Cooperative Automated Driving of JetRacer using Server and CCTV

임베디드시스템 최종발표

2019043836 김인호
2024152152 한정운

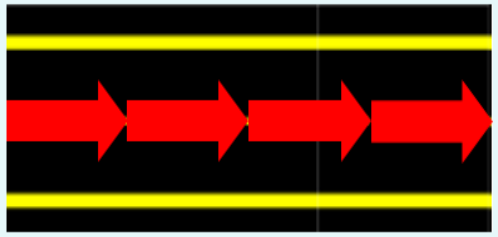
목차

01
라인트레이싱 개선

주행 알고리즘 개선

Smooth Driving

실제 자율 주행 차량에 가까운 Lane tracking system



Drive & Sleep

평가기준(Task1 & Task2)을 만족시키는 JetRacer를 만드는 것에 중점. 주행과 정지를 반복하는 drive & sleep 알고리즘을 선택

PD Control

실제 응용분야에서 가장 많이 사용되는 대표적인 제어 기법. 이동체의 output을 측정하여 오차를 계산하고 해당 오차 값을 gain을 통해 제어.

알고리즘 구현

```
# JoyStick
if joystick.get_button(11):
    Mode = 0
    continue
if joystick.get_button(6):
    ThrottleNormal = ThrottleNormal + 0.01
if joystick.get_button(7):
    ThrottleNormal = ThrottleNormal - 0.01
```

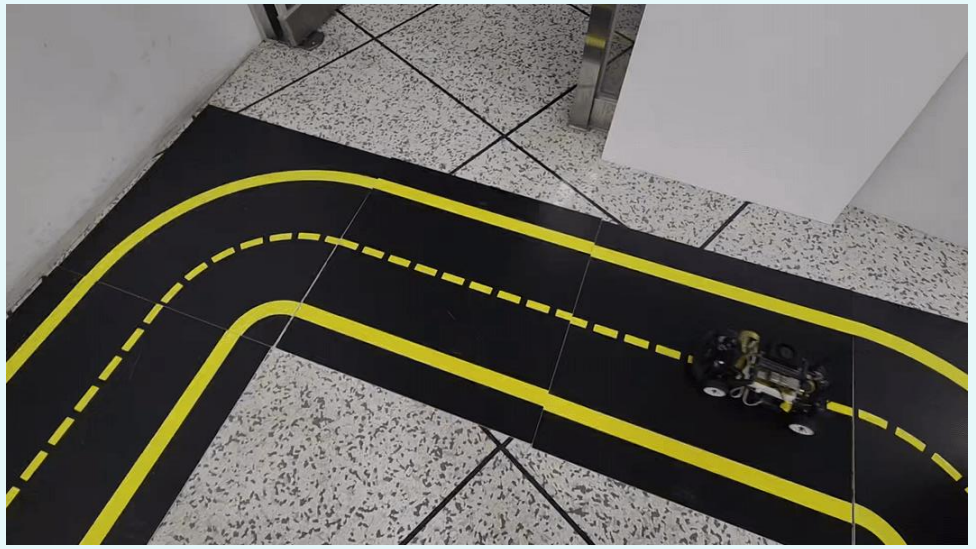
- **Throttle**
  - ✓ throttle 값을 고정해도 부팅 마다 차량 속도가 달라지는 문제 발생
  - ✓ R1, L1에 throttle up, down을 할당하여 주행 전 최적 속도 반영

```
# Function PD Controller
Kp = 1.0
Kd = 0.05
dt = 0.1
pre_error = 0
def PDController(error):
    global pre_error
    P = Kp * error
    derivative = (error - pre_error) / dt
    D = Kd * derivative
    steering = (P - D) * 0.0025
    steering = max(min(steering, 1), -1)
    pre_error = error
    return steering
```

- **Steering**
  - ✓ PD제어기를 통해 차량 횡방향 제어
  - ✓ 이미지 중앙(x=320)을 기준으로 gain설정
  - ✓ 시행착오를 통해 이미지 라벨링 y값(100) 및 제어 계수 설정

시연 영상

Video – Circuit Driving



✓ Throttle과 PD control 알고리즘을 통해 circuit에서 driving stability 확보

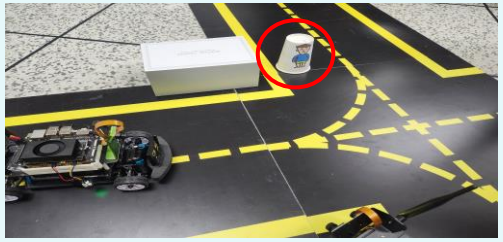
02

자율협력주행 구현

자율협력주행의 필요성

자율협력주행

감지범위 밖의 데이터를, V2X 통신을 통해 전달받아, 더 넓은 정보로 안전한 자율주행을 만드는 기술



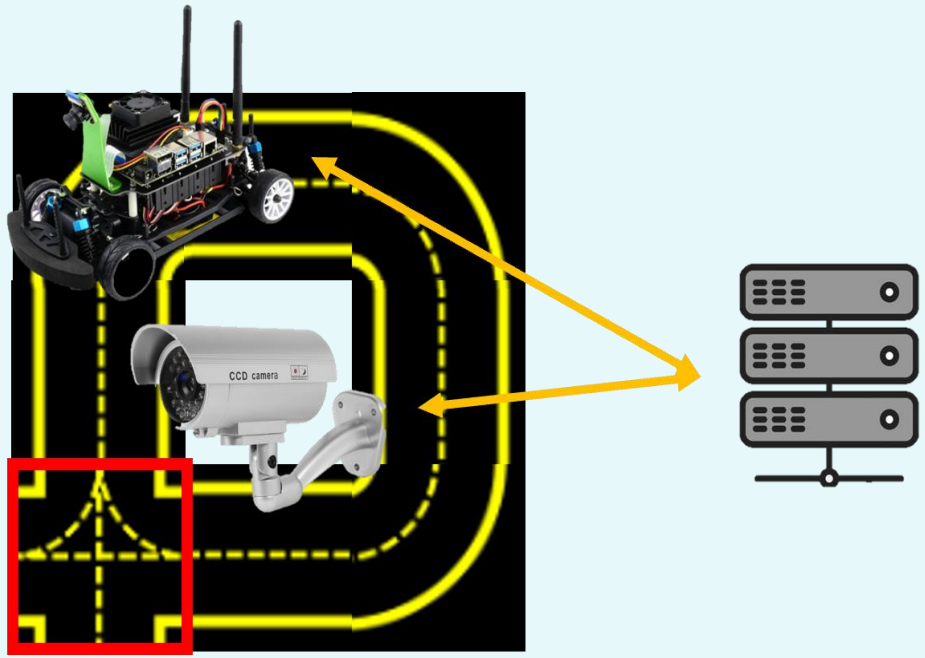
사각지대 정보 파악

교차로 회전시 건물이나 벽에 가려 차량이 볼 수 없는 장애물을, CCTV를 통해 인지하고 이를 전달해줄 수 있음.

표지인식 한계 극복

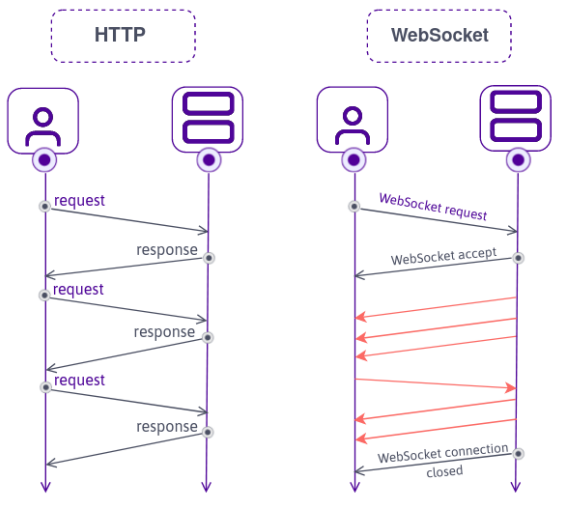
움직이는 차량이 표지를 인식하는 것보다는, 정지된 CCTV가 상황을 인식하는 것이 더욱 안정적임.

자율협력주행 구현 목표



- **CCTV**
  - ✓ 교차로 상황 인식 후 서버로 전달 (없음, 보행자, 공사)
  - ✓ 번호판 인식으로 교차로 차량 정보를 서버에 보냄
- **JetRacer**
  - ✓ 서버로부터 교차로 상황을 전달받음
  - ✓ 상황에 따라 주행 (좌회전, 정지, 우회전)
- **Server**
  - ✓ CCTV/JetRacer 간 데이터 교환 중재
  - ✓ 해당 차량 번호를 가진 JetRacer를 찾아 상황을 전달

웹소켓 프로토콜



Server

```
# 서버의 웹소켓/IP/Port 지정
server = websockets.serve(code, "192.168.0.206", 8000)
```

Client(CCTV/JetRacer)

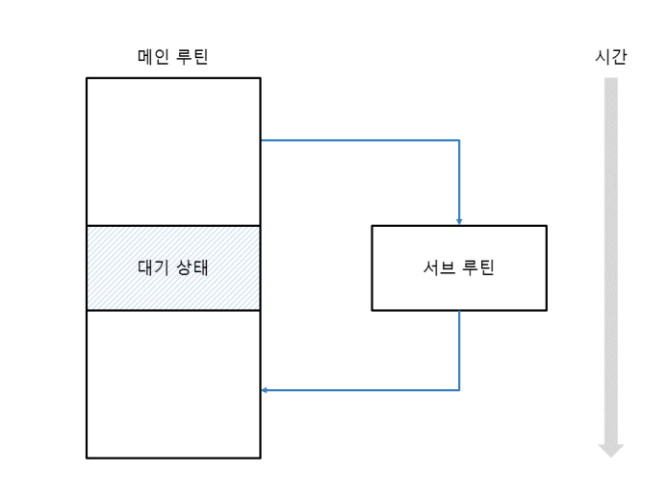
```
uri = "ws://192.168.0.206:8000"
async with websockets.connect(uri) as websocket:
    await websocket.send("Camera")
    connection = await websocket.recv()
    print("From Server :", connection)
```

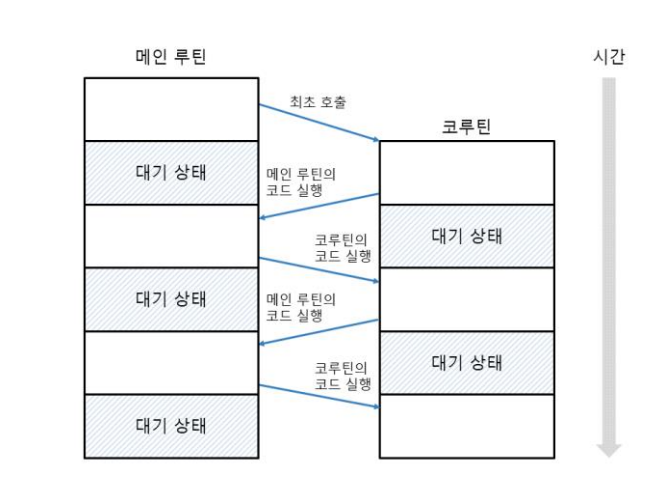
실시간 통신을 보장하는 HTTP 확장 프로토콜

- ✓ 서버와 클라이언트가 Ping과 Pong을 지속적으로 주고받아, 연결이 끊기지 않고 실시간으로 유지됨.
- ✓ 파이썬 모듈을 이용하여, 웹소켓 로컬 서버를 열고, 서버의 IP&Port를 입력하여 클라이언트가 연결.

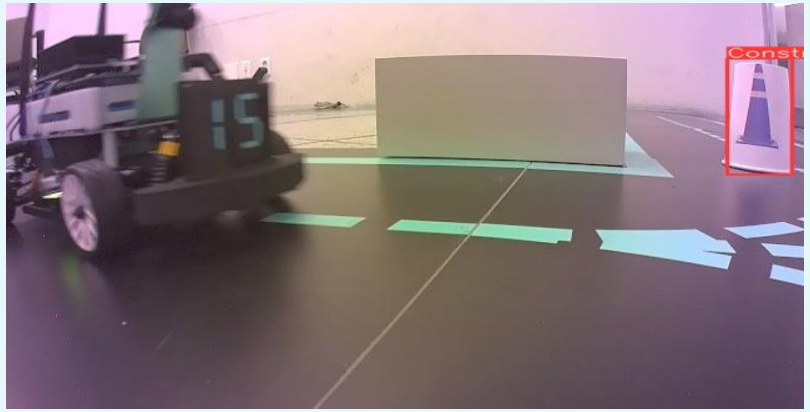
서버의 비동기 프로그래밍

- ✓ 서버에서 클라이언트마다 각각의 코루틴을 할당.
- ✓ 코루틴간에는, 특정 코루틴이 Wait 상태가 되었을 때, 다른 코루틴으로 흐름이 넘어가 이전 상태에서 이어서 진행.
- ✓ 멀티스레드와 유사한 동시성을 가지면서, 자원을 적게 쓰고 공유변수 동기화 문제가 적기때문에, 클라이언트들이 올바른 순서로 빠르게 데이터를 교환하도록 서버의 중재가 가능.



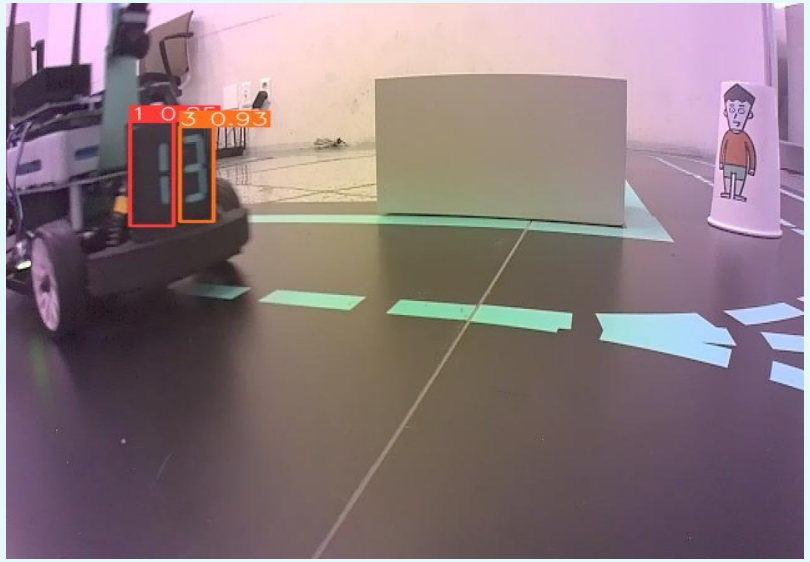


CCTV의 인식 모델



Situation Detection

- ✓ Colab에서 Yolo를 이용해 학습 진행.
- ✓ 사각지대를 보면서, 상황이 없는 경우, 보행자가 있는 경우, 공사 칼라콘이 있는 경우를 구분.



Car Number Detection

- ✓ Colab에서 Yolo를 이용해 학습 진행.
- ✓ 1~5로 이루어진 숫자판 2개를 이용하여, 25가지의 숫자에 대한 구분이 가능하도록 모델을 제작.
- ✓ 25개의 클래스가 아닌, 한 숫자로 2개를 동시에 인식한 뒤, 박스의 X좌표를 이용하여 올바른 순서로 번호를 조정.

시연 영상

JetRacer 번호 지정

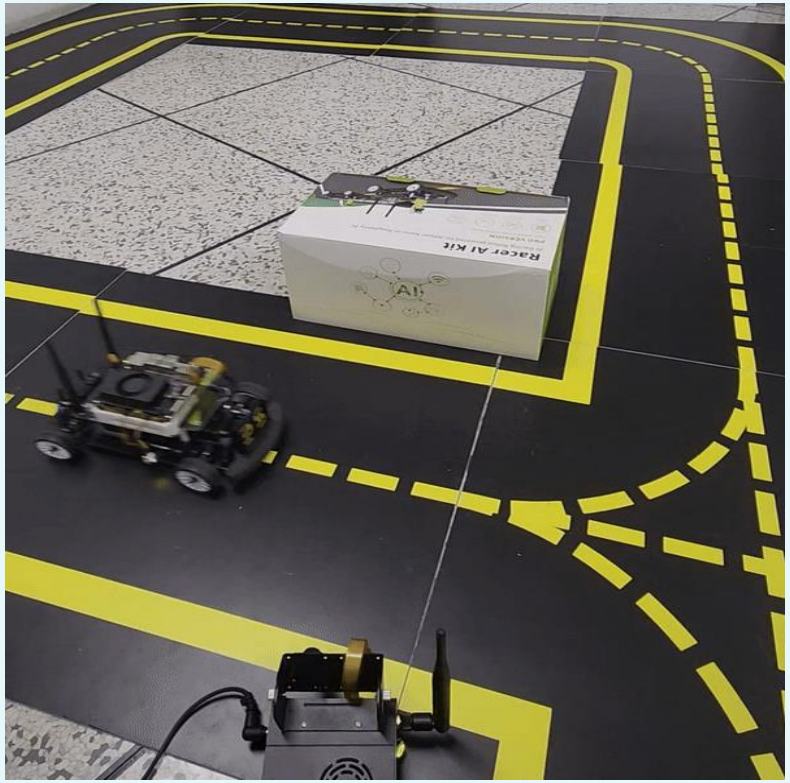
```
##### Connect #####
global Mode, Lock, ThrottleNorm
uri = "ws://192.168.0.206:8000"
async with websockets.connect(u
    await websocket.send("25")
    connection = await websocke
    print("From Server :", conn
```

서버에 해당 차량 등록

```
PS C:\Users\INNO\바탕 화면> code .
PS C:\Users\INNO\바탕 화면> cd ..
PS C:\Users\INNO> cd .\Git\
PS C:\Users\INNO\Git> cd .\Project\
PS C:\Users\INNO\Git\Project> python .\Server.py
From 25 : Connected
```

시연 영상

없음 → 좌회전 통과

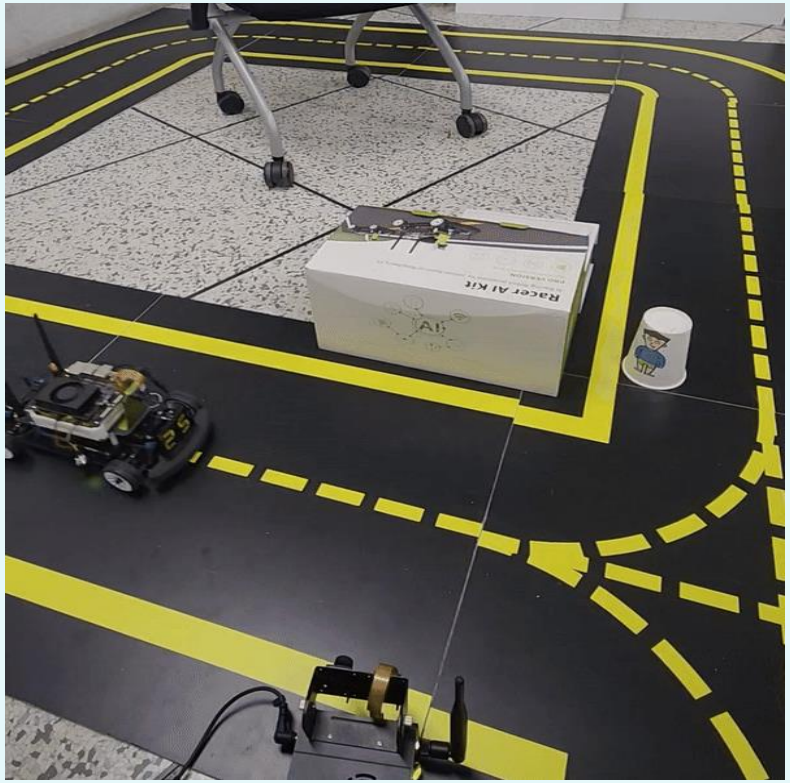


서버 데이터 로그

```
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
From Camera : 25/
```

시연 영상

사각지대 보행자 → 정지

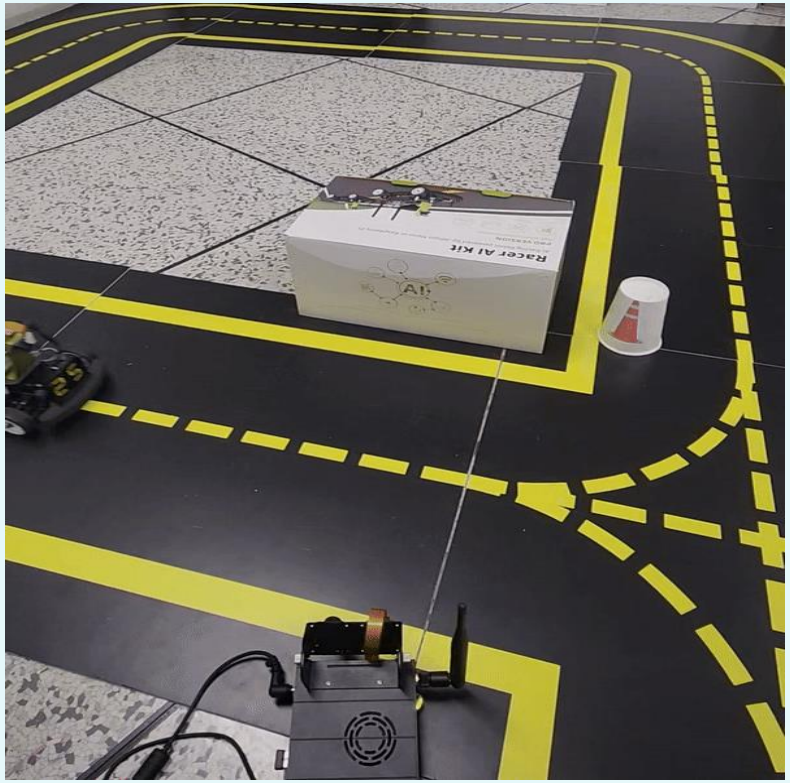


서버 데이터 로그

```
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
From Camera : 25/Pedestrian
To 25 : Pedestrian
```

시연 영상

사각지대 칼라콘 → 우회



서버 데이터 로그

```
From Camera : /Construction
From Camera : /Construction
From Camera : /Construction
From Camera : /Construction
From Camera : /Construction
From Camera : /Construction
From Camera : 25/Construction
To 25 : Construction
```

서버 처리 속도 계산

No.	Time	Source	Destination	Protocol	Length	Info
3	0.079126	192.168.0.221	192.168.0.206	TCP	76	51066 → irdmi(8000) [PSH, ACK] Seq=1 Ack=1 Win=501 Len=10 TSval=881177174 TSe
4	0.079541	192.168.0.206	192.168.0.221	TCP	69	irdmi(8000) → 51066 [PSH, ACK] Seq=1 Ack=11 Win=8194 Len=3 TSval=1138271431 T
5	0.082473	192.168.0.221	192.168.0.206	TCP	66	51066 → irdmi(8000) [ACK] Seq=11 Ack=4 Win=501 Len=0 TSval=881177178 TSecr=11
6	0.183635	192.168.0.221	192.168.0.206	TCP	76	51066 → irdmi(8000) [PSH, ACK] Seq=11 Ack=4 Win=501 Len=10 TSval=881177279 TS
7	0.184036	192.168.0.206	192.168.0.221	TCP	69	irdmi(8000) → 51066 [PSH, ACK] Seq=4 Ack=21 Win=8194 Len=3 TSval=1138271536 T
8	0.186676	192.168.0.221	192.168.0.206	TCP	66	51066 → irdmi(8000) [ACK] Seq=21 Ack=7 Win=501 Len=0 TSval=881177282 TSecr=11
9	0.286069	192.168.0.221	192.168.0.206	TCP	76	51066 → irdmi(8000) [PSH, ACK] Seq=21 Ack=7 Win=501 Len=10 TSval=881177381 TS
10	0.286506	192.168.0.206	192.168.0.221	TCP	69	irdmi(8000) → 51066 [PSH, ACK] Seq=7 Ack=31 Win=8194 Len=3 TSval=1138271638 T
11	0.290440	192.168.0.221	192.168.0.206	TCP	66	51066 → irdmi(8000) [ACK] Seq=31 Ack=10 Win=501 Len=0 TSval=881177385 TSecr=1
12	0.394909	192.168.0.221	192.168.0.206	TCP	76	51066 → irdmi(8000) [PSH, ACK] Seq=31 Ack=10 Win=501 Len=10 TSval=881177490 T
13	0.395307	192.168.0.206	192.168.0.221	TCP	69	irdmi(8000) → 51066 [PSH, ACK] Seq=10 Ack=41 Win=8194 Len=3 TSval=1138271747
14	0.401336	192.168.0.221	192.168.0.206	TCP	66	51066 → irdmi(8000) [ACK] Seq=41 Ack=13 Win=501 Len=0 TSval=881177493 TSecr=1
17	0.446143	192.168.0.62	192.168.0.206	TCP	73	47922 → irdmi(8000) [PSH, ACK] Seq=1 Ack=1 Win=501 Len=7 TSval=1012446850 TSe
18	0.446453	192.168.0.206	192.168.0.62	TCP	72	irdmi(8000) → 47922 [PSH, ACK] Seq=1 Ack=8 Win=8193 Len=6 TSval=1138271798 TS
19	0.449184	192.168.0.62	192.168.0.206	TCP	66	47922 → irdmi(8000) [ACK] Seq=8 Ack=7 Win=501 Len=0 TSval=1012446853 TSecr=11
20	0.458410	192.168.0.62	192.168.0.206	SSH	146	Server: Encrypted packet (len=92)
21	0.465130	192.168.0.206	192.168.0.62	SSH	146	Client: Encrypted packet (len=92)

- ✓ Wireshark 프로그램을 통해, 상황 전달시 서버로 인한 오버헤드 측정.
- ✓ 약 0.05s의 시간차로, 통신에 의한 오버헤드는 굉장히 영향이 미비함.

일반주행과 자율협력주행의 비교

일반주행	1	2	3	4	5	6	7	8	9	10	Total
없음	O	O	O	O	O	O	O	O	O	O	100%
보행자	X	X	O	X	X	X	X	X	O	X	20%
공사	X	X	X	X	X	X	X	X	X	X	0%

사각지대 장애물에 대한 대응 성능이 크게 향상

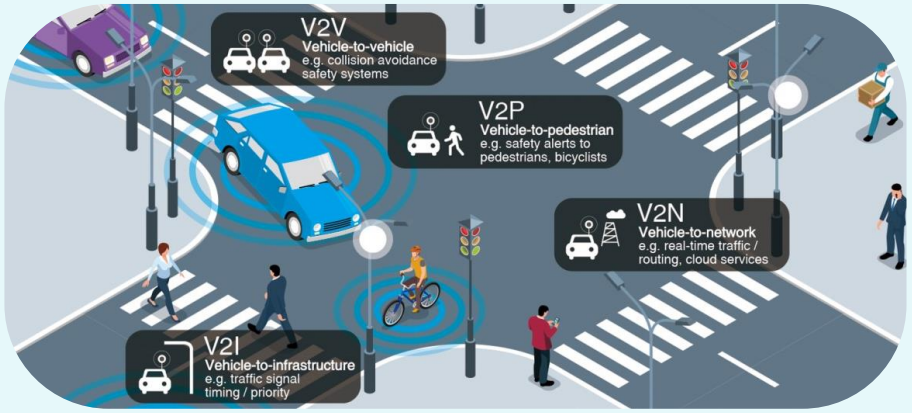
협력주행	1	2	3	4	5	6	7	8	9	10	Total
없음	O	O	O	O	O	O	O	O	O	O	100%
보행자	O	O	O	X	O	O	O	O	O	O	90%
공사	O	O	O	O	O	X	O	O	O	O	90%

03

결론 및 제언

결론 및 제언

- **라인트레이싱 개선**
  - ✓ PD 제어기를 통해, 안정적인 연속주행으로 개선 완료.
- **자율협력주행 구현**
  - ✓ CCTV의 번호 및 상황 인식 정보를 JetRacer에 전달하여, 교차로 사각지대에서의 안전한 주행을 구현.
  - ✓ 서버에 웹소켓 프로토콜과 비동기 프로그래밍을 적용해, 두 클라이언트간의 안정적이고 빠른 데이터 교환 구현.
  - ✓ 추후 더 많은 CCTV와 JetRacer를 이용하여, 더욱 복합적인 V2X 모빌리티 시스템을 개발할 수 있을 것임.



감사합니다