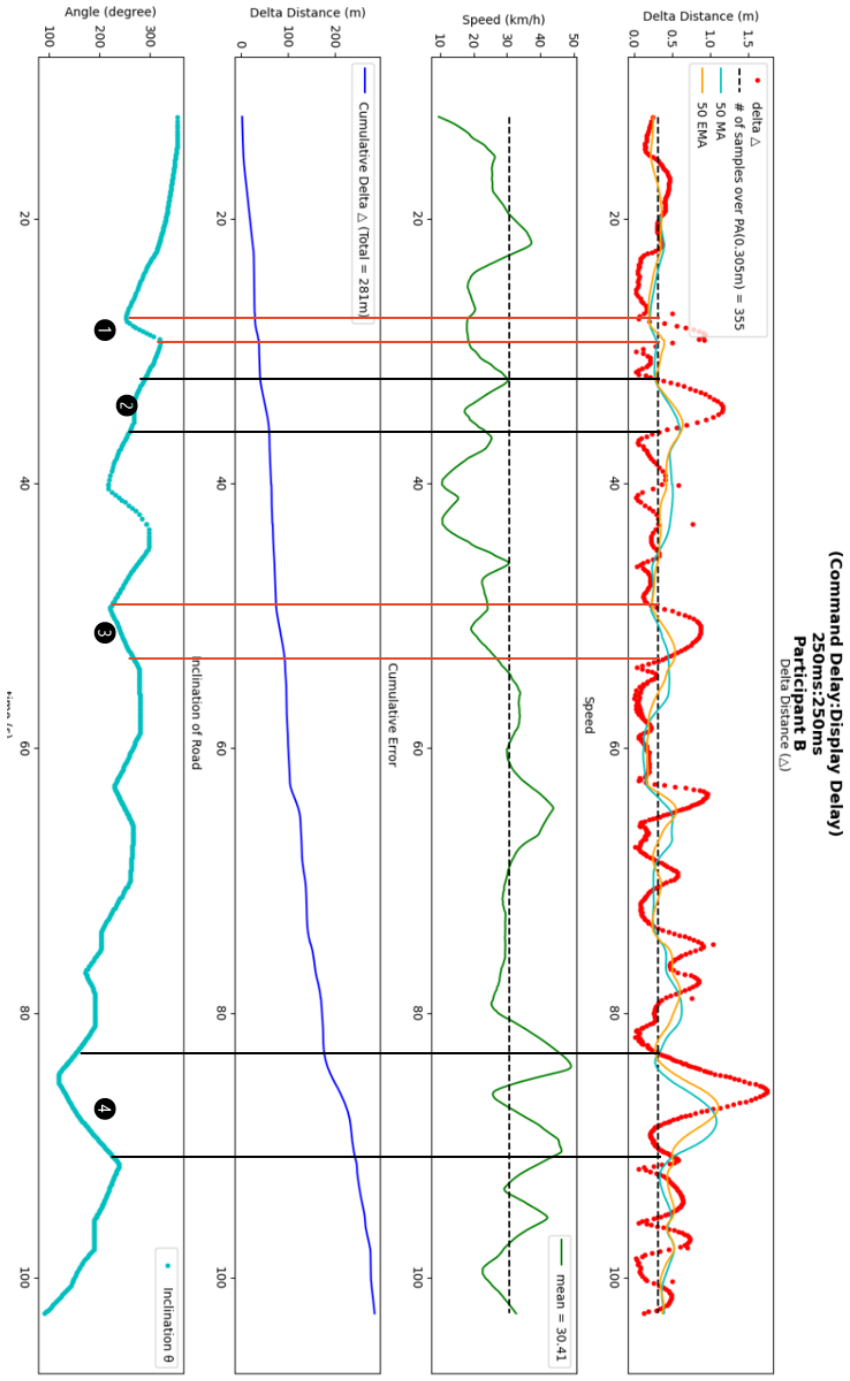

Fig. 5.1.3 Participant B's Δ Distribution Graph When RTT = 250ms [11]

[11] **All detailed graphs of participants A and B for each setup are in appendix.**

The labelled region implies where the large Δs dense to. The inclinations in those regions are characteristic with sharp variation. In other words, when curvature of the path is varying vitally, driver tends to make high Δ. Interestingly, this attribute almost appears to all recorded sample sets.

To explain this, basically it relates more to normal driving behaviours. Low variance in inclinations implies the part of road is relatively straight. Practically, straight road requires less operations. Reversely, much curved road (high variance in inclinations) requires more driving operations, especially the adjustment on steering wheel, speed varying on pedal. It concludes that RTT generates larger Δ, and mostly exists when the curvature of road is high.

ii. Comments on Isolation Test

The experiment separates the effects from Command Delay and Display Delay. The figures also compare the influences on both delay types.

Initially the expectation is Display Delay deteriorates more to driver's performance. Remind that graphics on driver's screen dominates all the information source. In other words, all the operations conducted from drivers are according to the display, because only the display gives out on-road information to drivers. The root cause of each operative command should be corresponding to what drivers seen.

Nevertheless, the results overturn the above argument. Either the total Δs and number of large Δ were recorded lower when the driving setup exists only a Display Delay. Practically, two participants endure less suffer from Display Delay. On the contrary, Command Delay degenerates their driving quality relatively more if relying on the data.

To reveal more, participant's feedback gave some clues on it.

Participant B is the only driver realized the difference of two isolation tests during experiment, and he clearly knew the disparity of two delay types. He pointed out that both were "laggy" in experience, but Display Delay reflected less "sensory legging". The main reason he provided is that the FPS (Frame Per Second) of the display is relatively low comparing to the online games he played usually. Lacking frames has more meanings to him, in stead of the delay issue. "Comparatively, the effect from delayed screen would be more negligible. Even though when I was doing the previous task (Command Delay Isolation Test), I still got a little bit suffers", as Participant B commented.

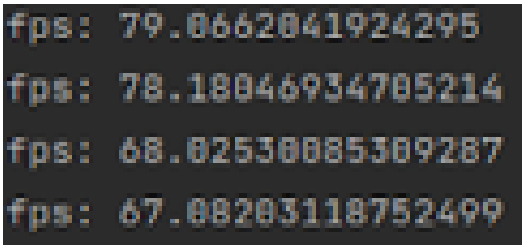

Fig. 5.2.1 A Randomly Jotted Figure about The FPS of Experiment

iii. Defections on Both Experiments

Carry on, the main defection is about the frame rate issue. Obviously, the experiment was conducted without concerning the FPS problem.

FPS relates to the amount of information that a driver can receive. For delayed screen or late arrival of operative command, driver can still “guess” or take pre-adjustment to fit the delay issue. Lacking frames limit how well the driver will know about the on-road condition. The effect is more direct and harsher to resolve. Any human estimation will be seemingly harsher if the information is in shortage.

Certainly, the figures on isolation test are still referenceable. But for a delayed screen, the arrival time of graphics is not the unique factor affecting the driving experience, there may exist other elements like frame rate which was not taken in consideration. And for the configuration in this experiment, it is challengeable to reach higher FPS due to limited processing power.

Those setups consume most in parallel computing power. Most scripts were run concurrently to operate data collection, environment setup etc. And the FPS was still being maximized in such case. The only way to improve is applying better graphic cards/processing unit to achieve speedy frame displaying.

As the Fig. 5.2.1, the Frame Rate mostly reached to 60 to 70 practically. Here is proposing that 60 is not enough for remote driving panel in reality, as participant B reflected.

Another issue is the sample size is less. Referencing to the similar experiment form US researching team, their figures were grounded by 11 participants [3]. Low sample size makes the results more judgeable. However, the situation right now is tough due to pandemic. It indivertibly restricted the number of participants.

iv. Comments on LSTM Model

To deal with the delay issue, this LSTM predictive model is one engineering solution being raised. The model aims to reduce the adverse effects from receiving delayed information. And the predicted real-time position might be deviated to the actual value, but from the performance result it shows the LSTM's prediction is still referenceable.

RMSE reflects more about how concentrated the predicted data is around the line of best fit [7]. In other words, it describes how far is the predicted tendency away from the driver's attitude. As illustrated, the position information is not only carrying the movement of car, but also the driver's attitude. The model fitting in some ways is matching the driving form/pattern of a driver. By the way, model fitting includes noise cancelation. As mentioned, another purpose of modelling vehicle position is to filter out the noise during transmission. Certainly, it can also be conducted in LSTM topology.

From the results, an interesting characteristic should be explored.

If the model is trained by zero RTT dataset, then the predive performance is worse to predict non-zero RTT situation. The initial expectation here is, applying different training sets will perform similarly to the same test set, because the driver is still having the same driving attitudes (the Δ might be different, but it does not affect the changes on position).

Yet, the results show that the difference really exists. The underlying reasons could be guessed as,

1. Seemingly, it is overfitting.
2. Driver was having different driving attitudes when behaving various RTTs.

Overfitting occurs when the model fitting is outer performing on training set, but not generalizing well on test set [8]. Simply, the model was trained specific on training set, which is irrelevant in other data. And in fact, LSTM is quite easier to generate overfitting, the reason is LSTM takes more parameters from training set to process model learning [9]. The extra parameters gave out the uniqueness to training set, so that the model is overly fitting to it. In such case, performance on a different test set will be relatively poor.

Drivers might behave differently under different delay time setups. They might be more careful, or some unusual moves will be unintentionally operated. The changes on RTT might lead a different driving experience to them. After the experiments, both participants were asked about the feedback. They both indicated that they will be careful than usual when delay exists. This 'carefulness' might be the root cause of driving differently.

v. Defection on LSTM Model

Obviously, the predictive is not guaranteed to apply in practice. Carla Simulator references physical parameters and laws to provide a realistic environment. The steering wheel and pedal settings are close enough to normal driver panel. However, there are many discrepancies in real life, like visibility on road .etc.

The main defection here is, the geometric changes on position height is not included. Here, the z-axis (along altitude) has not be taken into consideration. Then whole predictive system is just available on 2-D plane, although the maps chosen for sampling are mostly flat. The reason not considering 3-D movement is that the sampling will be harsher. There are more input features needed if considering three dimensional case, like the pitch and roll,

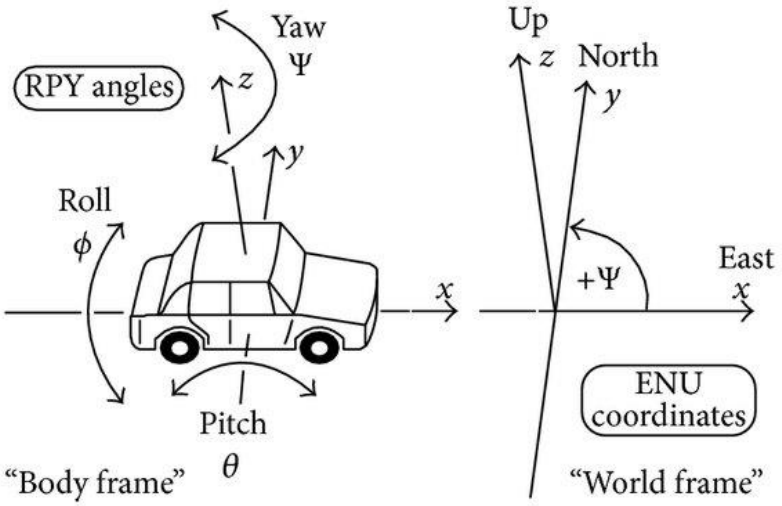

Fig. 5.5.1 Pitch-Yaw-Roll of a Car

The burden of sampling will be heavier, and it affects the driving environment setup by losing a portion to do additional samples. To maintain computer's efficiency, three-dimensional case is neglected.

Another drawback is the samples are only recording forward movements. There are not backward activities exist in all samples. The reason is the same as above, extra sampling will affect processing a stable driving environment. To consider backward cases, an extra Boolean input data is needed, to record whether the car activated reverse operation or not.

The above shortages can be actually resolved by upgrading the computational power.

Moreover, if enhancing the reliability of the LSTM model, the sample size is not acceptable. There are many features of road-conditions in the world. However, the selected maps are tending to be highway, country road or in a town. There are mountains and many different routes with unique geographical features. If enhancing such model to be highly applicable through the world, sample size is certainly not enough.

Let alone to say, the vehicle in experiment is fixed as Tesla Model 3. Different vehicle models might lead a variety of results. If increasing the compatibility, more driving experiences with various cars will be needed.

VI. Conclusion

The whole experiment was conducted successfully, although there are many defections. Again, really appreciate to Dr NEKOUEI, Ehsan, who directed the whole run of the project.

The project revealed how RTT relates to the driving deviation. Through teleoperation, the security of drivers might be ensured, but relatively, it limits a certain extent of flexibility on driving control. With such aftermath, the safety to pedestrian is not guaranteed. As the results, deviated control is the potential risk which might badly affect to on-road environment.

To resolve such issue, this paper suggested one modelling topology to get rid of the delayed effects from RTT. LSTM is good at predicting sequential data set, like the positions of vehicle. This assists to remote-control drivers by providing a referenceable “real-time data”. It eliminates the suffers from receiving delayed information.

From teleoperation to fully autonomy, machine learning is still the topology to assist human driving control. Either before or after advancing to L05 generation, machine learning is still the major element in driving history.

~ This page is intentionally left blank ~

Appendix

i. Selected Maps in Scenario II

Fig. 6.1.1 Town 01: A basic town layout consisting of "T junctions"

Fig. 6.1.2 Town 02: Similar to Town01, but smaller

Fig. 6.1.3 Town03 With a 5-lane junction, a roundabout, unevenness, a tunnel, and more.

Fig. 6.1.4 Town06 Long highways with many highway entrances and exits. It also has a Michigan left.

Fig. 6.1.5 Town07 A rural environment with narrow roads, barns and hardly any traffic lights.

ii. Δ's Distribution Curves

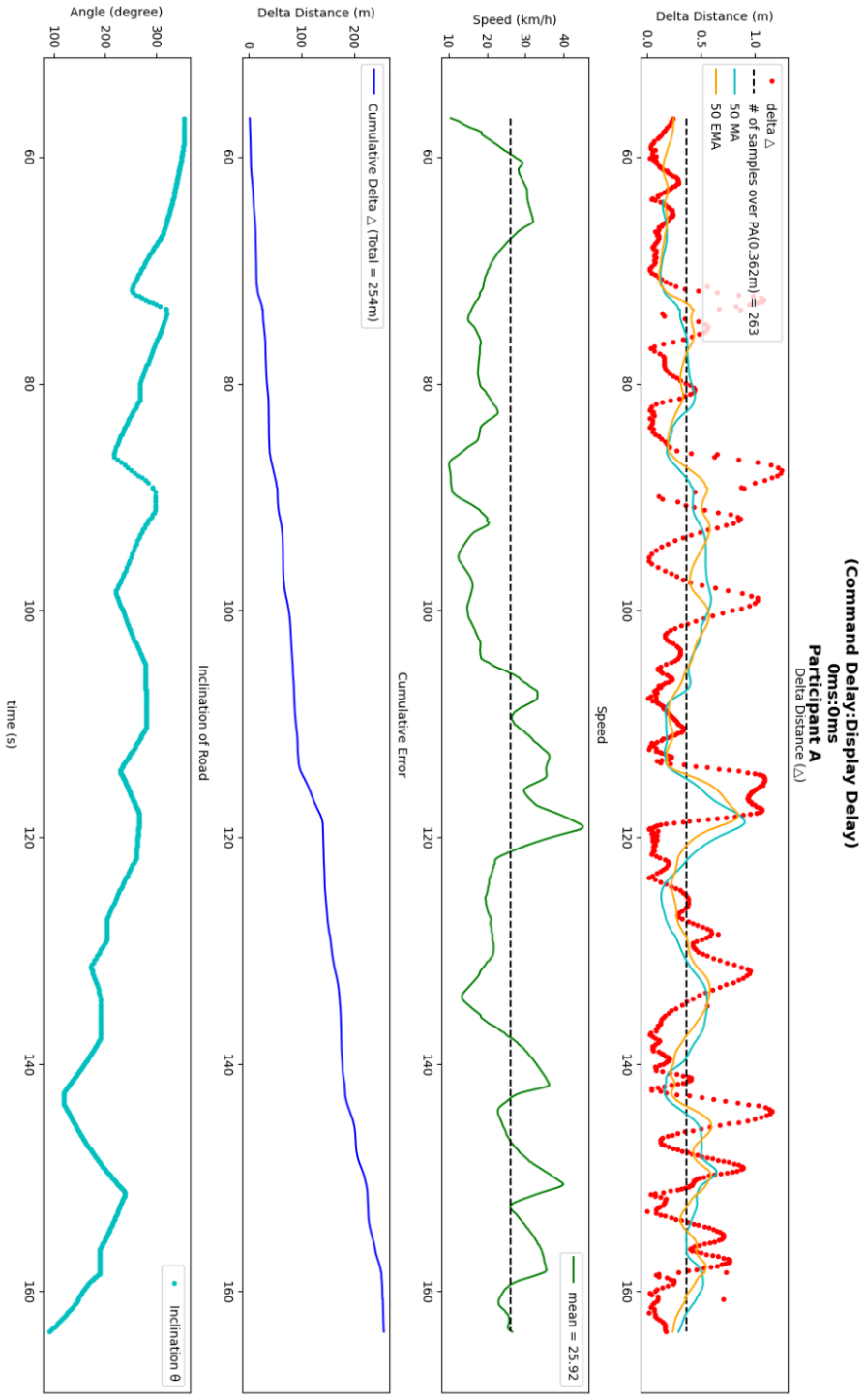

Fig. 6.2.1 Participant A: 0 ms RTT

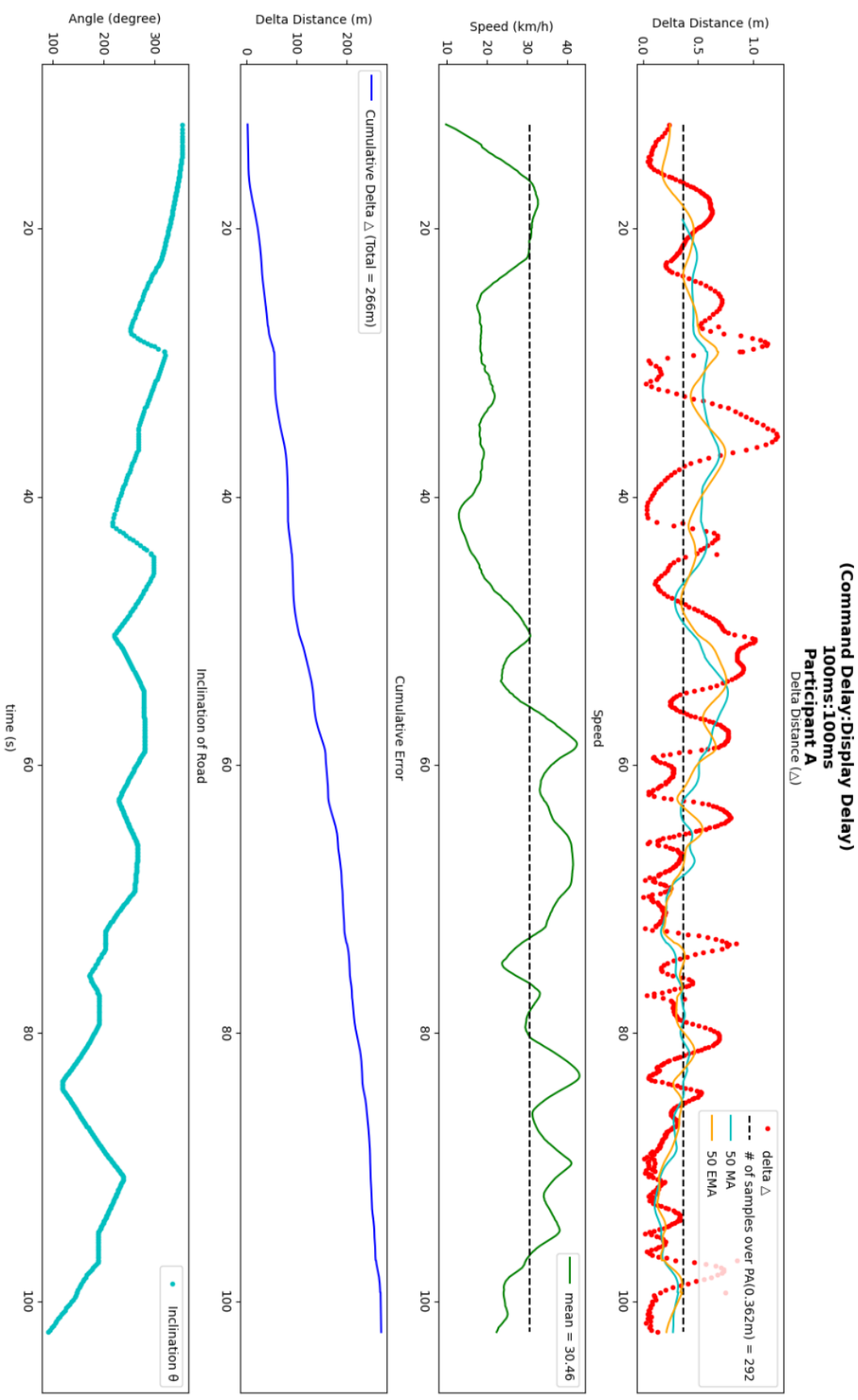

Fig. 6.2.2 Participant A: 200 ms RTT

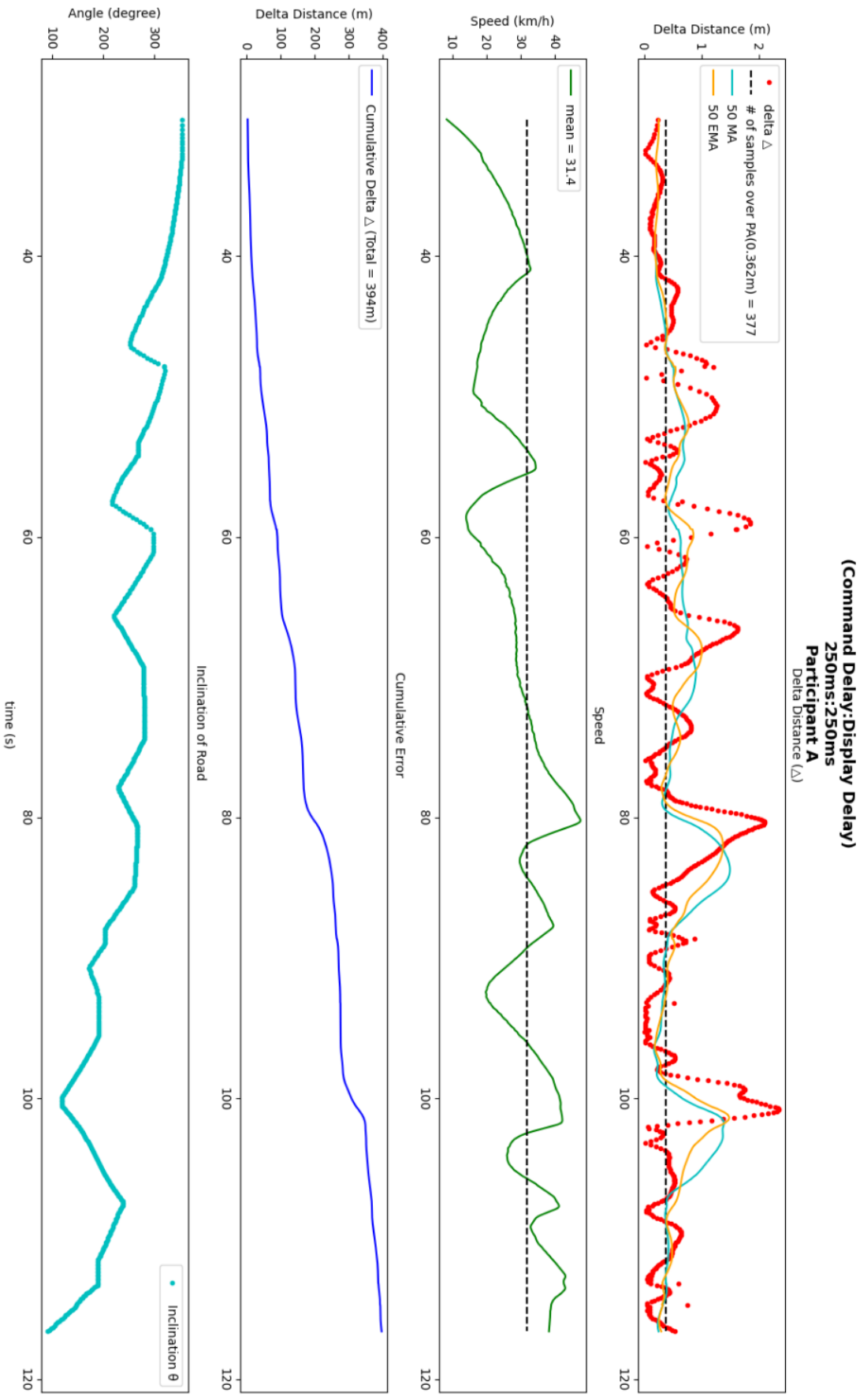

Fig. 6.2.3 Participant A: 500 ms RTT

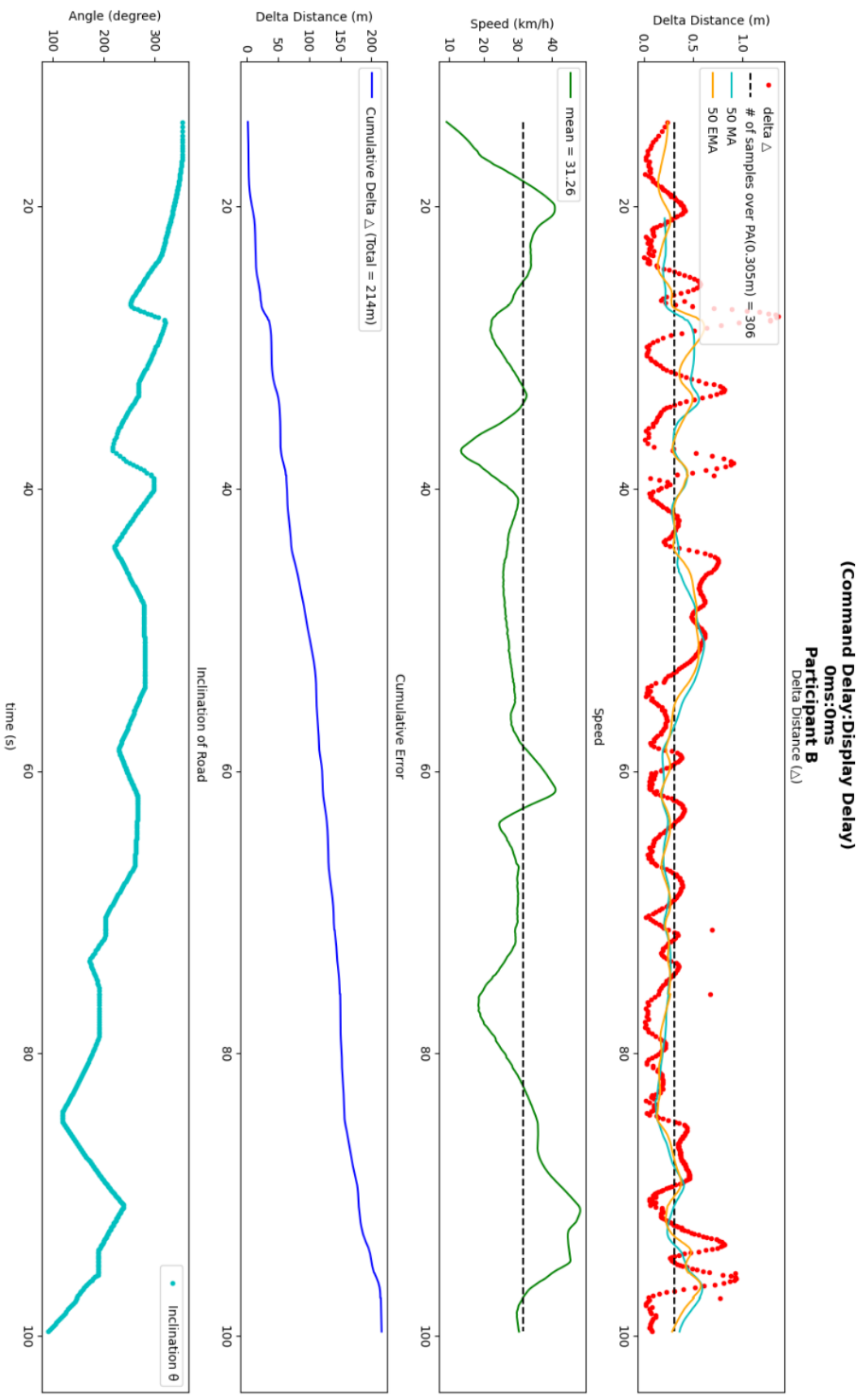

Fig. 6.2.4 Participant B: 0 ms RTT

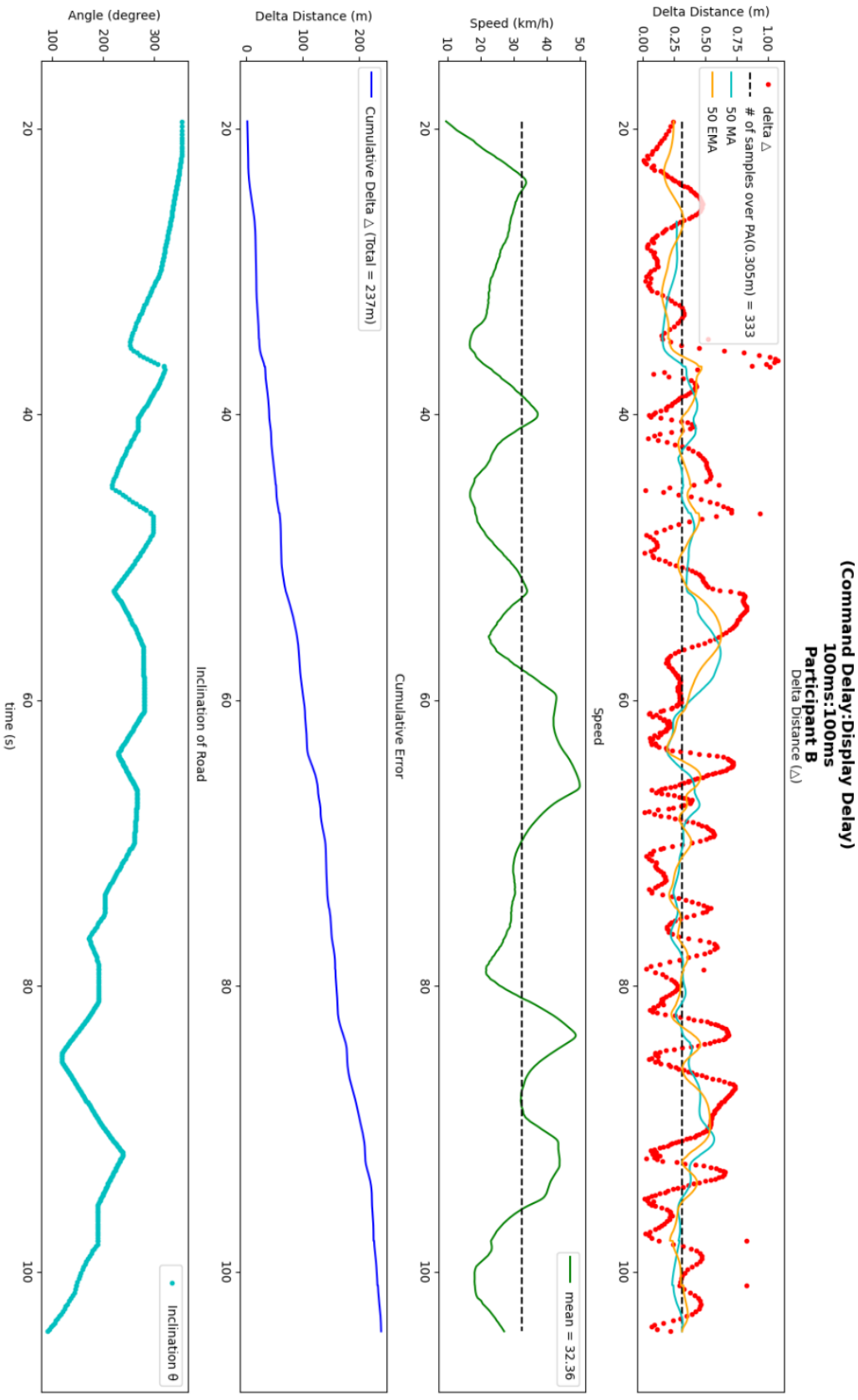

Fig. 6.2.5 Participant B: 200 ms RTT

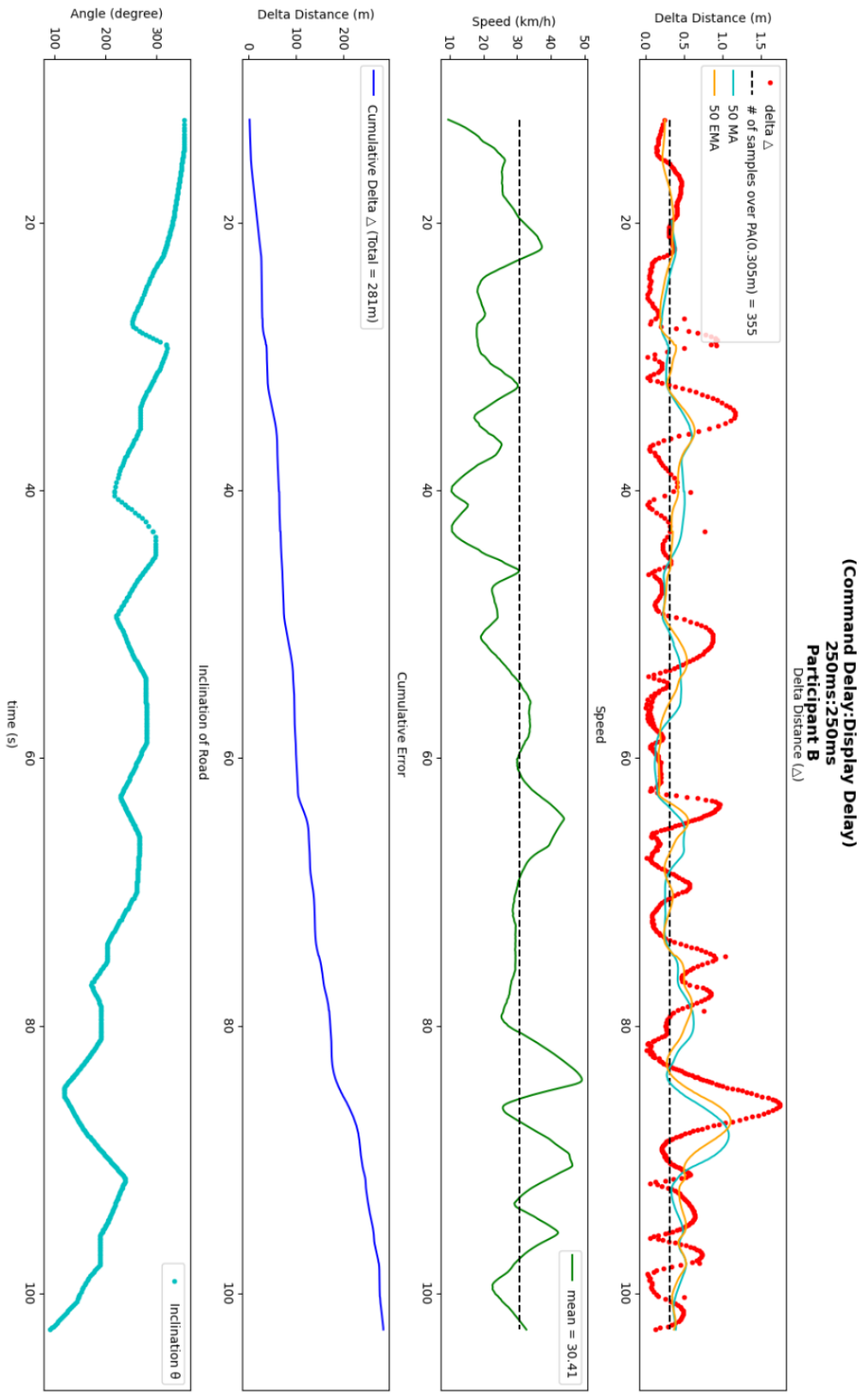

Fig. 6.2.5 Participant B: 500 ms RTT

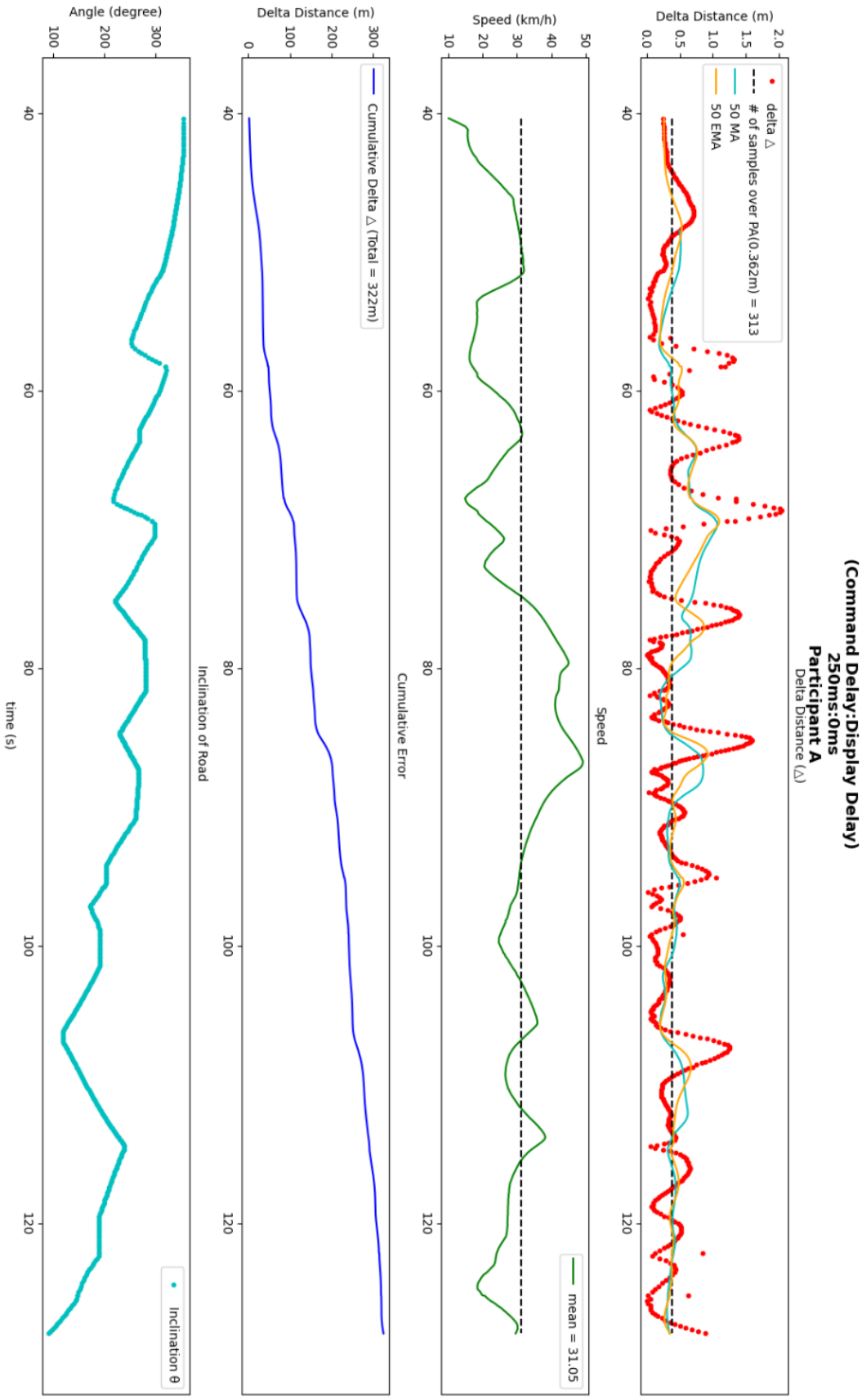

Fig. 6.2.6 Participant A: Command Delay Isolation Test

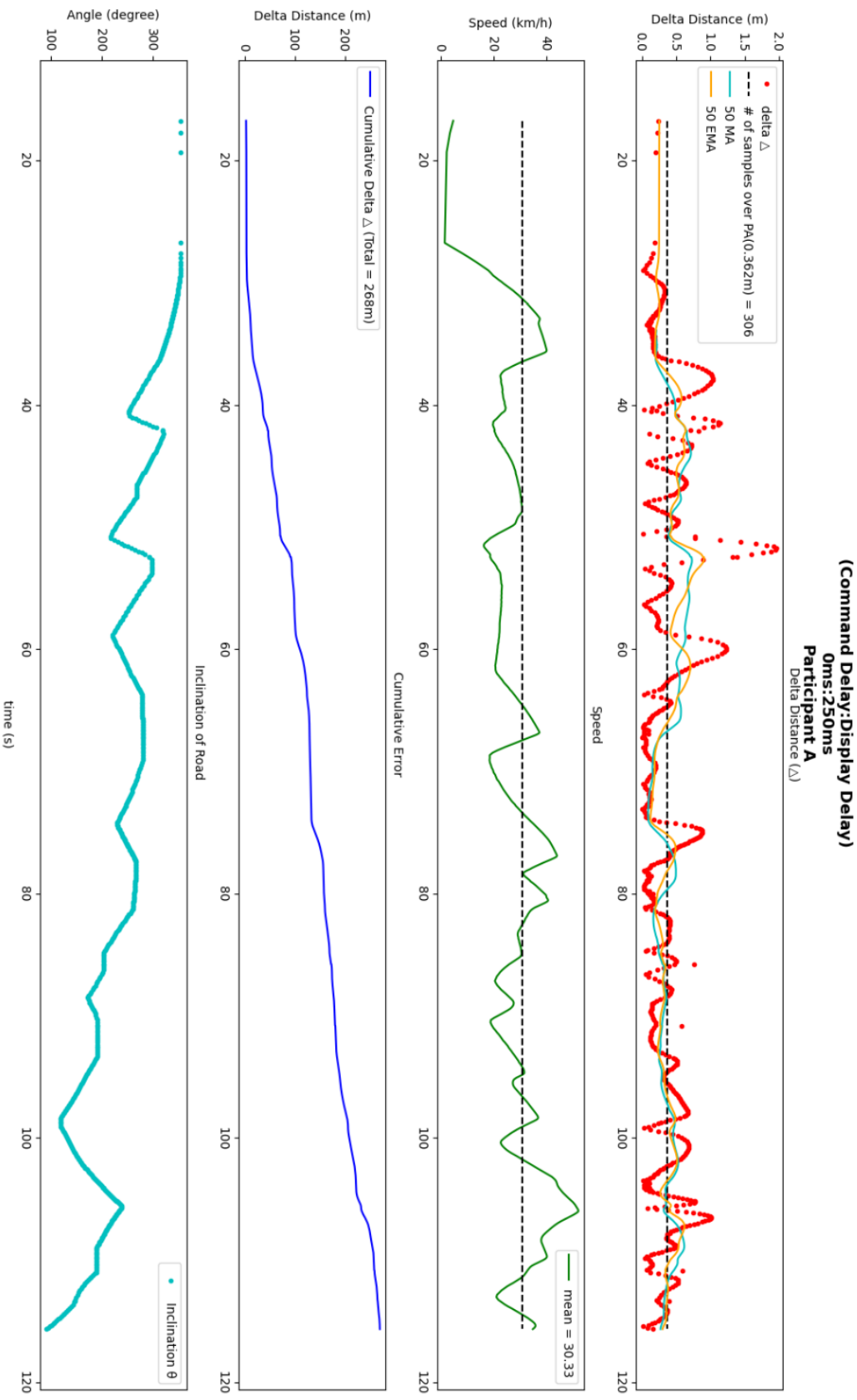

Fig. 6.2.7 Participant A: Display Delay Isolation Test

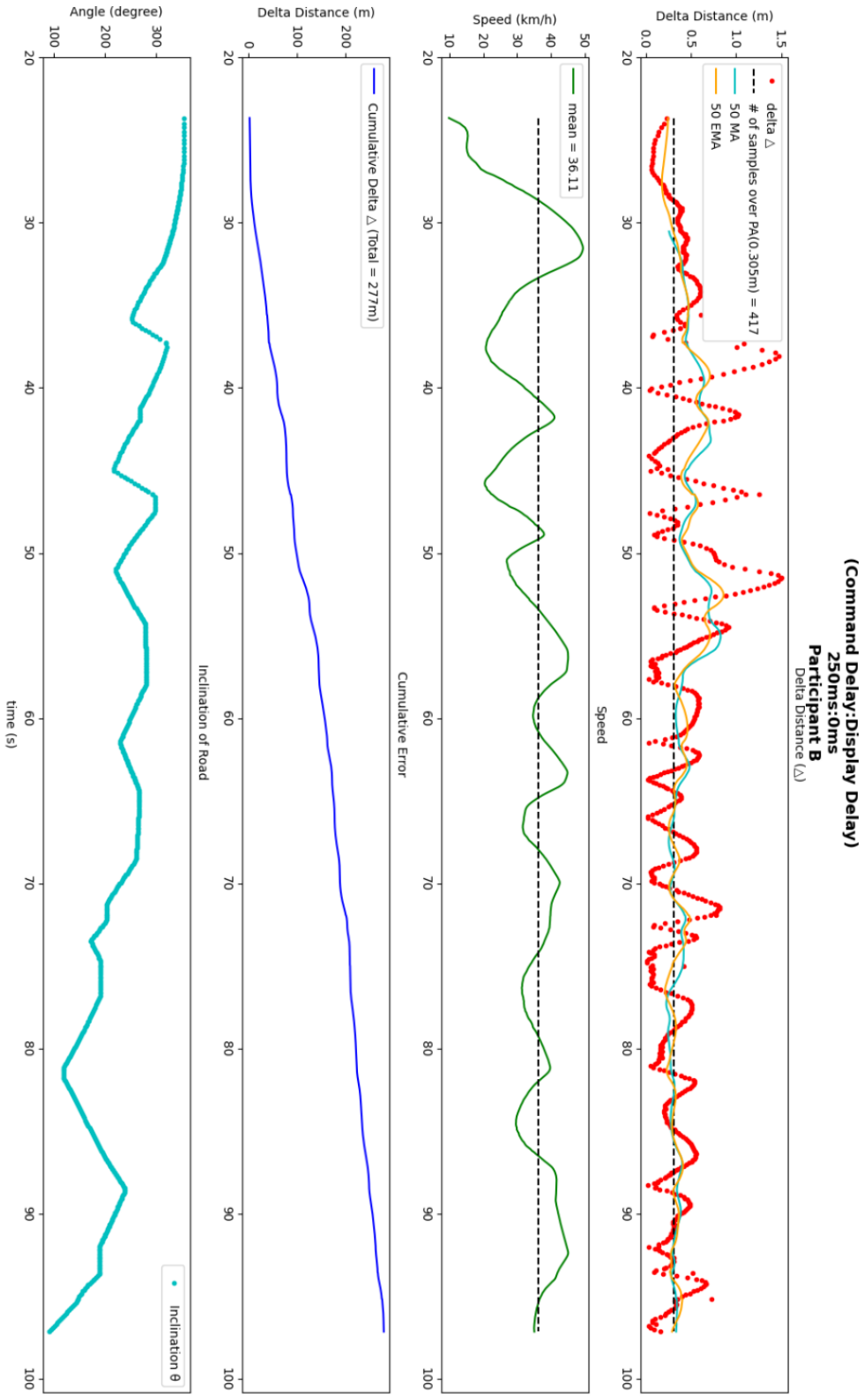

Fig. 6.2.8 Participant B: Command Delay Isolation Test

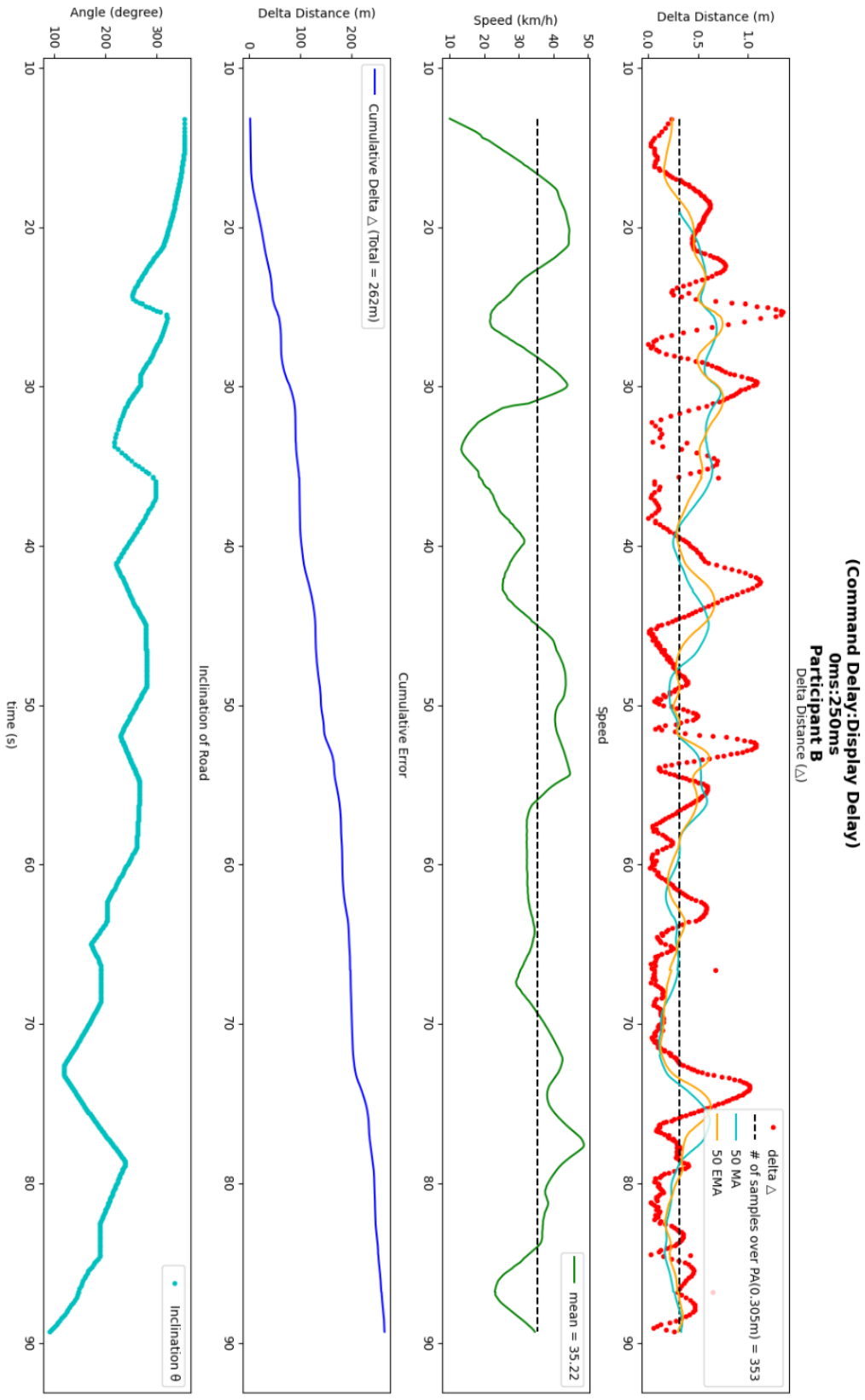

Fig. 6.2.9 Participant B: Display Delay Isolation Test

iii. LSTM Actual-Predicted Comparison Chart

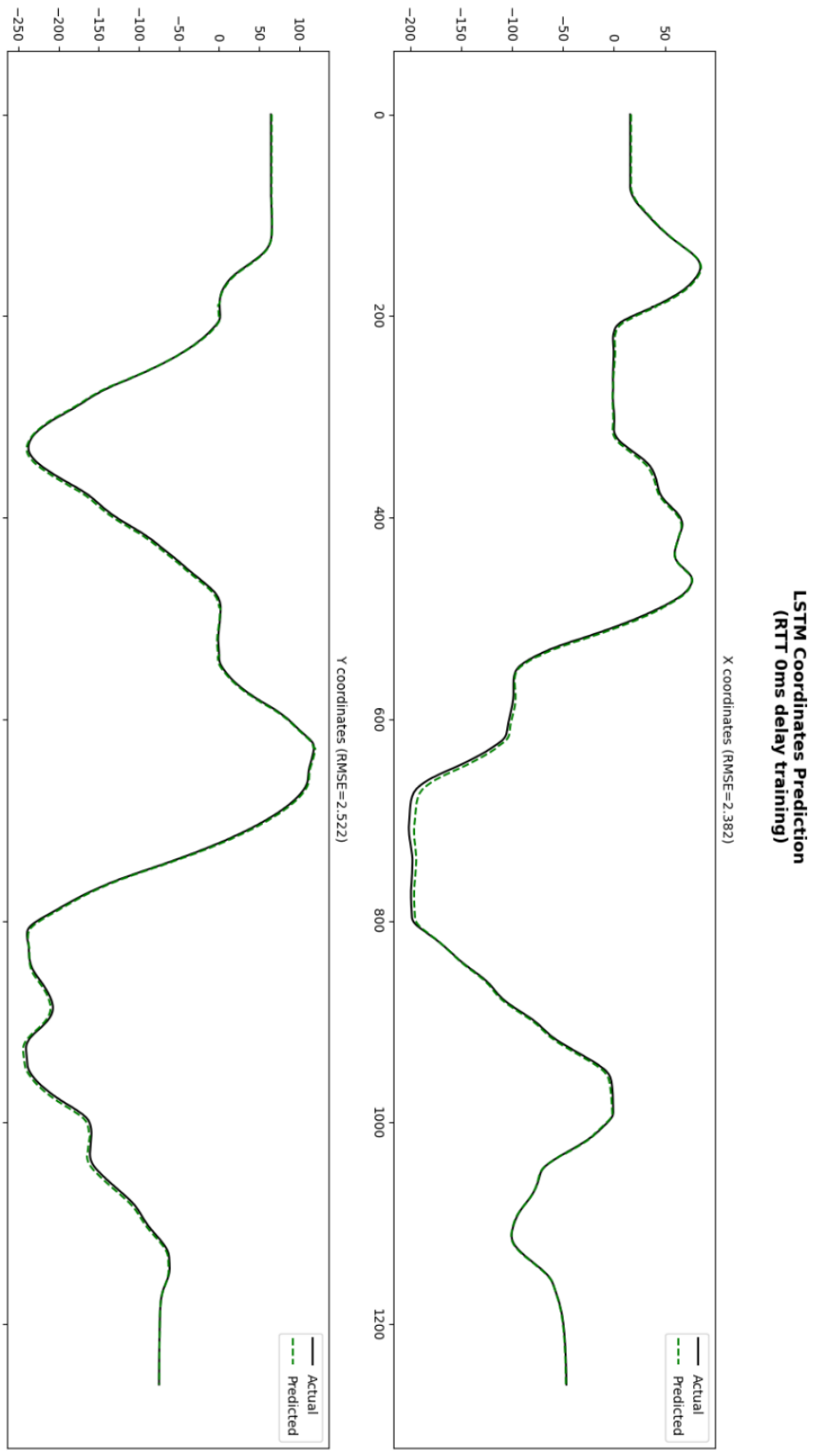

Fig. 6.3.1 LSTM Actual-Predicted Comparison Chart: 0ms RTT

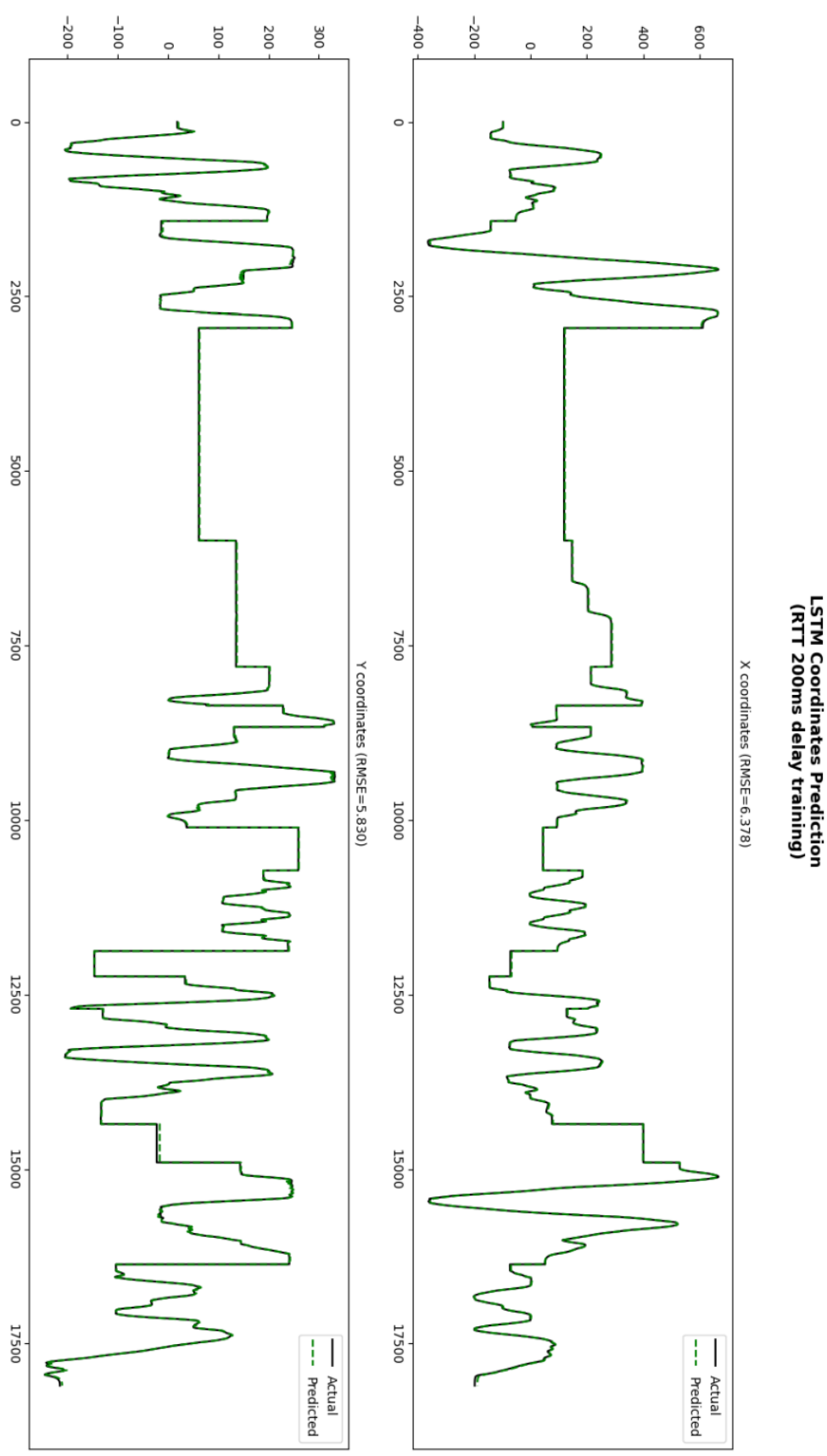

Fig. 6.3.2 LSTM Actual-Predicted Comparison Chart: 200ms RTT

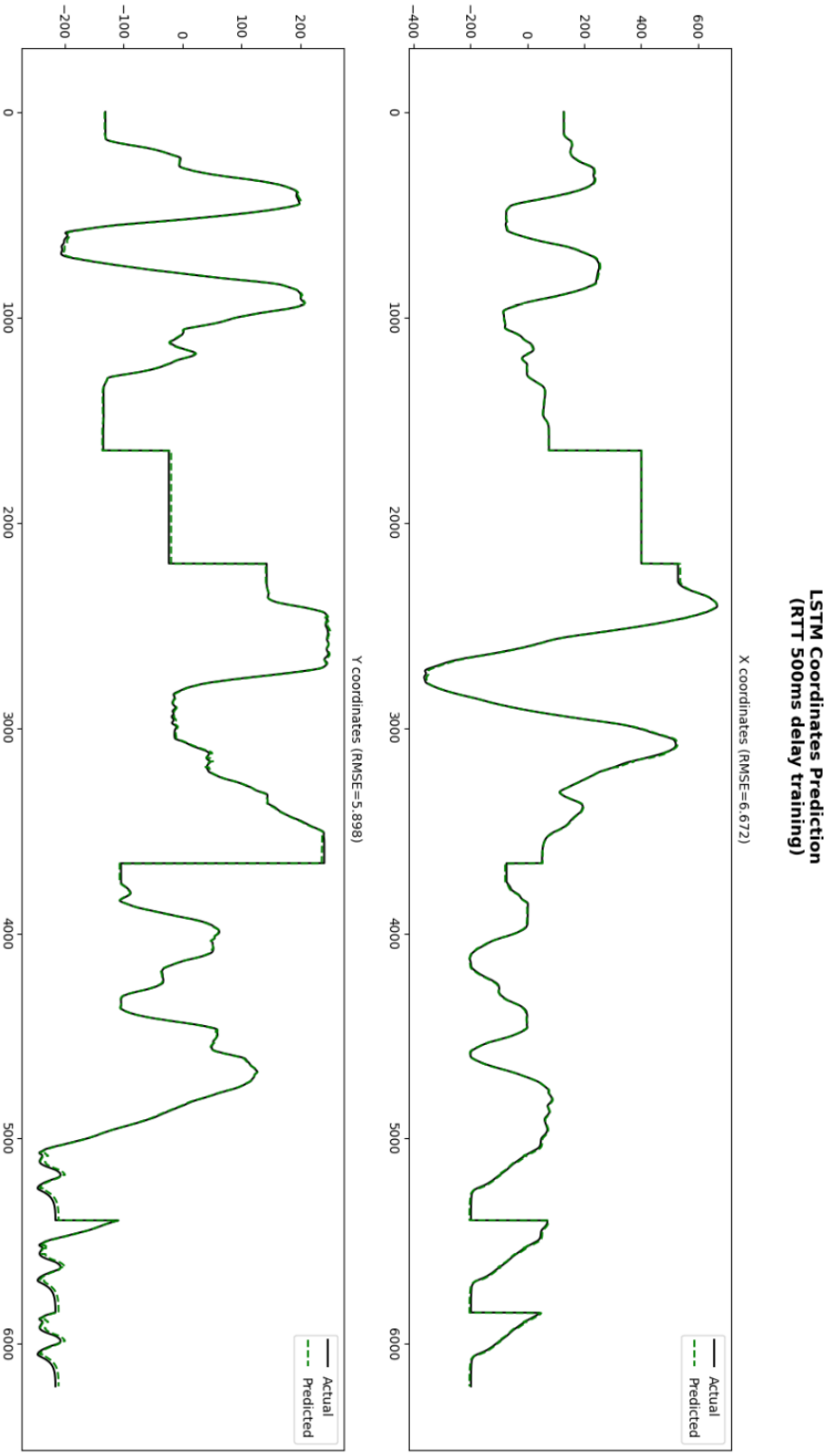

Fig. 6.3.3 LSTM Actual-Predicted Comparison Chart: 500ms RTT

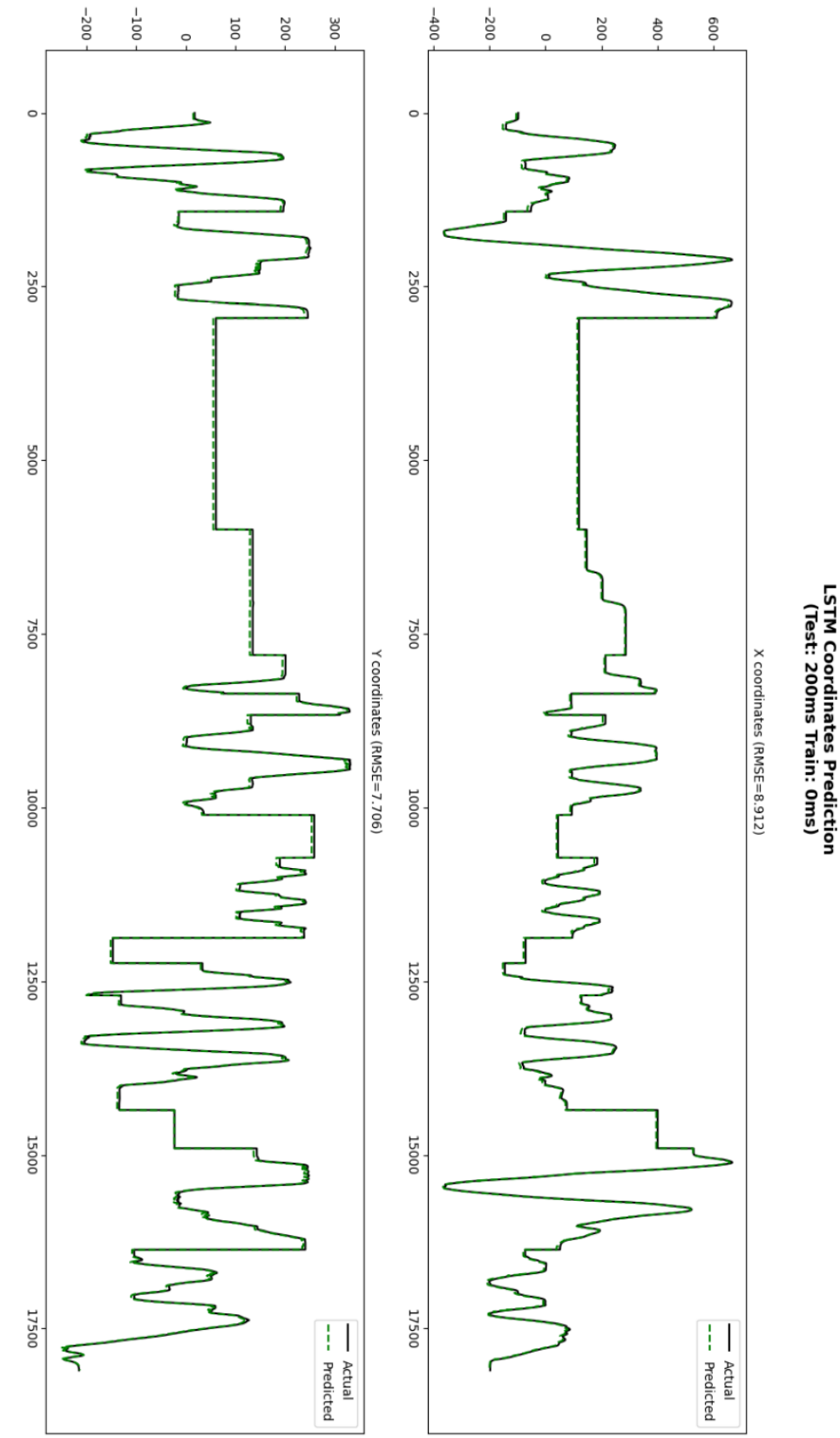

Fig. 6.3.4 LSTM Actual-Predicted Comparison Chart: 200ms RTT (Trained by 0ms RTT)

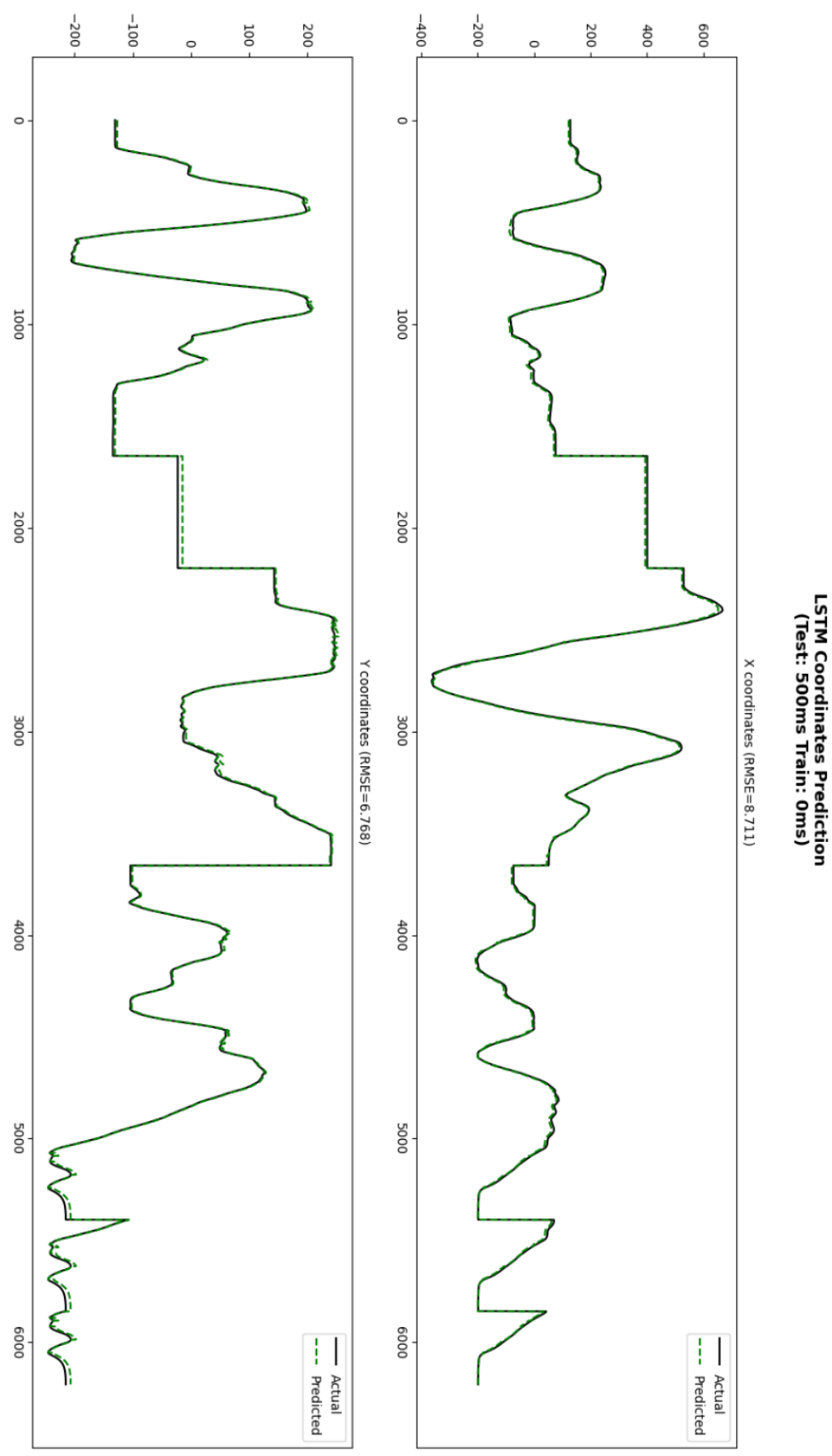

Fig. 6.3.5 LSTM Actual-Predicted Comparison Chart: 500ms RTT (Trained by 0ms RTT)

iii. Scripts in the Experiment

Git Hub Link:

https://github.com/vincent991214/CarlaFYP.git

References

[1] "Society of Automotive Engineers International," [Online]. Available: https://www.sae.org/.

[2] ERICSSON GROUP, "Remote operation of vehicles with 5G," ericsson.com, 2017. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/remote-monitoring-and-control-of-vehicles.

[3] Ruilin Liu, Daehan Kwak, Srinivas Devarakonda, Kostas Bekris, and and Liviu Iftode, "Investigating Remote Driving over the LTE Network," in *The 9th ACM International Conference on Automotive User Interfaces and Interactive Vehicular*, Germany, 2016.

[4] CARLA, "CARLA Simulator," [Online]. Available: https://carla.readthedocs.io/en/latest/start_introduction/.

[5] ERICSSON GROUP TEAM, “Remote operation of vehicles with 5G,” Telefonaktiebolaget LM Ericsson, [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/remote-monitoring-and-control-of-vehicles.

[6] S. Saxena, "Introduction to Long Short Term Memory (LSTM)," Analytics Vidhya, 16 3 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

[7] S. Glen, "RMSE: Root Mean Square Error," Statistics How To, [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/.

[8] B. Carremans, "Handling overfitting in deep learning models," Medium, 23 8 2018. [Online]. Available: https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e.

[9] EMIL HVITFELDT & JULIA SILGE, "Supervised Machine Learning for Text Analysis in R," [Online]. Available: https://smltar.com/dllstm.html.

[10] European Space, "What is GNSS?," 3 12 2021. [Online]. Available: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss.