IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 16, 2022, accepted June 3, 2022, date of publication June 16, 2022, date of current version June 23, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3183758

Remote Driving Control With Real-Time Video Streaming Over Wireless Networks: Design and Evaluation

YANG YU AND SANGHWAN LEE
Department of Computer Science, Kookmin University, Seoul 02707, South Korea

Corresponding author: Sanghwan Lee (sanghwan@kookmin.ac.kr)

This work was supported in part by the Global Scholarship Program for Foreign Graduate Students at Kookmin University, South Korea; and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), Ministry of Education, under Grant NRF-2019R1F1A1061136.

This work involved human subjects or animals in its research. The authors confirm that the research was performed in line with the Ethics Guideline for Researchers of KoNIBP (The Korea National Institute for Bioethics Policy).

ABSTRACT There is a large gap between current AI-based autonomous-driving cars and fully autonomous cars, where the remote control of vehicles can be a unique solution to fill the gap. The remote control enables valuable operational data to be obtained, thus laying the groundwork for gradually increasing autonomous driving performance in the future. Moreover, human assistance through remote driving can offer more flexibility and intelligence than a single artificial intelligence. However, the real-time transmission of data and images is particularly crucial for remotely driven vehicles. The latency between a vehicle and the controller depends on the video streaming communication methods and transport protocols. Furthermore, the control or driving performance also depends on the vehicle speed likewise. Therefore, in this paper, we explore the impact of different communication methods of video streaming and vehicle ''speed'' on the performance of remote driving. We design a vehicle remote driving system based on ROS (Robot Operating System) with ROS as the core communication architecture to realize remote control of vehicles. The video stream is transmitted using three different streaming methods: ROS multi-computer communication, TCP protocol, and UDP protocol. To be specific, we implement a simple remote driving system for a ''model'' car, drive it at different speeds, and analyze how the drivers perform in terms of whether the vehicle gets off the track while driving. Based on the results, we find that ''UDP-based video streaming'' achieves 720P video streaming with a latency of less than 50ms, which is helpful for further research on remote driving. The experiment by ''directly'' observing the vehicle with eyes instead of using video streaming is also conducted to remove the video streaming latency. The experiment results show that the speed of the vehicle and the video streaming methods have a significant impact on the driving performance, and UDP protocol-based video streaming method is better suited for remote driving. The results imply that remote driving should be used in a low-speed environment rather than at high speed.

INDEX TERMS Autonomous driving, feedback delay, human–vehicle interaction, live streaming, network latency, remote control, remote driving, round trip time, ROS, video streaming.

I. INTRODUCTION

Autonomous Driving (AD) has been a popular research area in recent years and has had a huge impact on the transportation industry. In particular, it plays an important role during the Covid-19 epidemic in 2020, where unmanned logistics and distribution technologies have been used to avoid contact distribution. However, due to safety, legal, and regulatory reasons, AD technology has not yet been widely popularized and applied.

The currently accepted standard for AD in the global automotive industry is proposed by the International Society of Automotive Engineers (SAE) and is divided into six levels, ranging from no automation (L0) to full automation (L5) [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

The details of the division are shown in the table below TABLE 1. AD at current stage mainly has the following functions: advanced predictive diagnosis technology, real-time remote monitoring, advanced remote driver assistance system (ADAS), and ultra-high-definition multimedia streaming service [2]–[4].

In the L4 level AD application scenario, the vehicle needs to drive autonomously on a limited section of an open road, such as Waymo, but still delineate the area to be driven. Under normal circumstances, the sensors and perception algorithms on self-driving vehicles can autonomously perceive the surrounding country environment, locate and plan the travel route in real-time, and travel automatically according to the planned route. When the self-driving vehicle encounters some special circumstances that prevent it from continuing its journey, it needs to take over and intervene manually to help it get out of the current predicament. The frequency of takeover of self-driving vehicles represents the maturity of the system, and the ultimate goal of the industry development is to achieve zero takeovers of AD and reach a real commercial landing state [6]. Although teleoperation shifts the concept of autonomy back to humans, it contributes to the development of automated systems [7].

In the current development stage, most AD vehicles still require driver takeover and intervention in certain complex scenarios and when encountering certain special situations [8]. To further reduce costs and promote the faster implementation of AD vehicles in scenarios, the industry has begun to experiment with remote monitoring and remote operation, replacing on-site drivers with remote drivers to improve efficiency [9], [10].

Remote driving is a kind of technology based on the characteristics of a network with ''high bandwidth and low latency'', which transmits the information around the vehicle collected by sensors such as vehicle cameras and radar to the control station, and the driver makes a judgment and operates to realize remote control of the car accordingly [11]. This technology can be used for remote control scenarios of vehicles, especially in harsh environments and dangerous areas, as well as remote supervision of vehicles in emergencies, and has a wide range of applications in the future.

As a technology with wide application prospects in the future, remote driving has strict requirements for accurate transmission of driver operation behavior information and timely return transmission of information from the vehicle end. Latency and latency jitter of the remote control greatly reduces the level of user experience [12]. The network latency brings great challenges to real-time vehicle control and video streaming transmission, etc. How to conduct low-latency and efficient communication in a limited network environment is a very important aspect that affects the remote driving of vehicles.

Whether it is AD or remote control technology, communication relies on the mobile Internet. Information is collected through various sensing devices in the vehicle, and then transferred to the vehicle control system or remote control person through sorting and aggregation, and then the computer or people give instructions to the vehicle to control the vehicle. Many researchers have studied the relationship between delay and driving performance [6], [12]–[14]. They run some experiment results and conclude that when the remote driver faces a random network delay and a network delay where the maximum delay is more than three times the average value of the random delay, performance degradation of remote driving is mainly caused by the variability of the network delay rather than the amplitude. The low latency and large bandwidth of the network are important guarantees for achieving high-reliability AD and remote high-precision control.

The commercialization of remotely driven vehicles requires wireless technology to connect remote vehicles to the network in a more economical way. Therefore, commercial wireless technologies should be used in remote control systems. Most of the commercial Wi-Fi protocols in use today are IEEE 802.11ac and IEEE 802.11ax. In this paper, we focus on the feasibility of remote driving of experimental vehicles in different network environments. In conducting the study, we base our remote control of the model vehicle on the observation of a live video stream and drive it at different speeds, aiming to approximate the real world. By evaluating the remote driving performance using different video transmission methods, we hope to answer the following questions:

- How large is the impact of the vehicle speed?
- Which video streaming method is more suitable for remote driving?
- What factors are the biggest causes of remote driving errors?

A. OUR CONTRIBUTION

In this paper, we take the application of ROS on intelligent vehicles as the entry point, design a ROS-based vehicle remote control system, and conduct extensive remote driving tests on the vehicles. The main research results of this paper are as follows:

- We designed the remote driving control system based on the communication framework of ROS. A series of functional nodes are designed to ensure the communication of the onboard system. The principle of the hierarchical design is used in the architecture, thus ensuring the portability of the system and facilitating the addition of subsequent functional nodes.
- We designed the remote console interface. It enables the operator to get a live view of the vehicle camera from the remote console to improve the human-computer interaction experience.
- The real-time video transmission systems based on TCP and UDP communication protocols are designed respectively, and the performance of video streaming methods based on different protocols is verified. Especially, we show that the video transmission system based on

TABLE 1. SAE (J3016) Automation Levels [5].

SAE Level	Name	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of dynamic driving task	System capability (driving modes)
Human driver monitors the driving environment					
0	No Automation	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	Human driver and system			Some driving modes
2	Partial Automation	System			
Automated driving system monitors the driving environment					
3	Conditional Automation	System	System	Human driver	Some driving modes
4	High Automation			System	Many drving modes
5	Full Automation				All driving modes

UDP communication protocol reduces the video transmission delay to less than 50ms.

The rest of the paper is organized as follows. Section II introduces the background and related works of AD and remote driving. In section III, we first introduce the experimental device, then describe the research methods and experimental content of remote driving. In Section IV, we evaluate the experimental results of remote driving, showing the experimental performance of remote driving using different video transmission methods. Section VI concludes the paper.

II. BACKGROUND AND RELATED WORKS

In this section, we describe the role of real-time multimedia streaming services in remote driving, then we describe the current status of the autonomous and remote driving industry, and finally, we describe middle-ware for remote driving communication.

A. REAL-TIME MULTIMEDIA STREAMING SERVICE

Nowadays, mobile multimedia streaming services are becoming more and more important in data transmission services, and their proportion in all mobile data streams is also increasing. In 2017 alone, mobile video traffic accounted for 59% of all mobile data traffic [15]. Real-time video streaming refers to low latency high-quality video transmission in a network environment with limited bandwidth so that the video content can be correctly identified and analyzed by a human-based or computer-based video surveillance layer [16]. Mobile video streaming adds uncertain environmental factors, such as vehicle speed and vibration, to pure video streaming, and these factors can increase packet loss rates, latency, and delay jitter variability. Advanced video streaming services can use more abundant recognition technologies, and can allow human-based or computer-based video surveillance layers to extract more visual features, minimizing information extraction omissions caused by video resolution factors the problem with inaccurate judgment results [17].

In autonomous driving radar is another means of perception, multi-sensor fusion is a common point of view to achieve autonomous driving at this stage, but the problem of Light Detection and Ranging (LiDAR) signals interfering with each other has not been solved so far, if the number of users on the road, the accuracy of LiDAR will be greatly reduced, it is difficult not to have an accident. The output of the radar often has spurious stray points, or some multipath propagation resulting in false targets. The biggest difference between camera data and millimeter-wave radar data is that millimeter-wave radar signal-to-noise ratio is very low, in other words, there are a lot of false detection, and when fusing visual perception results with millimeter-wave radar results, if the visual and millimeter-wave perception results do not agree, the usual practice is to trust the visual and ignore the millimeter-wave detection results. That is why the high quality of mobile video streaming is even more important.

At the same time studies have proved that in the case of high-speed vehicles (100 km/h), the throughput and RTT (Round-Trip Time) become worse, and the instability is extremely high [18]. The authors use the mobile phone to transmit video images to communicate between the vehicle and the remote driver, and find that the minimum delay of the video stream in the 3G network environment is 65ms, the maximum delay is 1, 299ms, and the average delay is 121ms [19]. The delay varies greatly. Keon Jang *et al.* investigate the data transmission performance of high-speed trains (maximum speed 300km/h) in 3G network environments and fast (100km/h) driving vehicles in 3G and 3.5G network environments, respectively [20]. In the experiment, it is concluded that the data transmission performance of the mobile node is far lower than that of the fixed node, and the conclusion of [18] is further verified. In the mobile measurement experiment, the data transmission process based on UDP and TCP transmission protocol has lower data throughput and greatly shocking data packet loss rate. Lei Kang and others measured the delay time of two-way video transmission based on UDP protocol in Wi-Fi and LTE network environment respectively as 50ms and 100ms [8].

Real-time video streaming is one of the key technologies that are highly dependent on remote driving scenarios that are currently being researched. To obtain better driving immersion and more road condition information, high-resolution

and large-size video streams have become an urgent need. Even if the quality video is compressed, the size of the video data is still very high, and the high-bandwidth network transmission requirements of video streams are far greater than the transmission rate that the low-power physical layer standard can provide [21], [22]. Therefore, on a network with limited bandwidth, how the system provides a high-efficiency video stream with low latency and low frame skipping becomes a challenge that cannot be ignored.

B. THE CURRENT STATE OF THE AUTONOMOUS AND REMOTE DRIVING INDUSTRY

In 2018, SAE (Society of Automotive Engineers) defined the level of autonomous driving [23]. Government agencies, academic institutions, and automobile manufacturers have always hoped to use autonomous driving technology to solve road safety-related problems and provide communication capabilities for transportation infrastructure and vehicles to solve a series of problems such as transportation efficiency. Some large companies have developed their AD driving platforms. For example, Baidu has developed an open Apollo autonomous driving platform, which brings together more than 90 members from all over the world [24]. Autoware has developed the autonomous driving platform Auto-ware, and more than 20 members worldwide have joined. Google started a car project called Waymo AD in 2009, and then became independent by Google in 2016, becoming a subsidiary of Alphabet, and it is a leading company in the development of autonomous driving [25]. As of 2020, the autonomous driving company Waymo announced that its autonomous driving cars have driven 20 million miles on public roads. It took Waymo ten years to complete the first 10 million miles, but it took less than a year for the second 10 million miles. Waymo's data also reflects the current status of the autonomous driving industry. Various autonomous driving companies are conducting a large number of tests. It is precise because of the joint efforts of various autonomous driving companies that the current development of autonomous driving is very rapid.

Tesla is very representative in the autonomous driving field. Tesla first equipped the production car with an L2 autonomous driving function and then used the test mode to help the AI system learn during the use and driving of the car owner. It is a method of transition from manned to unmanned driving, a method that requires car owners to take over in an emergency, and a method that is constantly controversial. Because every car owner who uses Tesla's AD is a safety officer of the vehicle, but a safety officer is not supposed to operate in a dangerous environment.

On May 7, 2016, a Tesla Model S crashed into the side of a truck on a Florida highway, killing the driver [26]. On March 19, 2018, although the sensor-equipped on an Uber self-driving car detected a pedestrian, the software system decided that it did not need to take immediate evasive action, resulting in the death of the pedestrian [27]. Facts have proved that this type of autonomous driving car accident is basically because the owner trusts the autonomous driving system too much and fails to take over the vehicle in time, causing the accident. Remote driving allows remote drivers to move the vehicles away from dangerous environments. When the vehicle system is difficult to make correct judgments, they can take over vehicle control promptly and make the most appropriate decision-making judgments.

There are naysayers in the autonomous driving industry who argue that it is difficult to imagine a remote driver hundreds of miles away being able to make timely judgments by adapting to complex situations. In most cases, the AD vehicle will stop and wait for further instructions. Another problem is that remote operation relies on common commercial networks, and if delayed, could prevent remote operators from making quick decisions at critical moments or lead to maneuvering errors [6].

Remote driving systems are stable theoretically, however in reality even a small network delay can trigger catastrophic consequences, so network delay is the primary problem faced by remote driving. The literature [28] investigates the effect between the length of delay duration and remote control and demonstrates that a short delay can reduce the remote control error rate by approximately 33% compared to a long delay. Immersion is very important for remote drivers to control vehicles remotely, and immersion, also known as situational awareness, is very important [11]. The motion of the vehicle may reduce the operator's situational awareness [29], thus making the operator more confused, which can negatively affect the control performance, and the increase in speed can lead to an increase in latency. A study of remote driving in an LTE network environment in literature [11], [13] proved that the factor that has a significant impact on the driving performance of remote drivers is the unstable delay. Literature [14] experimentally demonstrate that the changed latency creates greater difficulties for remote drivers to operate the vehicle, thus having a greater impact on the accuracy of the operation. The literature [30] demonstrates that an important factor in ensuring the safety of remotely driven cars when driving remotely is a high-quality (low latency jitter) video streaming service. Literature [31] demonstrates that advanced autonomous driving simulators can reduce less than human perception errors. In existing studies, researchers have focused on the study of video streaming quality of service and the study of remote driving based on the virtual environments, it is difficult to evaluate the performance of remote driving at the overall remote driving system level, while we used a model vehicle to build a remote driving system platform and reviewed the actual impact of delay on remote driving in a practical test.

C. MIDDLEWARE FOR AUTONOMOUS DRIVING COMMUNICATION

Robot Operating System (ROS) is an open source system that is used very, very much in the field of robotics and autonomous driving. ROS the most abundant robot operating system in the world has accumulated a lot of experience,

avoiding repeated development work by developers and improving development efficiency [32]. ROS needs to rely on the Linux system to run on top of the computer hardware. Linux is a general-purpose system and does not provide special middleware for robotics development, so ROS does a lot of work in the middle layer, the most important of which is the communication system based on TCPROS/UDPROS. The communication system uses the publish/subscribe and client/server models to implement data transfer with multiple communication mechanisms. In addition to the communication mechanism of TCPROS/UDPROS, ROS also provides an intra-process communication method called Nodelet, which can provide a more optimized data transmission method for multi-process communication and is suitable for applications with high requirements in terms of real-time data transmission. On top of the communication mechanism, ROS provides many libraries related to robot development, such as data type definition, coordinate transformation, motion control, etc., which can be provided to the application layer for use. ROS mainly integrates the robot's perception data of the environment and outputs the control of the robot. ROS makes the algorithm update and iteration faster and more convenient. It publishes and subscribes to topics in a certain format, separates each functional module and AD in general, it is also through the perception of the environment, the integration of people's willingness to control, and then output the control of the unmanned vehicle, so it is very convenient to use ROS to develop the autonomous driving system. Japan's Auto-ware and Baidu's Apollo, which are both developed based on ROS, use ROS as a middleware as a distributed processing platform.

III. REMOTE DRIVING EXPERIMENT METHODOLOGY

To more realistically study the network communication performance of remote-driving cars while driving, we use model cars instead of actual vehicles, and the terminal computer in the wireless network as the remote-driving terminal to conduct a more realistic remote-driving simulation experiment.

This section introduces the overall remote driving system from three major modules: a model car, a remote control system, and an in-vehicle control system. The system includes two data paths, one for the control commands and the other for the in-vehicle video.

A. EXPERIMENT ENVIRONMENTS

The onboard processor of a vehicle is one of the core units of the whole system, which needs to have powerful signal access, image processing capability, and real-time video encoding capability. In the overall latency of video streaming, the data processing capability of the sender is also an important factor affecting the sending speed. The performance of the NVIDIA Jetson TX2 processor can meet these requirements well. We use the Xycar-D model vehicle built on the TX2 processor, and the communication framework of the remote driving system built on this basis is shown in Fig. 2. In the test, we use the following devices:

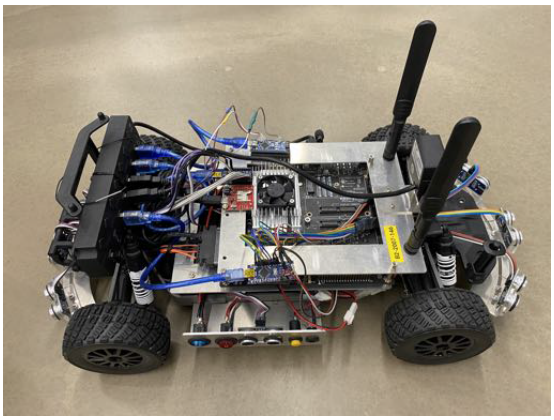

FIGURE 1. Xycar-D.

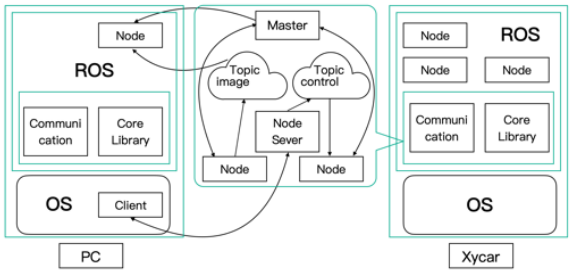

FIGURE 2. Communication structure between PC and Xycar.

- Remote driving control machine: MacBook Pro, Intel® Core i9 CPU @2.30GHz, RAM 16G. ROS on 64-bit Ubuntu on VMware virtual machine.
- Model car: Xycar-D self-driving model car (Fig. 1), NVIDIA TX2, RAM 8G, ROS on 64-bit Ubuntu [33].
- Language: C++, Python3.8

The Xycar-D vehicle that we use is manufactured using the body of 1/10 of the international standard size racing RC model car [33]. It has a brushed motor and drive gear, as well as a steering system based on Ackermann String. It uses the ROS platform and has driving characteristics very similar to real cars. Furthermore, Xycar-D is equipped with an Nvidia Jetson TX2 board, camera(170 degree wide angle lens, UVC1.1, 1280 × 720, 30fps), radar, and other equipment. The camera is located at the front of the vehicle and captures a panoramic view within one meter of the front of the vehicle, making it very suitable for remote driving studies. Before the start of the experiment, we carried out the vehicle steering, motor, network connection and other related functions, and after confirming that there are no errors, we started the relevant experiments. The speed of Xycar-D is specified as an integer and is not linear over the integer values, so we map the integer values into real speed shown in Table 2. However, it should be noted that the actual speed of speed value may differ depending on the battery level and the condition of the floor, thus Table 2 should be considered as a reference rather than the actual speed.

TABLE 2. Xycar speed comparison table.

Xycar Speed Value	10	15	20	25	30	35	40	45	50
Real Speed(cm/s)	15	41	68	87	108	118	144	151	154

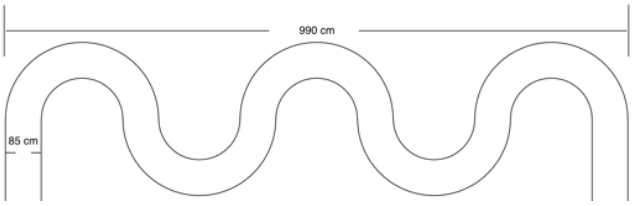

FIGURE 3. Test route.

In this remote driving experiment, the remote driver controls the model car based on the video stream displayed in the control terminal computer in real-time. The driver can send control messages to accelerate, decelerate and steer the vehicle. In the above system, network latency is a key factor in determining whether remote driving may work or not.

We have five participants in the experiments. Since the proportion of male drivers is much larger than that of female drivers in the transportation industry, we set the ratio of male to female drivers at 4:1 to more closely match the real situation in the transportation industry. To obtain statistically significant performance data, we recruited five participants: four males and one female, with an average age of 26 years in good health condition. We recruited remote drivers who were identical to drivers of real vehicles, had already obtained a regular model driver's license, and had at least 2 years of driving experience. All participants had correctable binocular vision of 1.0 and normal color vision. Participants have experience with gaming or remote-control cars. The fact that we recruited drivers without special training better highlights the reasons for errors during the remote driving experiment. The Official National Aeronautics and Space Administration (NASA) Task Load Index (TLX) is a subjective workload assessment tool that allows users to perform subjective workload assessments on the operator working with various human-machine interface systems [34]. In this experiment, we do not perform the workload assessment seriously because the experiments are similar to driving a car in a computer game. So the workload is considered very light. It should be noted that the main purpose of the experiments is to investigate the effect of different video streaming methods in the performance of remotely driving a car through communication networks.

The experimental route we designed for the model car consists of five 85 cm wide half-loops with continuous S-bends (Fig. 3). The relevant measure in the test is whether or not the predetermined route course is successfully passed. A non-successful run means that remote driving errors happen during the driving. Remote driving errors include hitting a wall, touching the edge of the track during a turn, or running off the track.

Actually, the delays of the video frame transmission in the remote control affect the performance of the remote control. It should be noted that video streaming methods may not provide real-time display of the vision of the car due to the delays. So we briefly discuss the relationship between the delay and the performance. In the streaming methods we adopt, there are mainly propagation delays, transmission delays, and video processing delays. Since the distance between the controller and the car is very short (about 1m), the propagation delay is negligible. So the main delay contributions are transmission delay and video processing delay. The video is captured at every 11ms, so the video processing delay is considered as 11ms. However, the transmission delay can be different due to different communication methods due to the packet retransmission. We describe the characteristics of the three communication types that we adopt in Table 3. However, the delay does not directly reflect the remote controlling performance. Thus in the experiment, we directly measure the remote controlling performance rather than measuring the delay itself.

B. VIDEO STREAMING METHOD

In realistic remote driving, the remote driver needs to make decisions and send back commands based on a combination of the information returned by the vehicle. To study the practical performance of data transmission while the vehicle is in motion, we focus on the most challenging case where the video source required for transmission is acquired by the camera in real-time and provided on demand. We use the following three different methods for real-time video transmission: ROS communication, TCP communication, and UDP communication Table 3.

1) ROS PUBLISHER/SUBSCRIBER BASED VIDEO STREAMING

The first method is to use ROS publisher/subscriber-based video streaming. We use the open-source project usb_cam to get the image from the USB camera and publish it as image information to the slave node (imag_raw) on the PC to get and display it. Actually, during the test, we find that the video streaming is stuck frequently and does not work well. We investigate the main cause and find that the required bandwidth for a color image (640 $\times$ 480) at 30fps is about 20MB/s, which is too much bandwidth usage and caused lag.

Thus, we add a node for node (usb_cam) to *compress* the image and send the image in compressed JPEG format, so that the compressed depth and color images only require about 2M/s bandwidth at the same time. Furthermore, to have a fixed compression rate for comparison with subsequent tests, we use the ROS system tool rqt_reconfigure to set the compression rate to 50%.

2) TCP BASED VIDEO STREAMING

The point-to-point distributed communication mechanism is the core of ROS. It uses TCP/IP-based communication to realize point-to-point loosely coupled connections between modules and can perform topic-based asynchronous data

TABLE 3. Comparison of communication types.

Communication Types	Advantages	Disadvantages
ROS Communications	Distributed, Peer-to-peer, Interface is independent of the programming language	Poor communication real time capability, Insufficient system stability, vulnerable to attacks
TCP Communications	Reliable and stable	Inefficient, High system resource usage, and vulnerable to attacks
UDP Communications	High communication speed	Unreliable, Poor stability, easy to lose packets

stream communication. TCP does not limit the size of the data to be transmitted, and there is a retransmission mechanism when the transmission fails, which can ensure the reliability of the transmission.

Since ROS publisher/subscriber method works on top of TCP, a custom TCP-based streaming method might work better than the ROS method. Thus, as the second video streaming method, we develop our TCP-based streaming service, which does not rely on ROS functionality. To be specific, we set up the server-side in Xycar and the client side in PC, bound both programs to IP ports, and used OpenCV to capture the camera images. TCP sends data as a byte stream. To reduce the impact of other factors, the image data in the communication we compressed by 50%, consistent with the ROS-based video streaming transmission.

3) UDP PROTOCOL BASED VIDEO STREAMING

TCP is very stable and has an ultra-low packet loss rate. But everything is double-sided, and stability will affect the transmission speed. Unlike TCP, UDP does not provide reliable data transfer functionality. The sender sends a data packet to a receiver, regardless of whether the receiver receives it or not. Therefore, compared to TCP, its packet loss rate may be higher, but the speed is faster.

UDP is suitable for real-time data transmissions, such as voice and video communications, because even if they occasionally lose one or two data packets, they will not have much impact on the reception result. For example, we occasionally lose one or two packets while watching a video and it does not affect the experience. A movie occasionally loses one or two frames, and our eyes simply cannot react, and the smoothness of the picture is not affected. Many webcasts currently use UDP to transmit images. To increase the transmission speed and reduce the transmission delay, we use the UDP protocol to replace the TCP protocol to control the data at the transmission layer for the forwarding of video data. But unlike TCP, UDP has strict restrictions on the size of the data sent, so we use data slicing and multi-threaded transmission.

Now we describe the data slicing and multi-threaded transmission in detail. We create a server-side on the Xycar side, a client-side on the PC side, and the server-side that uses OpenCV to capture the camera image. The original picture is very large (the size of 480p is $640 \times 480 \times 3 = 921,600$ bytes), even if the whole picture is compressed into jpg format, its size is also very large. However, UDP can only transmit data with a maximum size of 65,535

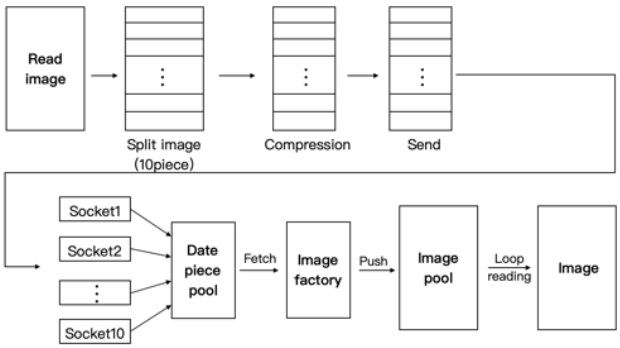

FIGURE 4. UDP transmission structure diagram.

bytes, so the picture is divided into blocks and the picture is divided horizontally. The advantage of this approach is that the number of pictures after segmentation can be one-to-one corresponding to the region (this thesis does not explore more complex image segmentation algorithms). Then we use multi-threading to compress the data (compression rate of 50%). The data is compressed into jpg format, the picture block data is numbered, and the corresponding data piece is updated. The test shows that in the experimental environment, the server-side in this article does not need to use the sending queue, and the newly generated frame can be transmitted by the Socket immediately. The client-side uses multi-threaded reception, each thread is a new connection, and the received data is stored in the data slice pool. Client-side opens another thread to repeatedly read data slices from the data slice pool, and update the screen according to the number of the data slice. Here, the screen is an array specially used for image display, and its dimension is 480p ($640 \times 480 \times 3$). The updated results are temporarily stored in the picture pool, and the main thread repeatedly reads and displays pictures from the picture pool. On the client-side, 10 threads are used for the asynchronous socket to receive data pieces. To ensure a smooth video effect at the receiving end, two queues are used to achieve secondary buffering of data reception. Fig. 4 is a schematic diagram of the structure of using UDP protocol to slice resources and perform multi-threaded transmission.

IV. MEASUREMENT RESULT AND ANALYSIS

In this section, we investigate the remote driving performance of various video streaming methods. We first describe the measurement test method and then analyze the test results.

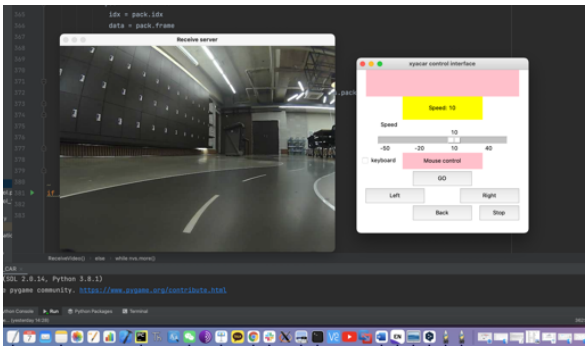

FIGURE 5. UI of test.

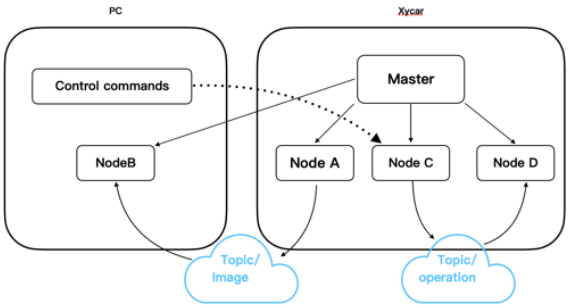

FIGURE 6. Publisher/subscriber model in ROS.

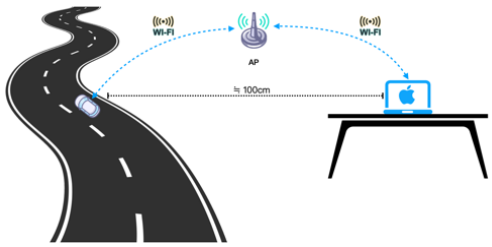

FIGURE 7. AP Wi-Fi.

A. REMOTE CONTROL TEST METHODOLOGY

In the test, to make the model vehicle acceptable for remote operation, we connect Xycar and MacBook Pro (PC) to the same wireless network, bound IPs, set Xycar as master and PC as a slave, and then connect both terminals. The system is assigned to the same Master. The slave can subscribe to messages from the Master.

We used "Tkinter" to design the operator interface for the remote driver (Fig. 5). The left window accepts images sent from the vehicle in real-time and displays them, while the right side is the operable window where the remote driver can send control commands to the vehicle. The remote driver can drag the progress bar to change the speed according to his will, choose the mouse to click the button to control the vehicle or choose the keyboard (W, A, S, D) to control the vehicle, and the right window displays the current vehicle speed and current vehicle dynamics (forward, left turn, right turn). In remote driving, the remote driver's maneuver commands must be received accurately by the vehicle, so the communication of maneuver commands uses TCP protocol. the ROS node C receives packets posted from the UI or keyboard converts the data information into ROS object types and publishes them locally. The local ROS node D running in the vehicle device system receives subject messages and these ROS messages are used to control the vehicle. The entire operation is performed by the vehicle sending live video to the control terminal, and the remote driver controls the vehicle through the terminal to form a closed loop.

We design and implement a middleware based on publisher/subscriber model Fig. 6 in ROS. Node A (camera) publishes image topics in the form of data, and node B subscribes to image topics. Node C receives vehicle operation command information and translates the data into ROS data information and publishes it, and Node D subscribes to vehicle operation command information.

The communication between the vehicle and the controller is done via Wi-Fi as shown in Fig. 7. Thus, the vehicle and the controller are at the same wireless LAN. Since the physical distance between the vehicle and the controller is only about 100cm, the latency of the communication is not very high.

On top of this remote driving architecture, we use the following three different video streaming methods, which will be described in detail.

1) CONTROL TEST BASED ON ROS COMMUNICATION

The control test based on ROS communication is as follows. We only use one Wi-Fi (Fig. 7) as the access source, connect Xycar and PC to Wi-Fi respectively, first run the central node Master in Xycar, and then run node A, node C, node D, and PC. The end runs node B and the client. You can see the video image and operation interface on the PC screen (Fig. 5), select the keyboard to control the vehicle to pass the test route, and set the speed to 10-50 (in increments of 5), each the speed was measured 10 times and the number of successes was recorded. This method we use is similar to the grid search method most used by Chui *et al.* [35].

2) CONTROL TEST BASED ON TCP PROTOCOL

For the control test based on TCP, we still use the same method to control the vehicle. We only change the method of video streaming to TCP (Fig. 8). We run the server-side that grabs the video on Xycar and the client-side that receives the video on the PC. We asked five remote drivers to control the vehicle on the PC to do the same pass test and recorded the number of successful passes on the test route.

3) CONTROL TEST BASED ON UDP PROTOCOL

For the control test based on UDP, we keep the vehicle control method unchanged. But the video transmission method is changed to UDP (Fig. 8), running a video capture client based on UDP protocol on Xycar and a video receiver based on UDP protocol on PC. Let five remote drivers control the vehicle through a PC to perform the same passing test.

So far the video quality of the above three methods is the same as 480p. Since the performance may depend on the data

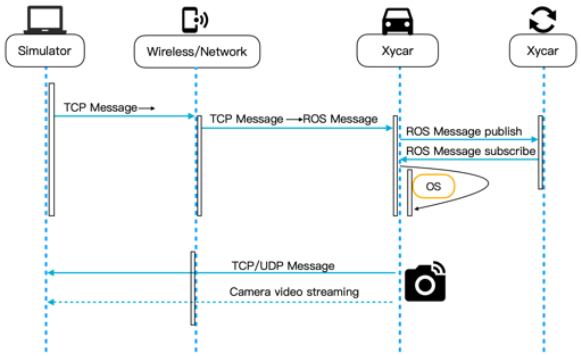

FIGURE 8. Chart of TCP/UDP information communication.

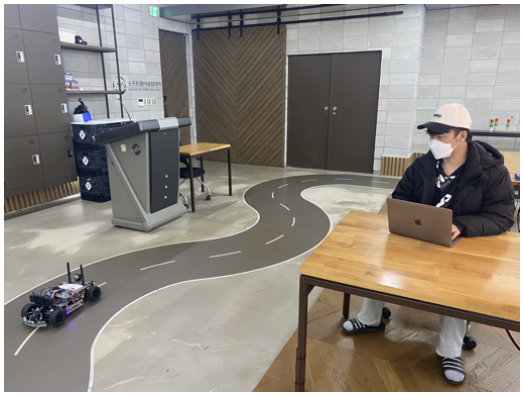

FIGURE 9. Control based on direct observation.

size, we vary the video quality to 720P (1280×720 pixels) for the UDP streaming method.

4) CONTROL TESTS BASED ON DIRECT OBSERVATION

To further feel the impact of network latency for remote driving, we add a set of experiments keeping the vehicle control method unchanged. Instead of controlling the vehicle based on the streaming video, we control the vehicle by directly observing the vehicle status with the made eye as shown in Fig. 9, and let the 5 remote drivers control the vehicle via PC to perform the same pass test, and record the number of successful pass test routes.

In this direct observation experiment, to see the impact of the different network environments, we replace the AP with iPhone hot spot. We still use the same test equipment and procedure, but change the AP with iPhone hot spot and do the same test for a total of 5 groups. The network connection in the iPhone11 (Wi-Fi 802.11ac) network environment is shown in Fig. 10.

B. RESULTS OF DIRECT OBSERVATION

First, we compare the results of the remote driving experiments by directly observing the vehicle status. In this experiment, there is no delay in observing the vehicle with naked eyes because we do not use any network video streaming methods. We only use two different network connections

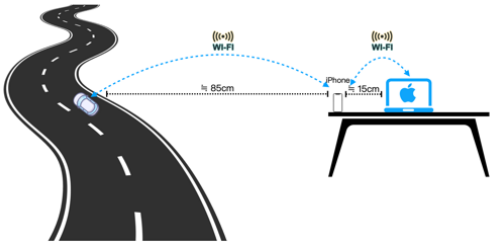

FIGURE 10. iPhone Wi-Fi.

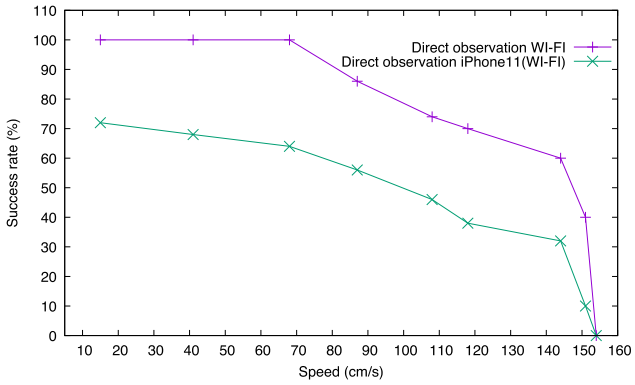

FIGURE 11. Results of remote driving experiments based on direct observation.

of the control path: Wi-Fi (802.11ax) and iPhone11 (Wi-Fi 802.11ac).

Because 802.11ax has a higher modulation scheme, changing from 802.11ac's 256 QAM to 1024 QAM, this means that more data is expressed per transmitted symbol, so the same amount of symbols are transmitted, i.e., a larger amount of data is transmitted. 256 QAM expresses 8 bits of information per symbol; 1024 QAM expresses 10 bits of information per symbol. So, at the same symbol rate, 1024 QAM capacity is increased by 25%. 802.11ax uses OFDMA (Orthogonal Frequency Division Multiple Access) modulations instead of OFDM (Orthogonal Frequency Division Multiplexing) modulation, reducing the subcarrier interval to 78.125 kHz, which is only 25% of the 802.11ac interval, and the symbols are 4 times longer than the latter. Therefore, 802.11ax is more efficient and stable, with better data transmission performance.

Fig. 11 shows the success rates averaged from 5 remote drivers over various speeds. The performance gap between the remote driving systems in the two network environments is around twenty percent, and the performance gap between the two decreases as the speed increases. When the speed reaches 50 (154cm/s), the success rate is zero for both. This strongly implies that in remote driving the design of network infrastructure may result in a significant performance difference. It should be noted that this experiment does not intend to claim that the iPhone Wi-Fi does not provide good performance. Rather we only show that different networks may show different performances.

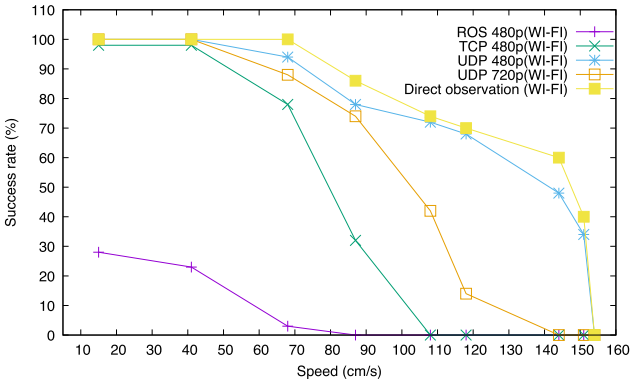

FIGURE 12. Comparison of the success rate of remote driving experiments based on different video transmission methods.

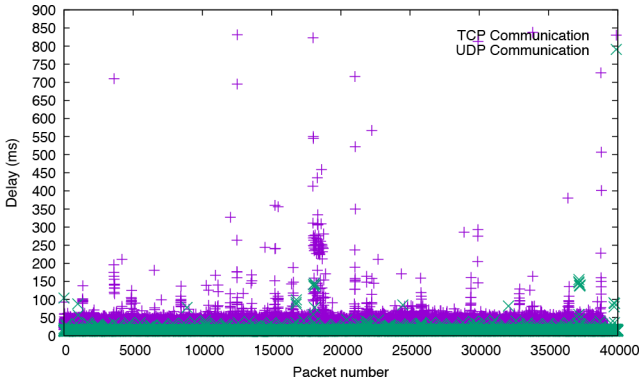

FIGURE 13. TCP vs UDP 480P video transmission latency comparison(WI-FI).

C. RESULT OF DIFFERENT VIDEO STREAMING METHODS

Now, we investigate the 5 different video streaming methods (including the one with naked eyes) in terms of success rate. Fig. 12 shows the success rates of remote driving experiments with different speeds observed in the WI-FI network environment. The success rates are the averages of the five remote drivers. We vary the video streaming methods among ROS, TCP, UDP (480p and 720p), and direct observation. As can be clearly seen in Fig. 11, when the vehicle speed is much higher, the latency requirement for the remote-control system is much higher, which requires not only low latency display of the screen but also low latency transmission of the remote control. Therefore, in this experiment, we keep the same remote control method (WI-FI 802.11ax) for all the streaming methods.

From the perspective of success rate (480P resolution), the ROS is 28%, TCP is 98% and UDP is 100% when the vehicle speed is 10 (15cm/s). When the vehicle speed increases to 20 (68cm/s), the success rate of ROS plummets to 3%, meanwhile, TCP is 78%, and UDP is 94%. When the speed is 25 (87cm/s), the success rate of ROS is 0. When the speed is 30 (108cm/s), the TCP success rate also plummeted to 0, UDP still has a success rate of 78%. When the speed is greater than 45 (151cm/s), the UDP success rate is 0. Even if we increase the amount of video data transferred by UDP by a factor of 3 (720P resolution), at a speed of 30 (108cm/s), TCP has a success rate of 0, while UDP still has a success rate of 42%, and at a speed of 40 (144cm/s), UDP's success rate drops to 0. We can see that UDP has a higher success rate than TCP at all speeds. The success rate of UDP at each speed is closest to the experimental results based on direct observation. This means that the UDP streaming method is not affected by the streaming delay much.

According to Fig. 12, it is easy to find the real-time video streaming methods by TCP and UDP perform significantly better than the ROS method in the experiment. Furthermore, comparing the results of UDP with the TCP method, the UDP is even better. We claim that we should use UDP based video streaming method in remote driving.

D. TCP VS. UDP VIDEO TRANSMISSION PERFORMANCE COMPARISON

Since we can find higher success rates for the video streaming methods using TCP and UDP, we further investigate the main reason for such high performance compared to ROS. For that matter, we record and compare the latency times of both TCP and UDP for transmitting 480P resolution video in a Wi-Fi network environment. To calculate the overall delay, we calculated the time from the start of video capture until the packet is received by the receiver and compile into an image cutoff. The time of the newly generated image is subtracted from the time of the last generated image to derive the delay time so that there is no problem calibrating the time difference between the two machines. The results are shown in Fig. 13.

We can see from Fig. 13 that most of the video transmission delays based on TCP and UDP are lower than 50ms, but there are many delays greater than 50ms in TCP-based video streaming transmission, and although UDP-based video transmission also has delay jumps, the magnitude and frequency of TCP transmission delay changes are much greater than that of UDP. We can clearly see that the main cause of performance degradation is the delay of the video transmission.

V. DISCUSSION

Now, we discuss some of the observations that are worth to mention. In the test, the test route design has a longer distance, consisting of five semi-circular paths, which can better reflect the remote stability of network delay during driving. In summary, as can be seen in Fig. 12, we can conclude that the remote driving performance of ROS-based multi-machine communication for transmitting video streams is extremely poor. The UDP-based method slices the resources and the multi-threaded transmission method has the strongest performance. The TCP-based method is slightly inferior to the UDP-based method in terms of video transmission performance. Although the communication mechanism of ROS is based on TCP/IP, the real-time performance and stability of this communication mechanism are not good, and it strongly depends on the central node ROS Master. For example, during

the remote driving experiment, we encountered many problems like the followings:

- At a certain point during the experiment (the time is uncertain), ROS Master went down unexpectedly, the entire vehicle system was no longer under control, and sudden node failures sometimes occurred.
- When there are a large number of topics and data transmission in the ROS system, the local transmission data delay is large and uncertain, and the remote transmission data is often affected by bandwidth and processing performance.
- The real-time communication is poor, and there is no way to achieve millisecond-level remote driving vehicle control.

ROS has such shortcomings but it is still adopted by Baidu and Auto-ware because, although the two autonomous driving platforms of Apollo and Auto-ware are developed based on ROS, they have also undergone a lot of optimization in the ROS communication mechanism. Apollo 3.5 and later versions replace the original ROS middleware and use its CyberRT middleware. Although ROS has certain shortcomings, it is still the platform used by most users to implement autonomous driving technology compared to other platforms.

In the test of remote control of the vehicle, our remote driver felt the difficulty of remote control caused by the unstable delay time. Furthermore, the performance may depend on the characteristics of the communication network. For example, Wi-Fi (802.11ax) and iPhone11 (Wi-Fi 802.11ac) network show different results. The UDP transmission capacity is very impressive, and the process increases the data processing capacity.

In this experiment, participants remotely drive a vehicle through a test route under different video transmission methods while experiencing different communication delays. We design the remote driving system to transmit surveillance video streams with 640 $\times$ 480 pixel resolution and less than 50 ms latency in a commercial Wi-Fi network environment. Real-time video observation of remote driving is important for remote drivers to perform emergency maneuvers.

The increase in vehicle speed deteriorates the driver's operational stability, especially when driving around curves, making the vehicle extremely difficult to navigate and even out of control. When the vehicle is moving fast, the driver's ability to recognize space is weakened, and the driver's reaction speed is required to be higher because the driver needs to make judgments and operation in a very short time. In our experiments, different video transmission methods at the same speed respond well to the effect of delay on operating performance at the current speed.

Although participants show some capacity to adapt to all forms of delay in this environment, our evaluation scheme does well (i.e., it passes the manipulation performance evaluation regardless of the instantiation of delay). The stable delay is extremely easy for the driver to adapt to, and it is the jump delay that causes the vehicle to lose control in the experiment. Therefore, these experimental results suggest that an appropriate approach to improve remote control performance in remote driving vehicle control is to reduce the degradation of driver posture perception in a limited hardware environment (communication delay). We believe that virtual driving technology based on the combination of Metaverse technology [36] and remote driving technology can reduce the decrease in driver perception. The virtual driving system mainly consists of the control system, vision simulation system, driving simulator, and other components such as audio equipment and display. Among them, the control system is physical, including the steering wheel, gas pedal, brake pedal, and shift lever, and the driver changes the motion of the vehicle in the virtual environment through the control system, and the visual simulation system solves the vehicle posture and transmits the vehicle information to the driver through the monitor so that the driver can perceive the vehicle motion posture and road environment changes in time, the driving simulator adjusts the vehicle motion posture according to the changes in the driving simulator adjusts the orientation and angle according to the change of the vehicle's posture, so that the driver's posture is consistent with the body posture, enhancing the experience and authenticity.

In our tests, we consider a success rate of 90% or higher to be the minimum guarantee for selecting remote driving, meaning that we do not recommend selecting a remote driving system to remotely control a vehicle through a roadway with extremely difficult access when the model vehicle speed reaches 20 (68cm/s) or more.

The experience obtained in our experiments can be scaled to other telerobotic applications. However, several factors must be considered for doing so. The first is the difficulty of the task being performed by the robot. If the robot is executing a relatively simple task, such as traversing an open area, the delay may not have a significant impact, whereas moving through dense obstacles may be severely affected by the delay. Next is the speed of the robot's motion, where latency affects the robot more like the speed of motion increases. The results of this paper show that the control performance increases with network bandwidth and the requirement for communication delay time increases with vehicle speed.

VI. CONCLUSION

In the actual control process, the situation faced by a remotely driven vehicle is very complicated, and the millisecond delay is just enough. From the perspective of ensuring safety, the system response time is of course as low as possible, and the requirements for communication delay will be higher. In this article, we conduct a large number of remote control simulation measurement experiments based on the ROS middleware platform to understand the impact of delay on remote driving in different network environments. To ensure the reliability of the experiment, we had five remote drivers perform a total of more than twenty hundred experiments. We record the test data separately and compare and analyze them. The test results show that it is feasible to use the ROS system for

better real-time remote driving, but the premise is to ensure the transmission of real-time video streams.

Below we can answer the questions we asked in the goal:

- Test results show that as the speed of the vehicle increases, the remote driving performance degrades. It strongly implies that remote driving should be used in a low-speed environment.
- We use an improved transmission method based on the UDP protocol, sending through image slicing and receiving asynchronously using a multi-threaded socket, which improves transmission efficiency and reduces transmission latency. 2 queues are used to achieve secondary buffering for data reception, ensuring a smooth video effect at the receiving end. We prove that the transmission performance of UDP is very high, and UDP can better provide high-quality and low-latency video transmission services for remote driving.
- During our experiments, all drivers showed a strong ability to adapt to delays, but still could not avoid operation errors, mostly due to the sudden lag of the screen that caused the vehicle operation to go out of control. The experimental results show that the jump delay is the biggest cause of control errors. The value of delay in remote driving is not simply to pursue low, but more importantly, to pursue ''deterministic delay''. Deterministic latency means that the latency value is in a stable state. In most of the remote scenarios, what is pursued is not the extremely low latency, but the stability of the latency. For example, in remote surgery and remote driving, if there is a delay jitter, the consequences will be fatal. And all five remote drivers who participated in the test all believed that the delay in jumping was the biggest cause of screen jams and consequent control errors. The results of the remote driving experiment suggest that any degradation of interaction beyond the survival level has a significant negative impact on performance in terms of remote driving.

In the experiments, we applied the ROS (not ROS2) version of the ROS system. Although the communication of ROS is also based on TCP/IP, it cannot achieve millisecond vehicle control due to the strong communication capability of the central node ROS Master, which has poor real-time stability.

Due to practical limitations, we know that the delay time of ROS information transmission in the local system is significantly less than the delay of multi-machine communication. This paper cannot comprehensively review the delay of the remote driving system driven in the actual network environment of Wi-Fi. The data collected in our constrained environment does not contain all contributing factors (road geometry, environment, vehicle, etc. related factors) that affect the strain variables, which also affect remote driving performance, therefore, video transmission delay is not the only factor affecting remote driving performance in our study, there may be other potential influences that are not observed by us, and the omission of these hard-to-get factors could potentially lead to unobserved heterogeneity. Since the vehicle, we remotely control over Wi-Fi networks is a Xycar-D model car, it cannot fully represent the remote operation performance of a real vehicle under actual conditions. The field study of real-time delay detection is beyond the scope of this study, and with each time the random vehicle starts, the preparation time in the test makes it difficult to evaluate the current state of the vehicle's ROS system, and there is a certain error in the vehicle's true speed. Readers should keep in mind that this research is based on a small number of remote driving examples based on Wi-Fi laboratory network conditions. In future research, we will improve the program, combine virtual reality technology, further improve the performance of real-time video transmission, and improve a certain degree of autonomy of the vehicle, through the virtual environment to complete the full range of observation and control of unmanned vehicles, to achieve virtual remote driving of vehicles, not only can effectively solve the current unmanned vehicles facing the lack of environmental recognition, vehicle autonomy control inaccuracy and other problems, but also can ensure vehicle safety, but also Make the operator have the feeling of immersive driving experience.

REFERENCES

[1] T. Litman, *Autonomous Vehicle Implementation Predictions*. Victoria, BC, Canada: Victoria Transport Policy Institute, 2017.

[2] Y. Ouerhani, A. Alfalou, M. Desthieux, and C. Brosseau, ''Advanced driver assistance system: Road sign identification using VIAPIX system and a correlation technique,'' *Opt. Lasers Eng.*, vol. 89, pp. 184–194, Feb. 2017.

[3] N. Martelaro and W. Ju, ''WoZ way: Enabling real-time remote interaction prototyping & observation in on-road vehicles,'' in *Proc. Companion ACM Conf. Comput. Supported Cooperat. Work Social Comput.*, Feb. 2017, pp. 169–182.

[4] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, ''Advanced driver-assistance systems: A path toward autonomous vehicles,'' *IEEE Consum. Electron. Mag.*, vol. 7, no. 5, pp. 18–25, Sep. 2018.

[5] A. Driving, ''Levels of driving automation are defined in new SAE international standard J3016: 2014,'' *SAE International: Warrendale, PA, USA*, vol. 1, 2014.

[6] S. Neumeier, P. Wintersberger, A.-K. Frison, A. Becher, C. Facchi, and A. Riener, ''Teleoperation: The holy grail to solve problems of automated driving? Sure, but latency matters,'' in *Proc. 11th Int. Conf. Automot. User Interfaces Interact. Veh. Appl.*, Sep. 2019, pp. 186–197.

[7] Z. Fallah, V. K. Shukla, and M. N. Khalid, ''Redefining safety in autonomous vehicle through remote teleoperation,'' in *Computational Intelligence in Pattern Recognition*. Cham, Switzerland: Springer, 2022, pp. 219–230.

[8] L. Kang, W. Zhao, B. Qi, and S. Banerjee, ''Augmenting self-driving with remote control: Challenges and directions,'' in *Proc. 19th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2018, pp. 19–24.

[9] S. Higginbotham, ''Autonomous trucks need people [opinion],'' *IEEE Spectr.*, vol. 56, no. 3, p. 21, Feb. 2019.

[10] A. Joglekar, A. Rawat, A. G. Colaco, P. Ranjan, R. Doreswamy, R. Chopra, A. Bharadwaj, B. Govindraju, F. Rahman, H. Tyagi, N. Arulselvan, P. Patil, S. V. R. Anand, and V. Sevani, ''Tele-driving an electric vehicle over a private LTE network,'' in *Proc. 14th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2022, pp. 441–443.

[11] A. Hosseini and M. Lienkamp, ''Enhancing telepresence during the teleoperation of road vehicles using HMD-based mixed reality,'' in *Proc. 4th IEEE Intell. Vehicles Symp.*, Jun. 2016, pp. 1366–1373.

[12] N. Sendhil Kumar, U. Kaur, T. Anuradha, S. Majji, S. R. Karanam, and R. G. Deshmukh, ''5G network virtualization for the remote driving enhancement,'' in *Proc. 4th Int. Conf. Comput. Commun. Technol. (ICCCT)*, Dec. 2021, pp. 458–463.

[13] R. Liu, D. Kwak, S. Devarakonda, K. Bekris, and L. Iftode, "Investigating remote driving over the LTE network," in *Proc. 9th Int. Conf. Automot. User Interfaces Interact. Veh. Appl.*, Sep. 2017, pp. 264–269.
[14] J. Davis, C. Smyth, and K. McDowell, "The effects of time lag on driving performance and a possible mitigation," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 590–593, Jun. 2010.
[15] G. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," *Update*, vol. 2017, p. 2022, Feb. 2019.
[16] G. Gualdi, A. Prati, and R. Cucchiara, "Video streaming for mobile video surveillance," *IEEE Trans. Multimedia*, vol. 10, no. 6, pp. 1142–1154, Oct. 2008.
[17] R. Cucchiara, "Multimedia surveillance systems," in *Proc. 3rd ACM Int. Workshop Video Surveill. Sensor Netw.*, 2005, pp. 3–10.
[18] Q. Xiao, K. Xu, D. Wang, L. Li, and Y. Zhong, "TCP performance over mobile networks in high-speed mobility scenarios," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, Oct. 2014, pp. 281–286.
[19] F. Chucholowski, T. Tang, and M. Lienkamp, "Teleoperated driving robust and secure data connections," *ATZelektronik Worldwide*, vol. 9, no. 1, pp. 42–45, Feb. 2014.
[20] K. Jang, M. Han, S. Cho, H.-K. Ryu, J. Lee, Y. Lee, and S. B. Moon, "3G and 3.5G wireless network performance measured from moving cars and high-speed trains," in *Proc. 1st ACM Workshop Mobile Internet Through Cellular Netw.*, 2009, pp. 19–24.
[21] J. A. Gutierrez, E. H. Callaway, and R. L. Barrett, *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors With IEEE 802.15.4*. Piscataway, NJ, USA: IEEE Standards Association, 2004.
[22] J. Suhonen, M. Kohvakka, V. Kaseva, T. D. Hämäläinen, and M. Hännikäinen, *Low-Power Wireless Sensor Networks: Protocols, Services and Applications*. Cham, Switzerland: Springer, 2012.
[23] S. S. Shadrin and A. A. Ivanova, "Analytical review of standard SAE J3016 taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles with latest updates," *Avtomobil. Doroga. Infrastruktura.*, vol. 3, no. 21, p. 10, 2019.
[24] F. Graf, *Apollo*. Evanston, IL, USA: Routledge, 2008.
[25] S. Gibbs, "Google sibling waymo launches fully autonomous ride-hailing service," *Guardian*, vol. 7, p. 1, Nov. 2017.
[26] V. A. Banks, K. L. Plant, and N. A. Stanton, "Driver error or designer error: Using the perceptual cycle model to explore the circumstances surrounding the fatal Tesla crash on 7th May 2016," *Saf. Sci.*, vol. 108, pp. 278–285, Oct. 2018.
[27] A. Efrati, "Uber finds deadly accident likely caused by software set to ignore objects on road," *Information*, vol. 5, p. 7, May 2018.
[28] J. P. Luck, P. L. McDermott, L. Allender, and D. C. Russell, "An investigation of real world control of robotic assets under communication latency," in *Proc. 1st ACM SIGCHI/SIGART Conf. Hum.-Robot Interact.*, 2006, pp. 202–209.
[29] R. P. Darken, K. Kempster, and B. Peterson, "Effects of streaming video quality of service on spatial comprehension in a reconnaissance task," Dudley Knox Library, Monterey, CA, USA, Tech. Rep. 10945-46077, 2001.
[30] K. Kato, K. Suto, and K. Sato, "Deterministic video streaming with deep learning enabled base station intervention for stable remote driving system," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2021, pp. 1–6.
[31] G. Prokop, T. Tüschen, N. Eisenköck, and J. Bönninger, "Highly immersive driving simulator for scenario based testing of automated driving functions," in *Proc. Internationales Stuttgarter Symp.* Cham, Switzerland: Springer, 2022, pp. 145–154.
[32] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, 2009, vol. 3, nos. 3–2, p. 5.
[33] ThemeIsle. *XYCAR-D*. Accessed: 2021. [Online]. Available: http://xytron.co.kr/?page_id=500
[34] B. G. P. So. *NASA-TLX*. Accessed: 2022. [Online]. Available: http://humansystems.arc.nasa.gov/groups/tlx/
[35] K. T. Chui, R. W. Liu, M. Zhao, and P. O. D. Pablos, "Predicting students' performance with school and family tutoring using generative adversarial network-based deep support vector machine," *IEEE Access*, vol. 8, pp. 86745–86752, 2020.
[36] S. Mystakidis, "Metaverse," *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.

YANG YU received the B.S. degree in computer engineering from Qingdao University, Qingdao, China, in 2018, and the M.S. degree in computer science from Kookmin University, Seoul, Republic of Korea, in 2021, where he is currently pursuing the Ph.D. degree in computer science. His current research interests include autonomous driving, remote driving, the Internet-of-Things (IoT), and video streaming. (yuyang@kookmin.ac.kr).

SANGHWAN LEE received the B.S. and M.S. degrees from Seoul National University, South Korea, in 1993 and 1995, respectively, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota, in September 2005. After his M.S. degree, he worked at Hyundai Electronics, South Korea, for five years. From June 2005 to February 2006, he worked at the IBM T. J. Watson Research Center. He joined Kookmin University, Seoul, South Korea, in March 2006. His research interests include software-defined networking (SDN), scalable routing selection for multimedia, and various theory and services needed in internet. (sanghwan@kookmin.ac.kr).