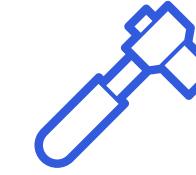
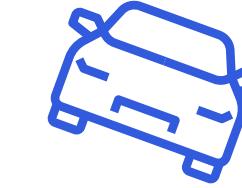




* 자율주행프로그래밍



장희빈

충돌 후 복귀 시스템

TinyLidarNet & PP 를 활용한 충돌 후 raceline 복귀 시스템

박수빈, 장수인, 최희우

20241966, 20243298, 20241992

제안서

목차



- 1 문제 정의
- 2 Detection & State - 박수빈
- 3 Planner - 장수인
- 4 Controller - 최희우
- 5 overview
- 6 Project Schedule
- 7 참고 자료

1. 문제 정의

자율주행 차량의 충돌 후 복귀 기능 부재에 따른 문제 정의

현재 상황

현재 오픈소스 자율주행 시뮬레이터 ForzaETH에서는 차량이 주행 중 충돌이 발생했을 때, 차량이 자동으로 복귀하거나 주행을 재개하는 기능이 구현되어 있지 않다.

문제의 중요성

자율주행 차량은 실제 레이스 환경에서 언제든 예기치 못한 충돌이나 오작동이 발생할 수 있다. 하지만 매번 수동으로 복구해야 한다면 실질적인 자율 주행 시스템이라 보기 어렵다.

해결해야 할 문제 정의

- (1) 자율주행 중 충돌 발생을 정확히 감지할 수 있는 알고리즘을 설계한다.
- (2) 충돌 감지 후 차량의 자세 위치 오차를 분석하여 정상 주행 궤도로 복귀시키는 복귀 알고리즘(recovery system)을 구현한다.
- (3) 복귀 과정에서 경로 계획(planning) 및 제어(control) 모듈과의 연동이 원활히 이루어지도록 한다.

PART 1. DETECTION & STATE

DETECTION : 충돌 감지의 필요성



HTTPS://WWW.ARENAEV.COM/TESLA_AUTOPILOT_TRICKED_BY_PAINTED_WALL_EXPOSING_CAMERAONLY_LIMITS-NEWS-4519.PHP

레이스/주행 중 차량이 정체·추돌·벽 접촉 등으로 구동

명령은 있는데 이동이 없는 경우가 발생

→ 자율 복구 필요

기존 ForzaETH 파이프라인에는 충돌 판별 로직 부재

→ 상위 Recovery 모듈이 시작 신호를 받지 못함

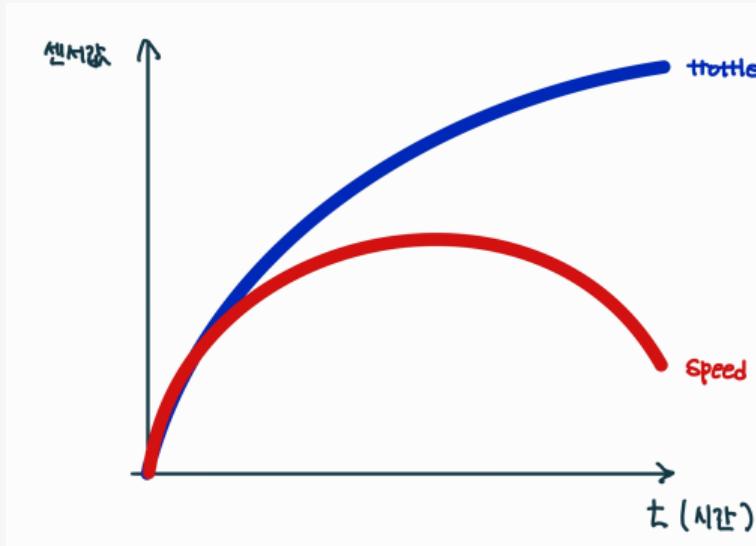
센서 융합 기반 임계치 판단 + 래치로 저지연·저복잡도 충돌
감지를 제공

충돌 감지는 차량의 상태를 스스로 인식해 복구 절차를 시작하게 만드는 시스템의 출발점

DETECTION : 센서 융합 기반 충돌 판단 로직 설계

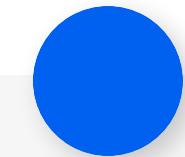


Odom

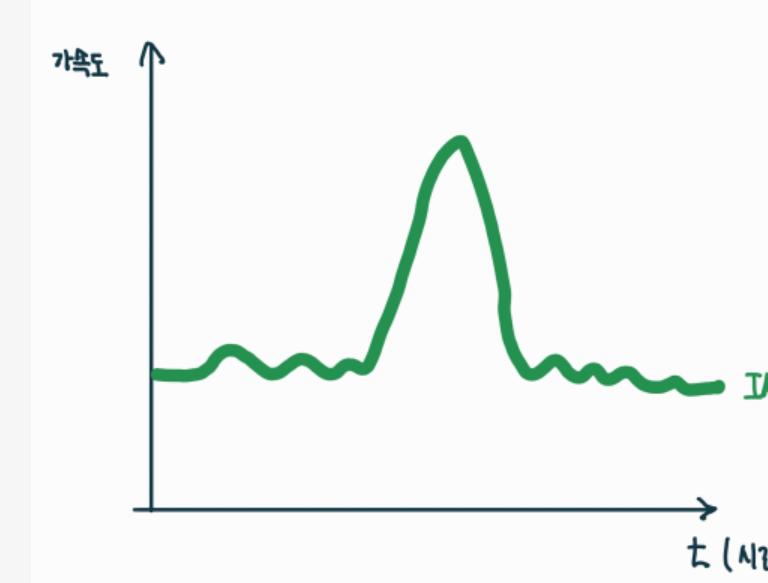


속도 $< 0.05\text{m/s}$ & throttle > 0.2

구동력은 있으나 차량이 정지 상태



IMU

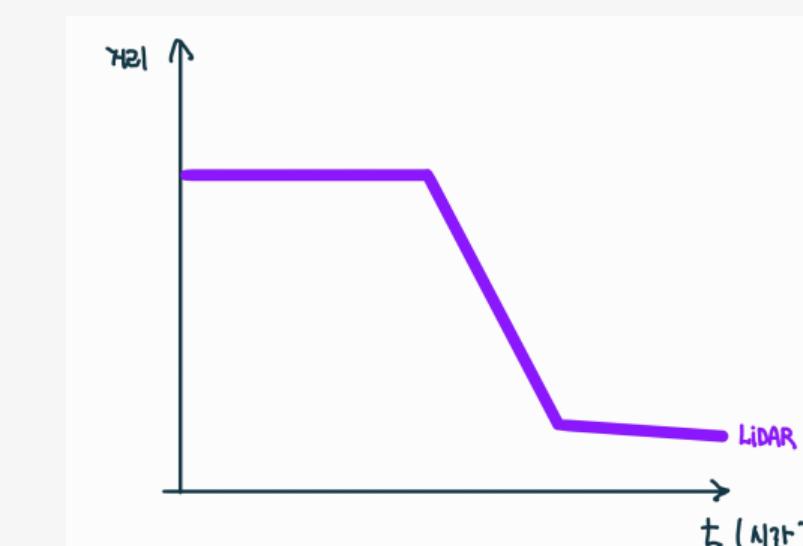


$|a_x| > 4.0 \text{ m/s}^2$

가속도 급상승 (충돌 충격)



LiDAR

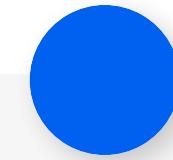


$\Delta \text{range} > 1.0\text{m}$

전방 거리 급감 (장애물 접촉)

1개 이상 센서에서 임계치 초과 시 $\rightarrow /collision_detected = \text{True} \rightarrow \text{latch}$

DETECTION : 센서별 충돌 탐지 동작 원리



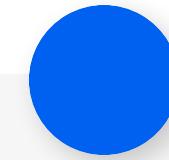
Odom

`/odom → 속도(v) 추출`
`/throttle → 스로틀(τ) 수신`

IF ($v < 0.05$) AND ($\tau > 0.2$):
 odom_flag = True
 ELSE:
 odom_flag = False



차량은 스로틀 신호를 보내고 있으나 실제 속도가 0에 가까우면 구동력이 벽이나 장애물에 막힌 상황으로 판단함.



IMU

`/imu → 선형가속도 a_x 추출`

IF $|a_x| > 4.0$:
 imu_flag = True
 ELSE:
 imu_flag = False



평상시 진동($\pm 1.5 \text{m/s}^2$)과 구분되는 순간적인 충격 피크($\geq 4 \text{m/s}^2$)를 탐지하여 외부 충돌 이벤트를 감지함.



LiDAR

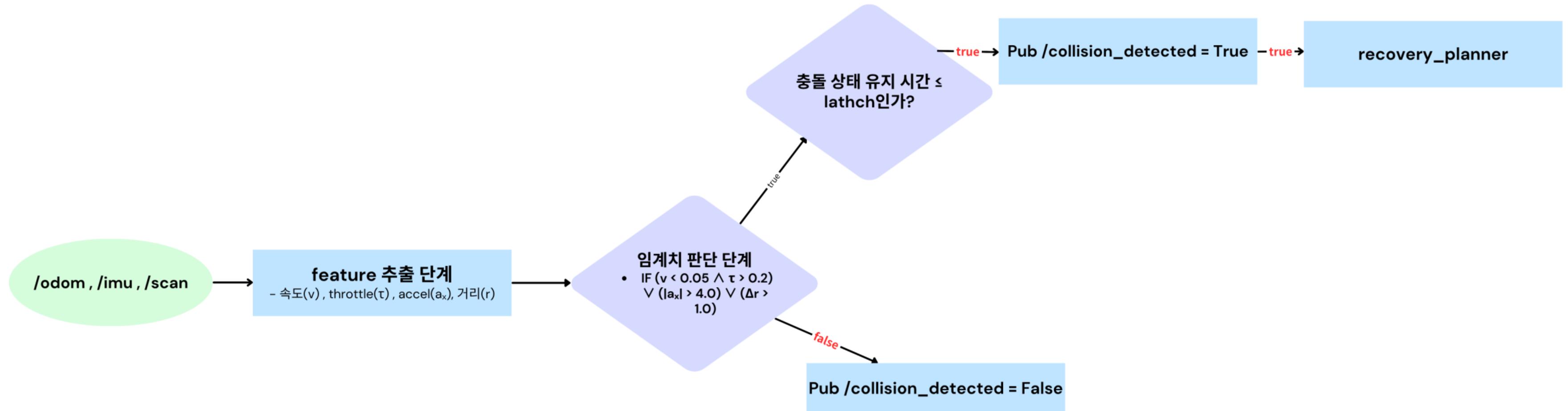
`/scan → 전방거리값 추출`

IF $\Delta r > 1.0$ or $r_{\min} < 0.7$:
 lidar_flag = True
 ELSE:
 lidar_flag = False



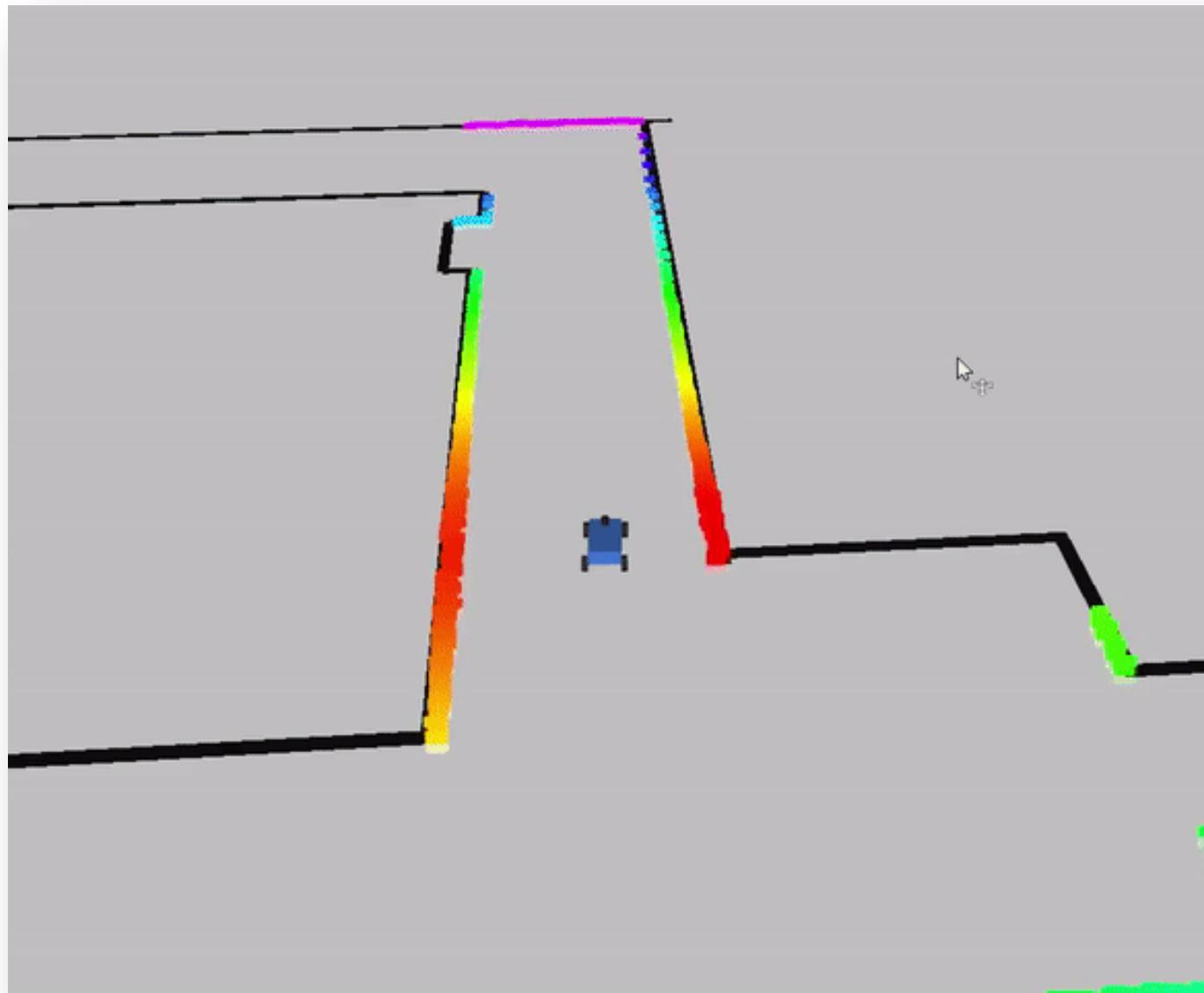
LiDAR는 전방 거리의 급격한 감소나 일정 시간 이상 근접 상태가 유지될 때 물리적 접촉으로 판단함.

DETECTION : 충돌 감지 알고리즘



1. ODOM, IMU, LiDAR 데이터를 통합하여 충돌 여부를 판단
2. 하나라도 임계치를 초과하면 /collision_detected=True 발행
3. latch 후 recovery_planner로 전달

STATE MACHINE



두 가지 주요 상태

- 1. NORMAL RACE MODE
- 2. RECOVERY MODE

상태 전환 트리거

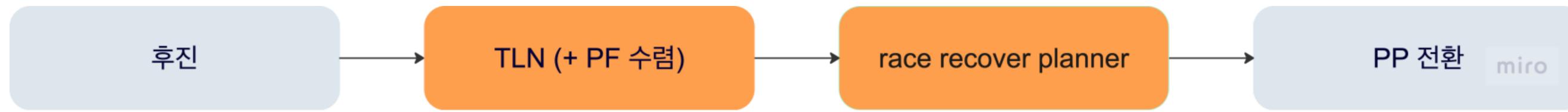
- 충돌 감지 -> RECOVERY MODE 활성화
- 복구 완료 -> NORMAL RACE MODE 복귀

상태에 따라 경로 추종 방식을 변경

https://korea-race23.f1tenth.org/ko/getting_started.html

PART 2. PLANNER

PLANNING: TLN 복귀 및 POSE 안정화 + 기준 경로 복귀 경로 생성



TLN: Lider기반 particle filter를 통해 자차 pose를 안정화(재수렴)

race recover planner: TLN 복귀 주행의 마지막 위치에서 기준경로까지의
부드러운 경로를 생성

PLANNING: TLN 복귀 주행 및 POSE 안정화

TinyLidarNett(TLN): LiDAR scan 데이터 입력 → 차량이 트랙 중앙선을 추종

LiDAR Scan

CNN (TinyLidarNet)

Steering Angle

miro

TLN은 정확한 pose없이 Lidar 패턴 만으로 주행

충돌 후 pose가 불안정한 상황에서 시간 낭비 없이 바로 주행 시작 가능

PF가 안정화될 때까지 차량을 안정적으로 트랙 중앙 근처에 유지하는 역할을 수행

PLANNING: TLN 복귀 주행 및 POSE 안정화

Particle Filter: TLN 구간 동안 LiDAR scan과 트랙 맵을 정합시켜 자차의 Pose를 점진적으로 재수렴

두 지표를 통해 particle filter를 통해 구한 pose의 신뢰도를 확인

particle 분포의 분산 (Covariance) 정도 약 0.05 이하

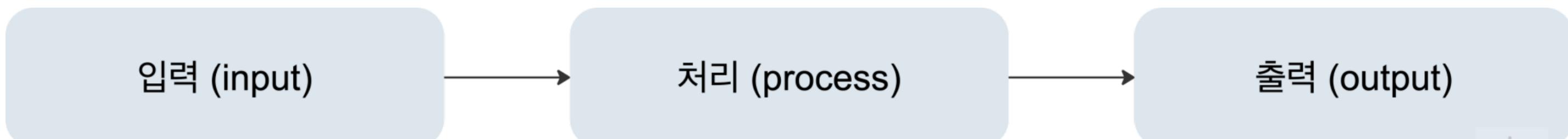
→ pose가 거의 한 점으로 모였는지

PF pose 방향 vs 기준 경로 방향 비교 (Heading Error)가 약 5° 이하

→ pose가 주행 방향과 같은 방향을 바라보고 있는지

PLANNING: RACE RECOVER PLANNER

부드럽게 기준 경로(RACELINE)로 복귀시켜서
PURE PURSUIT로 전환할 수 있게 하는 연결 계획 노드



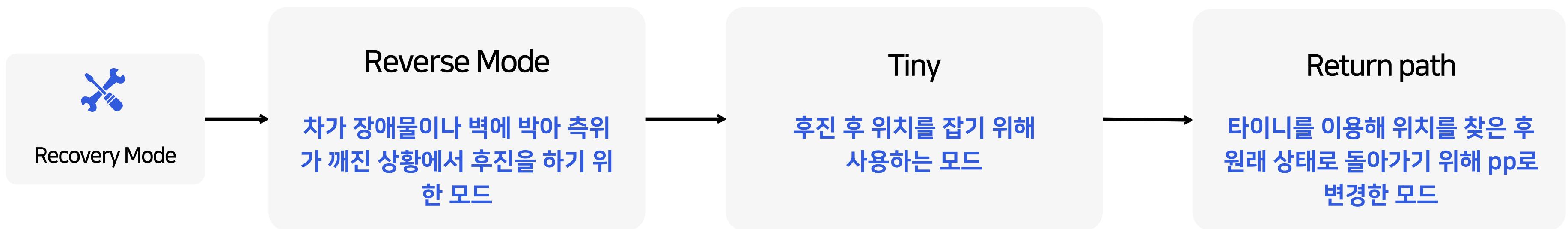
TLN 종료 시점에 수렴된
pose와 기준 경로를 입력으로
넣는다

TLN pose와 Raceline 근접점
→ cubic spline으로
/transition_path 생성

/transition_path를 만들어
Controller(PP 제어기) 쪽으로
보내줌

PART 3. CONTROLLER

세부 모드



후진 구현

01

상태 전환

Reverse Mode에 진입
했는지 확인

02

후진 속도 결정

안전한 음수값으로 설정

03

조향각 결정

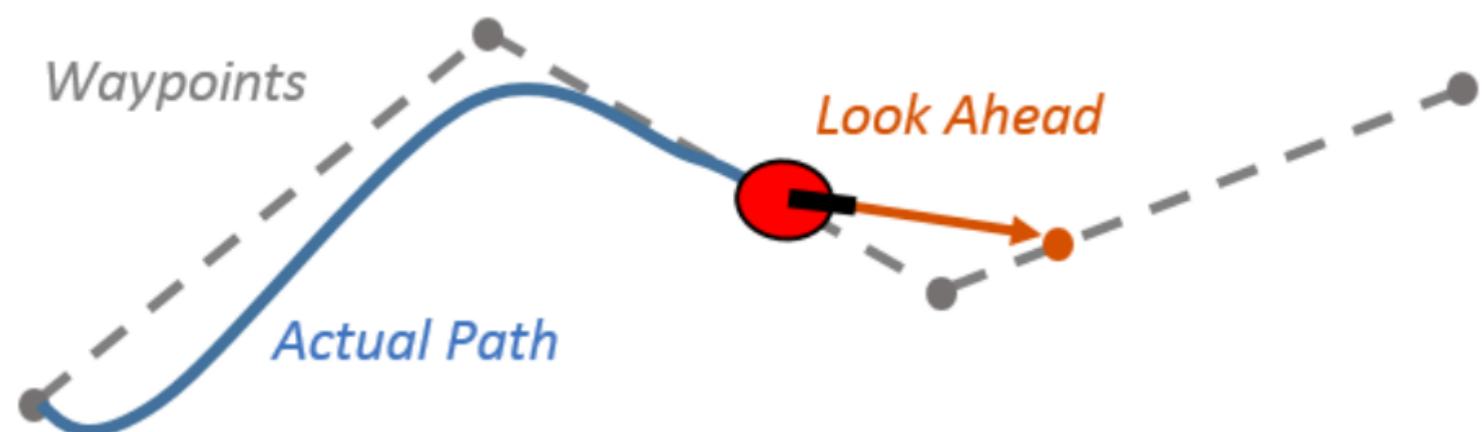
1. 단순 후진
조향각을 0으로
2. 경로 추종
PP를 사용하여 계산

04

후진 완료



복구 경로 추종



<https://kr.mathworks.com/help/nav/ug/pure-pursuit-controller.html>

RECOVERY MODE일 시에 PLANNER가 준 복귀 궤적을
받아 PP로 추종하며 진행

복구 시 안정성과 정확성을 위해 L_D를 낮게, V를 저속으
로 설정

안정화 판단 후 새로운 노드 발행
-> 복구 노드를 비활성화 한 후 FORZAETH의
/LOCAL_WAYPOINTS를 다시 추종하도록 제어권을 전환

PP사용 이유

기하학적 관계

$$l \approx 2\alpha R$$

$$\sin \alpha = \frac{e_{ct}}{l}$$

곡률 표현

$$\kappa = \frac{1}{R} = \frac{2\alpha}{l} \approx \frac{2 \sin \alpha}{l}$$

$$\Rightarrow \kappa = \frac{2e_{ct}}{l^2}$$

조향각 δ 와 곡률 관계

$$\delta = L\kappa = L \frac{2e_{ct}}{l^2}$$

저속 환경에서 안정성
sina 기반 기하학적 관계라 속도에 무관하게 안정적

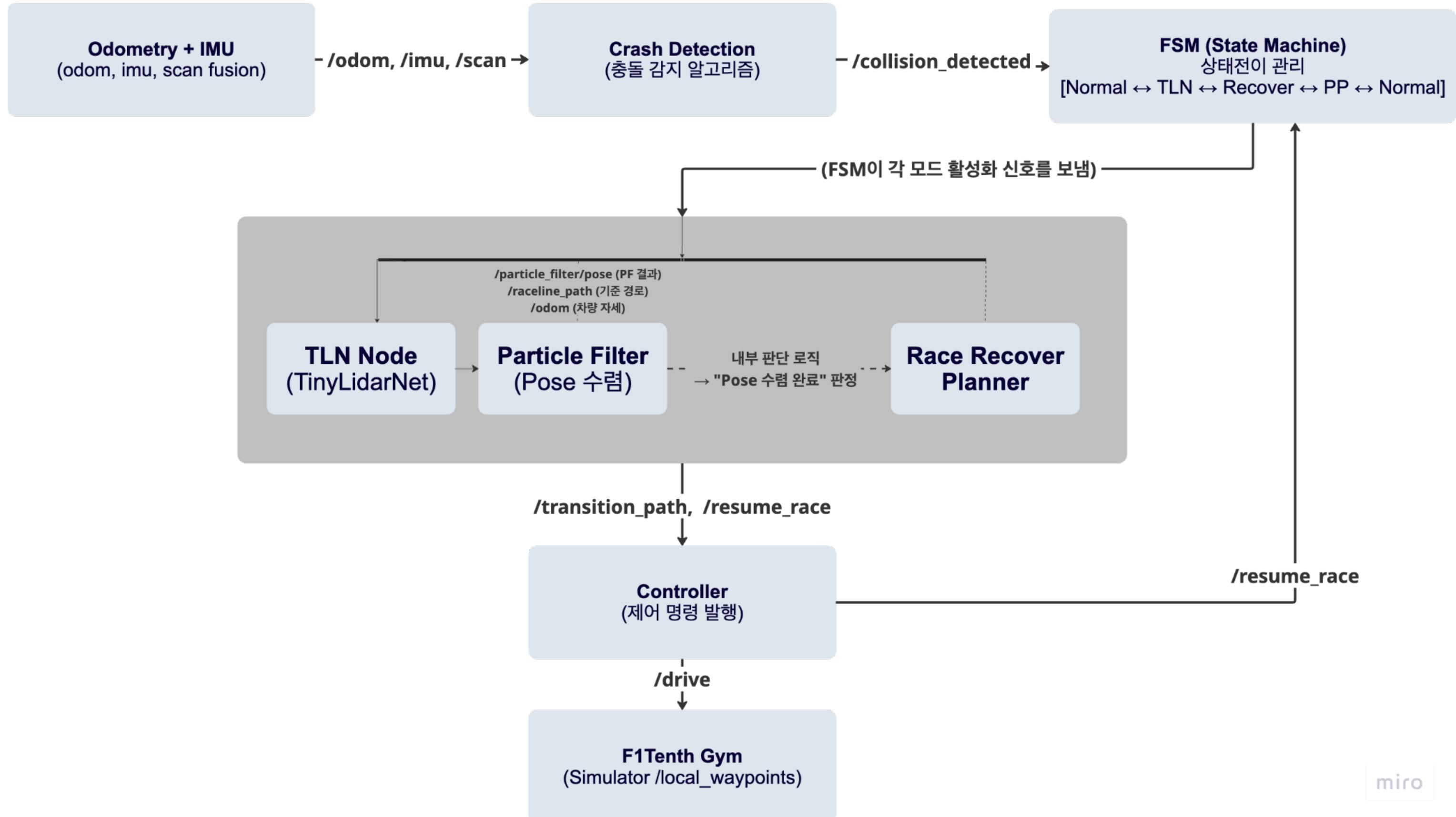
짧고 불규칙한 경로에 대한 반응성

lookahead distance 만으로 조향 반응성을 제어 가능

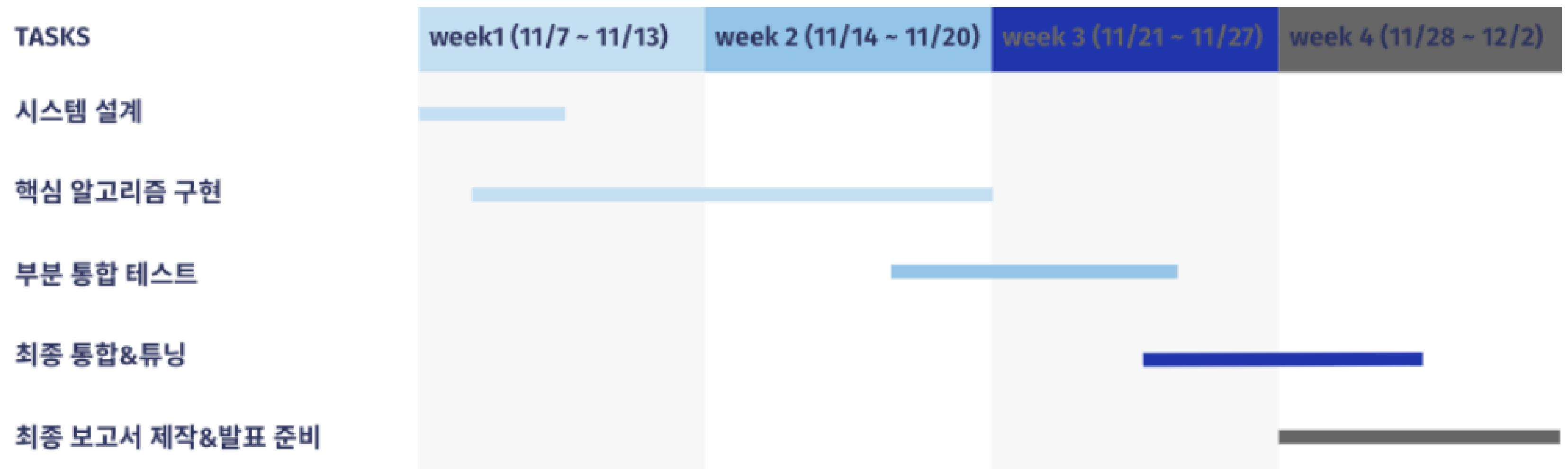
기존 ForzaETH race controller 구조와 호환

구현 단순성 및 통합 용이성

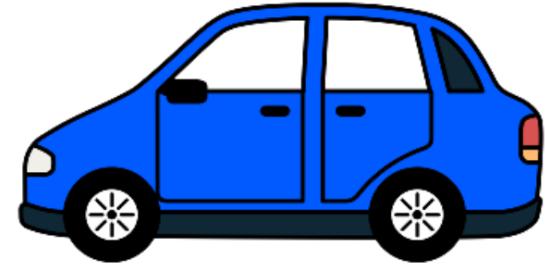
5. OVERVIEW



PROJECT SCHEDULE



**개발하는 패키지는 ROSCPP 패키지입니다.
F1Tenth Gym 시뮬레이터와 연동합니다.
ForzaETH 레이스 스택과 연동합니다.**



감사합니다

Thank you
