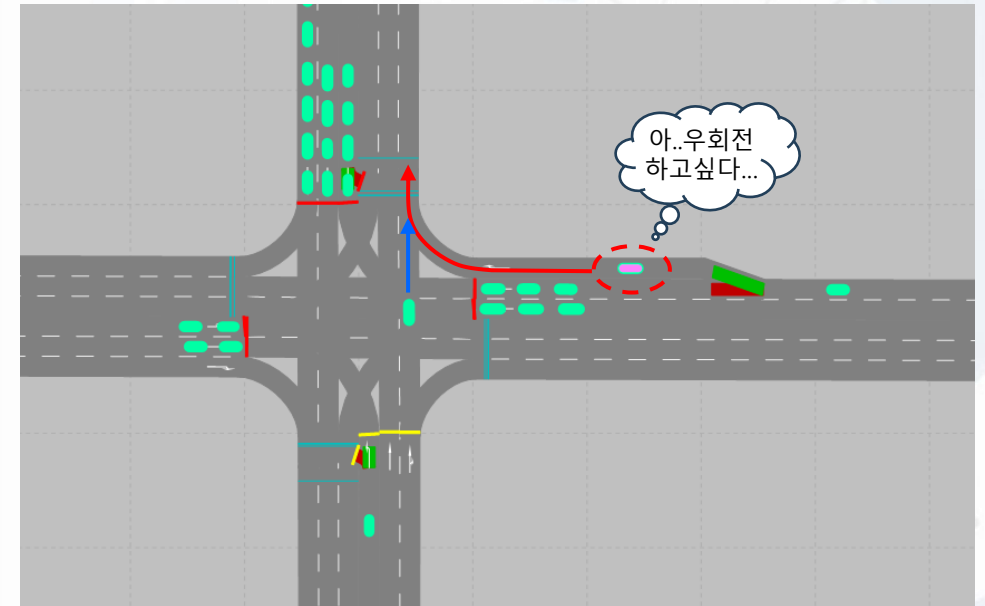


# 신호교차로 자율주행차량의 우회전 상황에서 가감속 패턴 최적화 모델 개발

한양대학교 박사과정    강 가 원

## 1 - 배경 및 학습 목적

- 본 프로젝트는 도시 신호 교차로에서 우회전 차량과 직진 차량 간 상충 위험을 최소화하는 문제를 해결하는 것을 목표로 함
  - 차량의 우회전은 다양한 우선권 규칙(직진차량)과 시야/곡률 제약으로 판단 난이도 높음
- 기존 차량 속도 제어 시 룰베이스 방식/휴리스틱 접근법의 한계: 실시간 적용 어려움
  - 차량이 신호교차로 주행 시 인프라(RSU, Road Side Unit)가 신호현시, 교통 상호작용 정보를 실시간 제공하는 환경을 가정하여 실시간 속도 최적화
- VISSIM COM Interface+Pytorch 분석 환경에서 자율주행차의 주행 정책 최적화 수행
  - 한계: VISSIM에서 시뮬레이션 중간에 lane change 행태 실시간 변경 구현은 어려움
- 따라서, 차로변경이 필요하지 않은 우회전 접근-우회전-우회전 후 구간의  
종방향 제어 정책(가속도, 감속도 조정)을 중점으로 학습
- 최종적으로 차량의 안전하고, 효율적인 우회전 주행을 목적으로 강화학습을 설계함

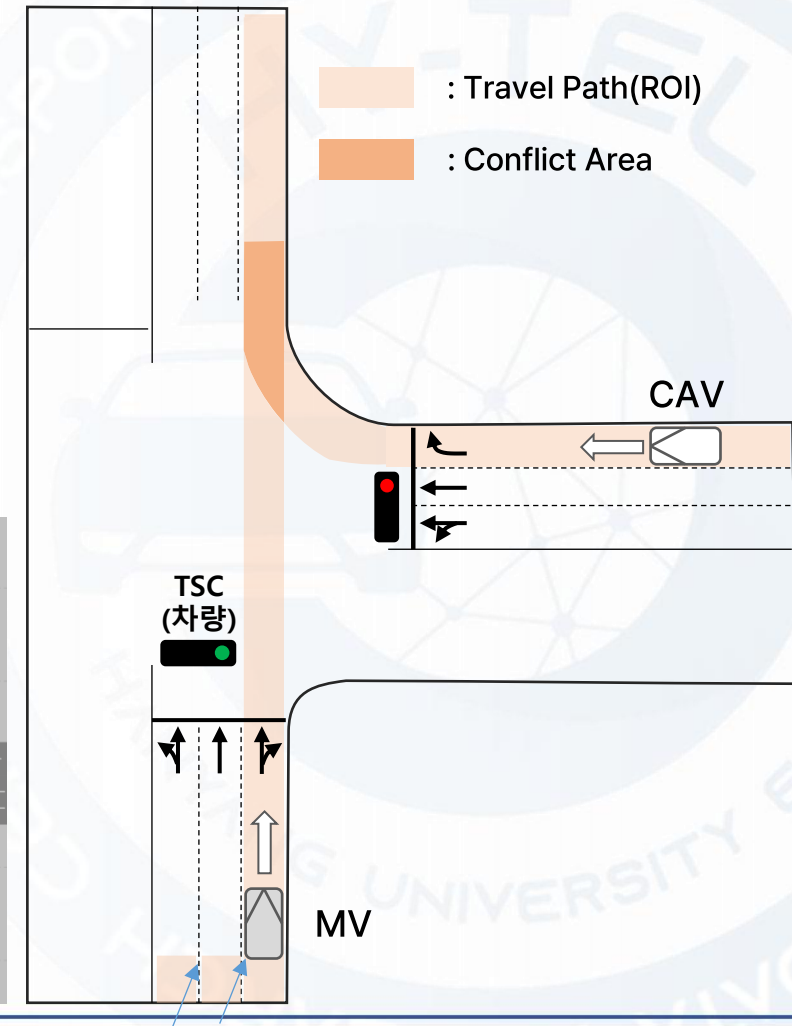
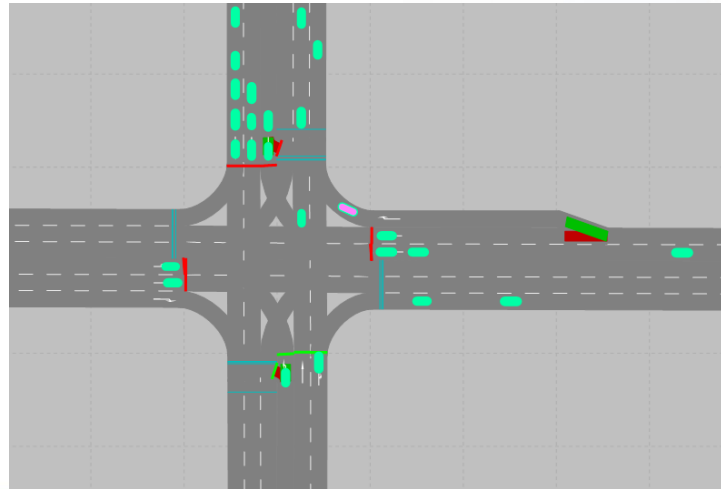


## 2. 연구 내용

### 1 - 분석 네트워크

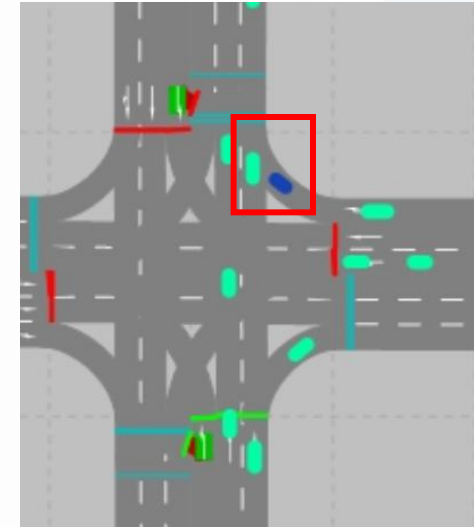
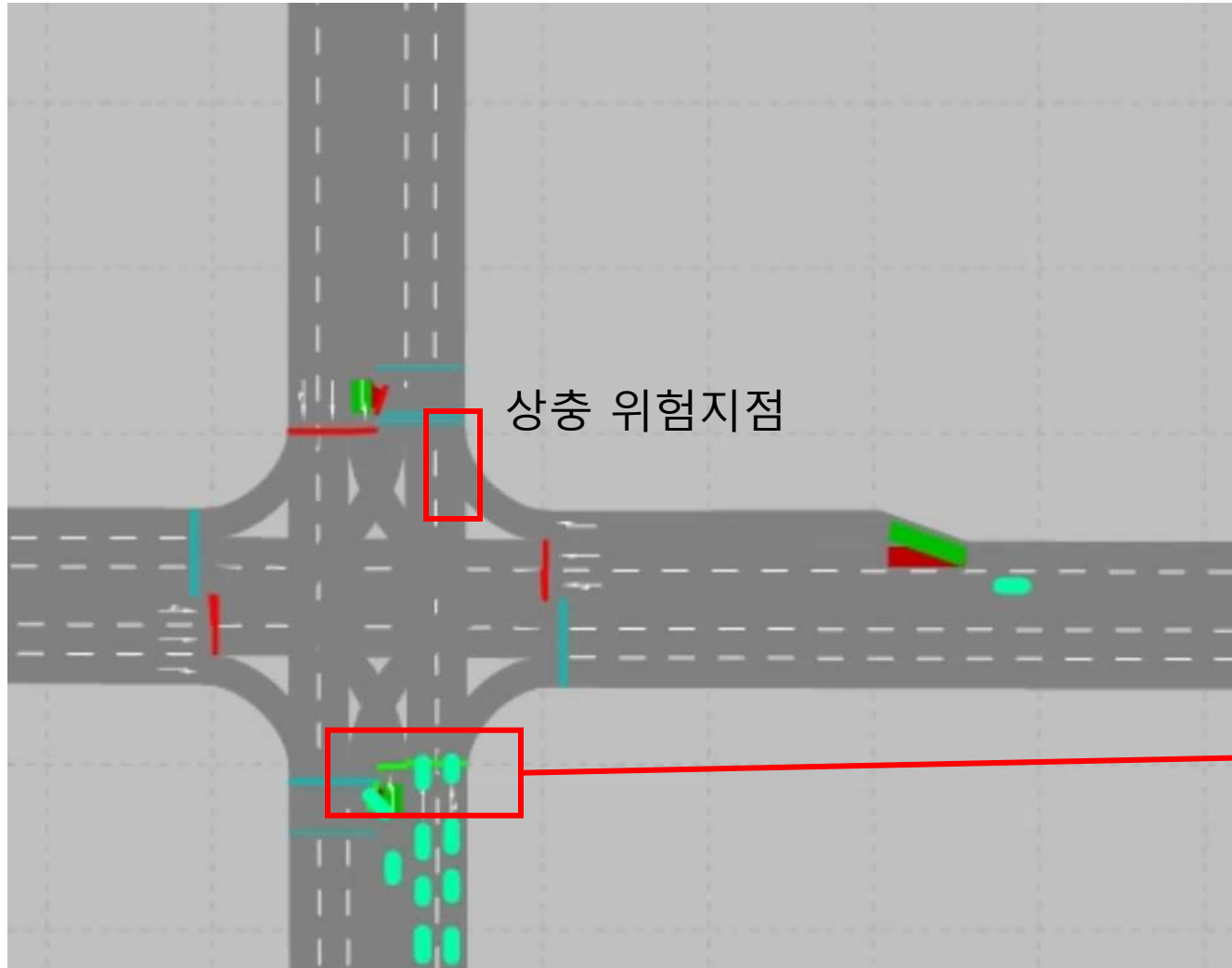
- 4지 신호교차로: 교차로 동측(←)에서 진입하는 상황  
적신호시 우회전(비보호 우회전) 가능 신호 교차로 - 전방 차량 신호등의 신호에 상관없이 우회전 가능  
(ex 동측 신호(RED) - 남측 신호(GREEN) 상황에서도 우회전 가능)
- 제어차량(ego-Veh) 교차로 동측 진입: 최우측 차로 주행, 우회전 테이퍼 존재(우회전 전용 차로)
- 대향차량(other-Veh) 교차로 남측 진입: 최우측 차로 주행, 데이터 수집 구간에 따라 타 차선 진입 고려 필요
- 주행 속도 제어 케이스
  - Case1: 동측 ego-Veh 전후 차량과 소통하며 우회전 테이퍼 진입을 위한 가감속 수행
  - Case2: ego-Veh이 우회전 수행 중 대향차량(남측, other-Veh)들의 회피를 위한 가감속 수행

No	Signal group	Signal sequence	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190						
1	동직	Red-G	0.0							77.0													0	77				3
2	동좌	Red-G	0.0							97.0													0	97				3
3	서직	Red-G	0.0							77.0													0	77				3
4	서좌	Red-G	0.0							97.0													0	97				3
5	남직	Red-G											100.0										100	147				3
6	남좌	Red-G											100.0										100	147				3
7	북직	Red-G																150.0					150	197				3
8	북좌	Red-G																150.0					150	197				3
9	북유	Red-G																150.0					150	197				3
10	남유	Red-G											100.0										100	147				3

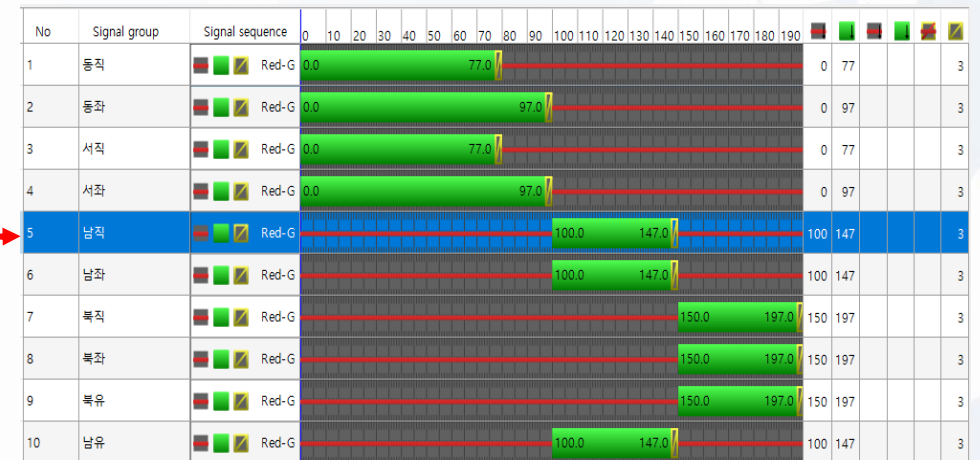


## 2. 연구 내용

### 1 - 분석 네트워크

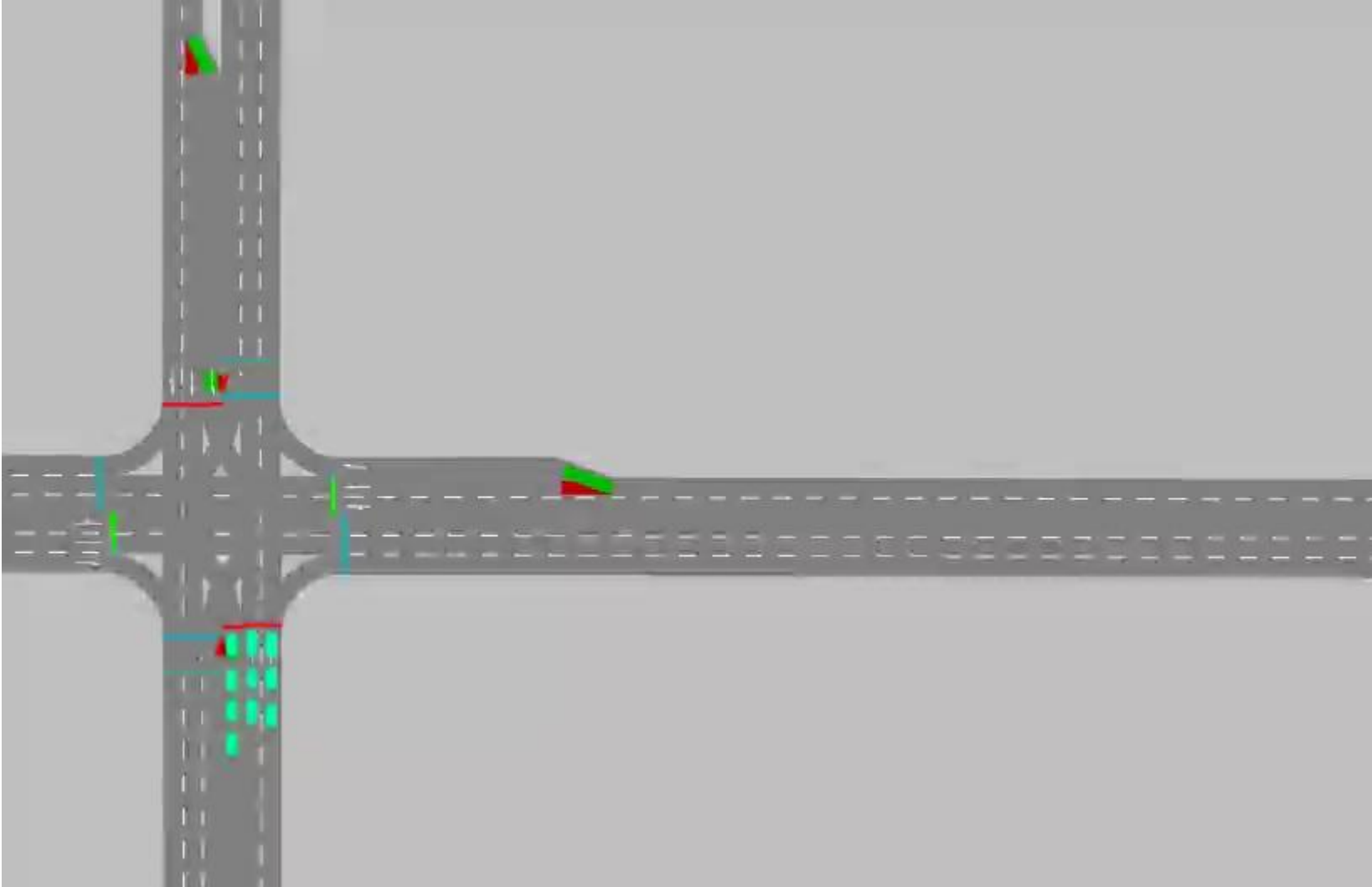


신호현시(Cycle)



## 2. 연구 내용

### 1 - 분석 네트워크



## 2. 연구 내용

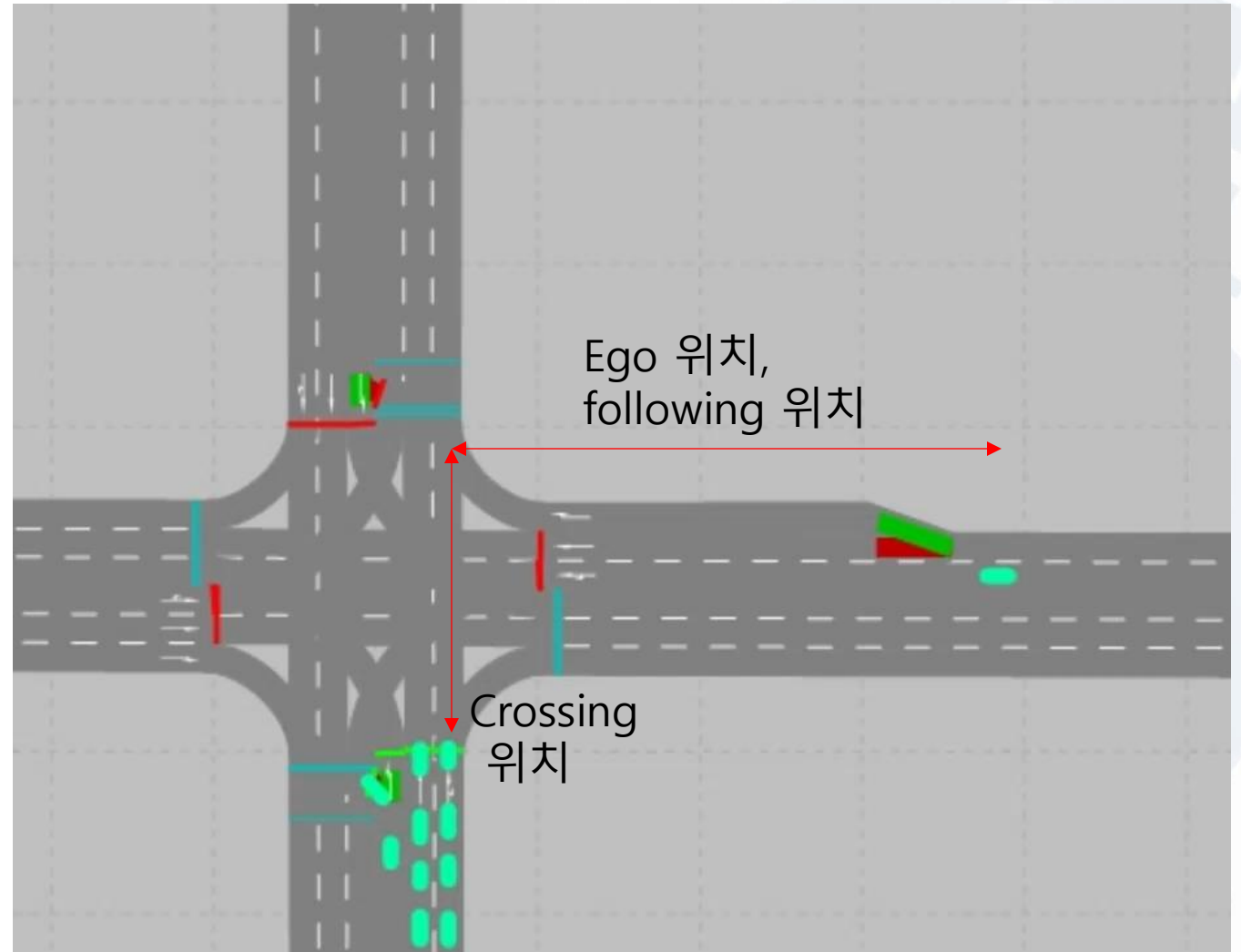
### 2 - 문제 정의

#### 1) Timestep

- 0.1s, 1s

#### 2) State space

- $v_1 \sim v_5$  시도: 8~13차원 continuous vector
- 구성요소(raw/normalize 가공해서 비교)
  - ✓ Ego 차량 속도:  $v_E$
  - ✓ Ego 차량 위치-conflict point까지 거리:  $d_E$
  - ✓ Ego 예상 도달시간:  $t_E$
  - ✓ 대향차 선두 속도, 거리:  $v_w, d_w$
  - ✓ 예상 PET surrogate:  $PET_t = t_O - t_E$
  - ✓ 남측 대향차로 직진 신호(Green time) 시작까지 잔여시간





### 4 - State 설계 2

#### 차량 관련 요소

- 전방/합류/교차류 차량 대상 현재 상태 지표
  - 현재 도로 link ID, 교차로 중심 기반 좌표(x,y), 속도
  - Ego veh와의 최소 상대거리, 상대속도(같은 방향 주행 차량간 연산)
  - 상호작용하는 차량간 상충 위험도 - 대리안전지표(Surrogate Safety measure): TTC(Time-to-Collision), PET(Post Encroachment Time) 등

#### ① 좌표기반 TTC

- TTC 임계값: 1.5s, 2.0s, 2.5s, 3.0s

$$TTC = \frac{\sqrt{(X_{sub} - X_{opp})^2 + (Y_{sub} - Y_{opp})^2}}{|V_{sub} - V_{opp}|}$$

현재 상대속도와 간격이 그대로 유지된다고 가정할 때  
충돌까지 남은 시간(= 거리 ÷ 접근 상대속도)

#### ② PET

- PET 임계값: 1s, 2s, 3s

$$PET = \left| \frac{\sqrt{(X_{sub} - \alpha)^2 + (Y_{sub} - \beta)^2}}{|V_{sub}|} - \frac{\sqrt{(X_{opp} - \alpha)^2 + (Y_{opp} - \beta)^2}}{|V_{opp}|} \right|$$

동일 충돌지점에서 선행 주체가 벗어난 순간과 후행 주체가  
도달한 순간 사이의 시간 간격

Where,

*sub* : The vehicle ID of subject vehicle  
*opp* : The vehicle ID of opponent vehicle  
*X* : The vehicles GPS X positions  
*Y* : The vehicles GPS Y positions  
*V* : Speed components  
 $\alpha$  : Conflict point of X positions  
 $\beta$  : Conflict point of Y positions

## 2. 연구 내용

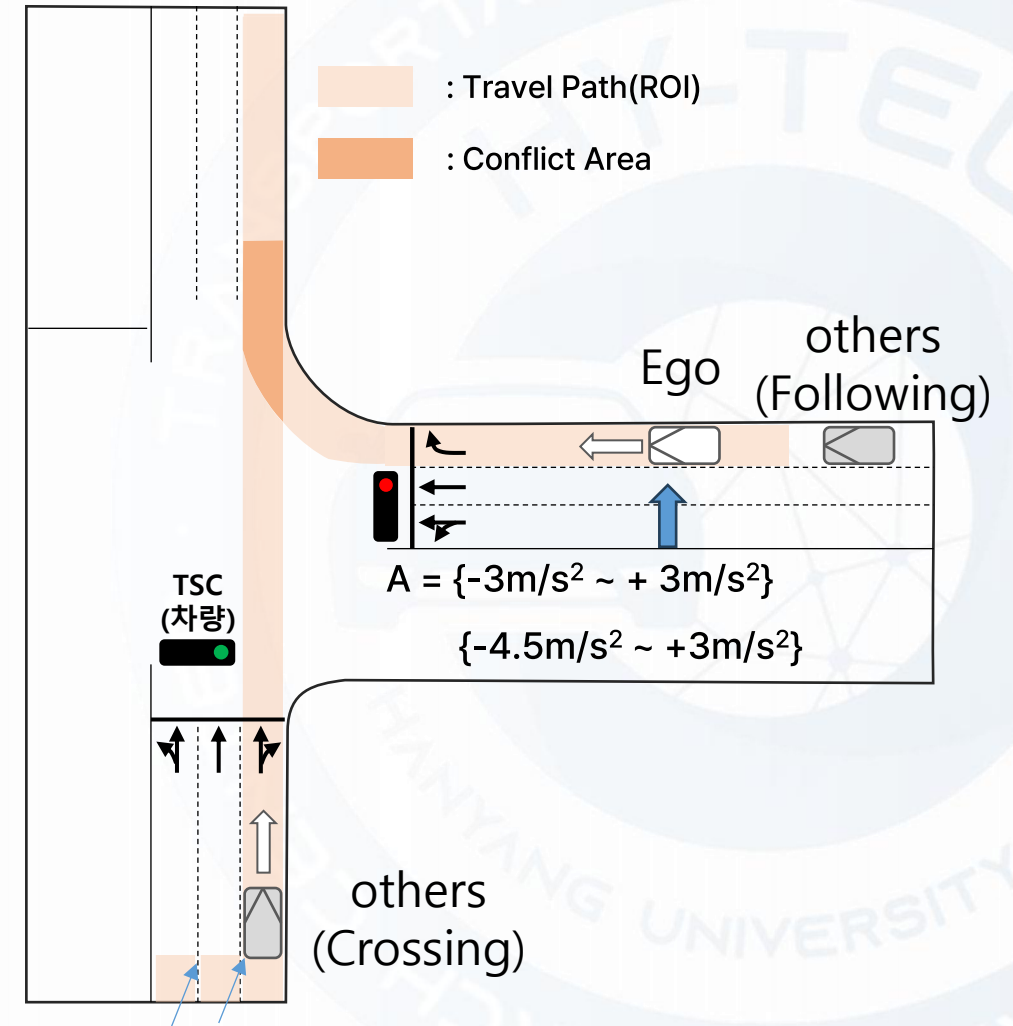
### 2 - 문제 정의

#### 3) Action space(행동공간)

- Continuous한 가감속 Action 수행
- $\{-3\text{m/s}^2 \sim +3\text{m/s}^2\}$
- $\{-4.5\text{m/s}^2 \sim +3\text{m/s}^2\}$  <sup>1)</sup>

#### 4) Terminal Condition(Episode 종료 조건)

- 우회전 차량(Ego)이 conflict zone 통과 시(7-1 도로구간 도달 시)
- 아래 조건 발생 시 즉시 종료 및 큰 패널티(-reward 함수에서 다시 설명)
  - $\text{spacing} < 2\text{m} \rightarrow$  Ego와 others 차량간 거리가 2m 미만일 시(충돌 발생)
  - 비정상 속도( $0 \sim 75\text{km/h}$ 에서 벗어난 속도)





## 5) Reward 함수

- 
- Diagram illustrating a vehicle merging into traffic from a side road, showing the Travel Path (ROI) and Conflict Area.
- Legend:
- Travel Path(ROI)
  - Conflict Area
- Key elements and labels:
- TSC (차량)
  - Ego
  - others (Following)
  - others (Crossing)
  - Acceleration ranges:  $A = \{-3\text{m/s}^2 \sim +3\text{m/s}^2\}$  and  $\{-4.5\text{m/s}^2 \sim +3\text{m/s}^2\}$

## 2. 연구 내용

### 3 - 학습 알고리즘

- 다양한 학습 모형간 비교 – 정책 수렴성, 평균 누적 reward, 안전·효율성 지표 평균값 결과 비교
  - (Base) 제어 없음. VISSIM 자체 운전자 모델 기반 평균 충돌 위험도 산출
  - ~~— (Old) Heuristic 기반의 가감속 제어 (Rule based — PET 임계값 등 규칙 임계값에 따라 감속/가속/유지 액션을 결정)~~
  - 최신기법 (DDPG) 연속 행동을 직접 출력하는 결정적 Actor-Critic(오프폴리시)으로, 리플레이 버퍼와 타깃 네트워크로 학습
  - 최신기법(TD3) DDPG를 개선한 Twin Critic+정책 스무딩+지연된 정책 업데이트로 과추정과 분산을 줄인 안정형 Actor-Critic
  - ~~— (Extra) DQN(이산 가속도): 가속도를 몇 개의 이산 액션으로 양자화하여 Q함수로 선택하여 값기반 학습~~
- 학습 프레임워크: PyTorch 기반으로 Actor-Critic(TD3/DDPG) 구현

### 3 - 학습 알고리즘

#### DDPG의 주요 문제점

DDPG는 Actor-Critic 기반 deterministic policy gradient 알고리즘으로, 아래 한계가 존재함

- Q-value overestimation

단일 Critic(Q-network)을 사용하여 target Q 계산  $\rightarrow$  noise-function approx. error가 Q값을 과대추정(overestimation bias)

- 정책 업데이트 불안정성

Actor가 over-estimated Q를 따라가면서 발산 혹은 sub-optimal 학습

- action noise 처리 부족

exploration은 Gaussian noise로 처리하지만, target action perturbation 안정화 부족

- 학습 진동(oscillation)

Actor와 Critic이 동시에 빠르게 업데이트되어 policy가 자주 흔들리는 문제 발생

### 3 - 학습 알고리즘

#### TD3의 개선 요소

##### 1) Clipped Double Q-Learning

두 개의 독립된 Critic 네트워크  $Q1(s,a)$ ,  $Q2(s,a)$  학습

target Q 계산 시  $\min(Q1', Q2')$  사용

⇒ Q-value overestimation 제거

⇒ 안정적 value function 학습

⇒ deterministic policy의 과대강화(over-optimistic policy) 방지

##### 2) Delayed Policy Updates

Critic는 매 step 업데이트, Actor는 N step (예: 2)마다 업데이트

Critic이 안정된 Q-function을 형성한 뒤 Actor를 업데이트

⇒ policy oscillation 감소

⇒ gradient noise 감소

⇒ 학습 안정성 증가

##### 3) Target Policy Smoothing

target action 계산 시 작은 Gaussian noise( $\epsilon$ )를 의도적으로 더함

$$\tilde{a} = \pi_{\theta'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$$

Clipped Gaussian noise ( $\epsilon$ ) 추가 → small perturbation smoothing

⇒ deterministic policy의 overfitting 방지

⇒ sharp Q-function 근처의 불안정한 gradient 완화

⇒ target value variance 감소

#### 분석 환경

Pytorch 2.7.1

Cuda 11.8

Cuda sdk cudnn 8.9.7.29 cuda12

NVIDIA GeForce RTX 5080

python 3.13.0

### 3. 적용 결과

#### TD3 모델1

Timestep : 0.1sec(Observe, action, reward 계산 수행)

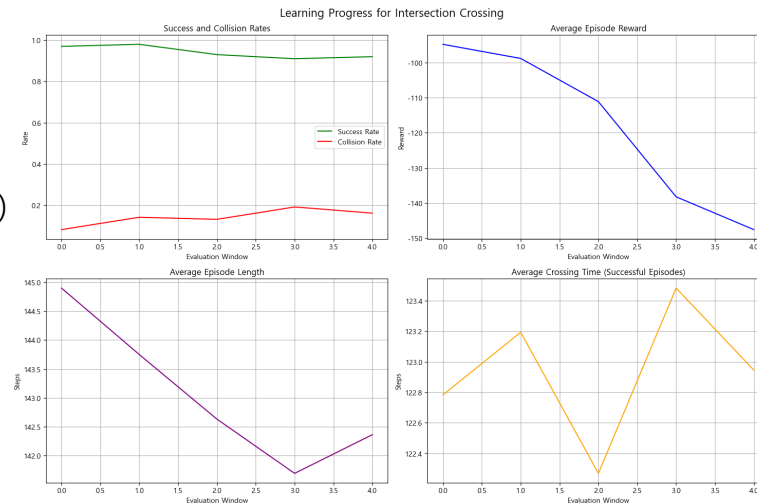
Action =  $\{-3\text{m/s}^2 \sim 3\text{m/s}^2\}$

State space=

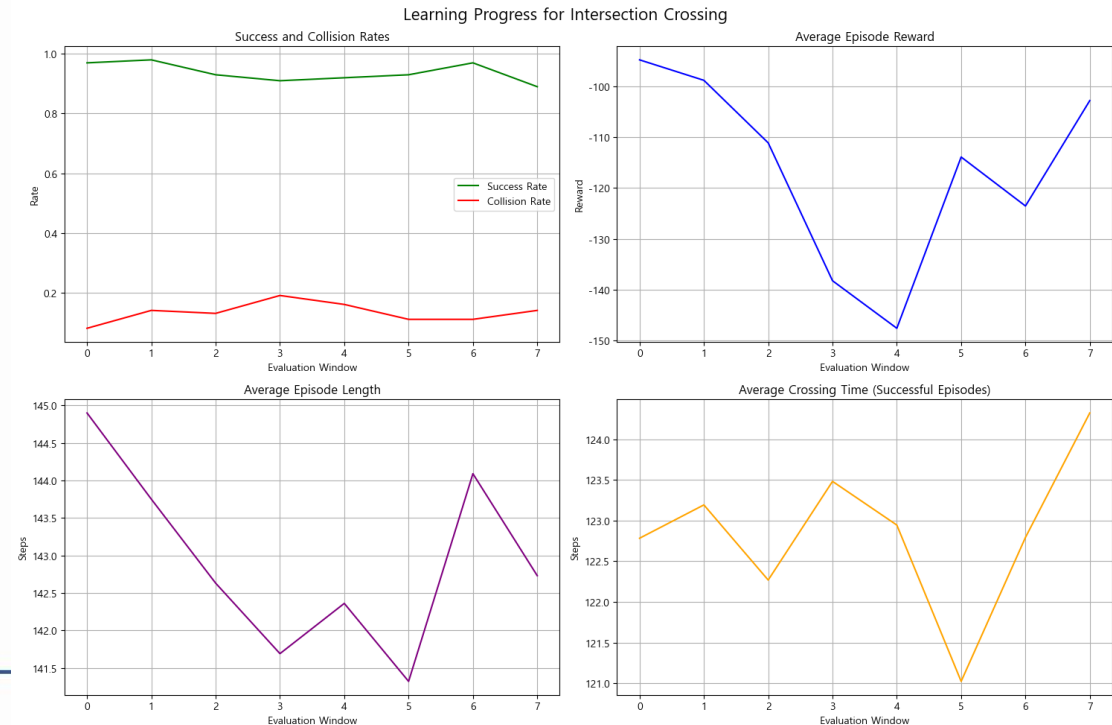
```
state = np.array([
    ego_speed,
    ego_pos,
    time_to_Gn,      # 대향차로 신호현시(Green)까지 남은 시간
    min_spacing,
    min_ttc,
    min_pet,
    rel_speed,
    risk_type
], dtype=np.float32)
```

Reward func =  
Terminate reward  
(충돌, 속도 이상, ttc, pet),  
Step reward  
(올바른 루트 +0.1, 적정 속도 범위 +0.1)

#### 모델1, 500Episodes 수행(150분)



#### 모델1, 800Episodes 수행(1205분)



### 3. 적용 결과

#### TD3 모델2

Timestep : 0.1sec(Observe, action, reward 계산 수행)

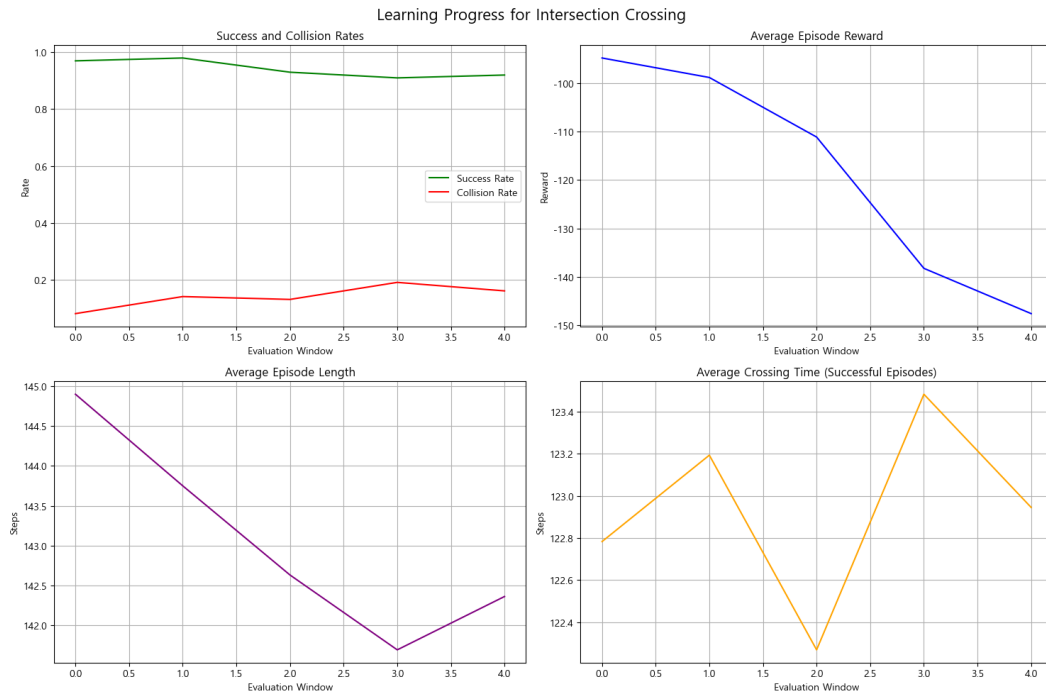
Action =  $\{-3\text{m/s}^2 \sim 3\text{m/s}^2\}$

Reward func = Terminate reward(충돌, 속도 이상, ttc, pet 미만),  
Step reward(올바른 루트 +0.1, 적정 속도 범위 +0.1)

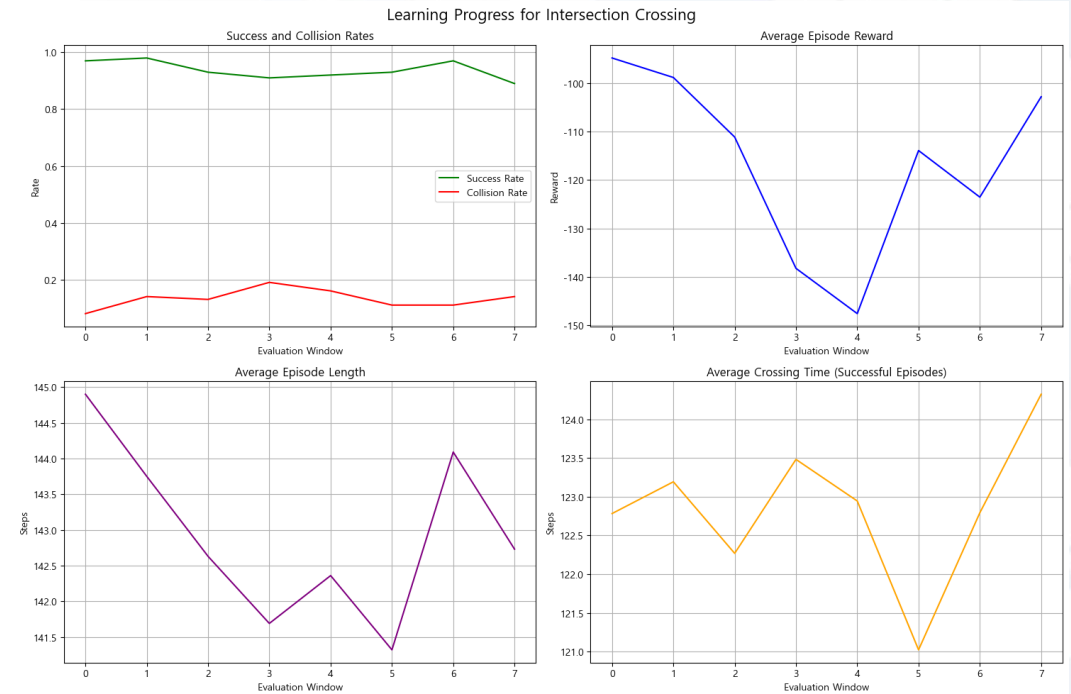
State space=

```
state = np.array([
    ego_speed,
    ego_pos,
    time_to_Gn,      # 대향차로 신호현시(Green)까지 남은 시간
    min_spacing,
    min_ttc,
    min_pet,
    rel_speed,
    risk_type
], dtype=np.float32)
```

500Episodes 수행(150분)



800Episodes 수행(1205분)



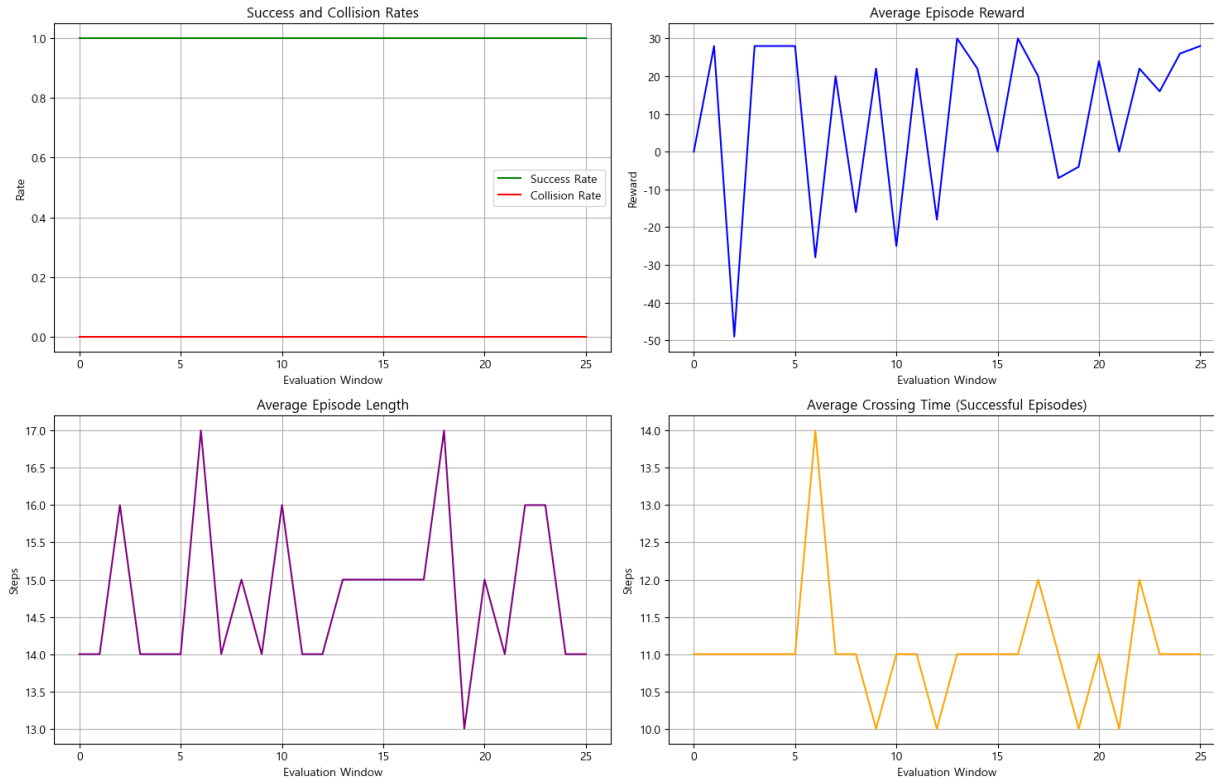


### 3. 적용 결과

#### TD3 모델5

2600Episodes 수행(150분)

Learning Progress for Intersection Crossing



Timestep : **1sec**(Observe, action, reward 계산 수행)  
State space=

```
state = np.array([
    ego_speed,
    dtc_norm,
    time_to_Gn_norm,
    f1['spacing_norm'], # 가장 가까운 following
    f1['ttc_norm'],
    f1['rel_speed_norm'], # 가장 가까운 following 상대속도
    f2['spacing_norm'], # 두 번째 following
    f2['ttc_norm'],
    c1['spacing_norm'], # 가장 가까운 crossing
    c1['pet_norm'],
    c1['rel_speed_norm'], # 가장 가까운 crossing 상대속도
    c2['spacing_norm'], # 두 번째 crossing
    c2['pet_norm']
], dtype=np.float32)
```

Action =  $\{-3\text{m/s}^2 \sim 3\text{m/s}^2\}$

Reward func = Terminate reward(충돌, 속도 이상, ttc, pet 미만),  
Step reward(올바른 루트 +0.1, 적정 속도 범위 +0.1)

결과 그래프 잘못 구성...

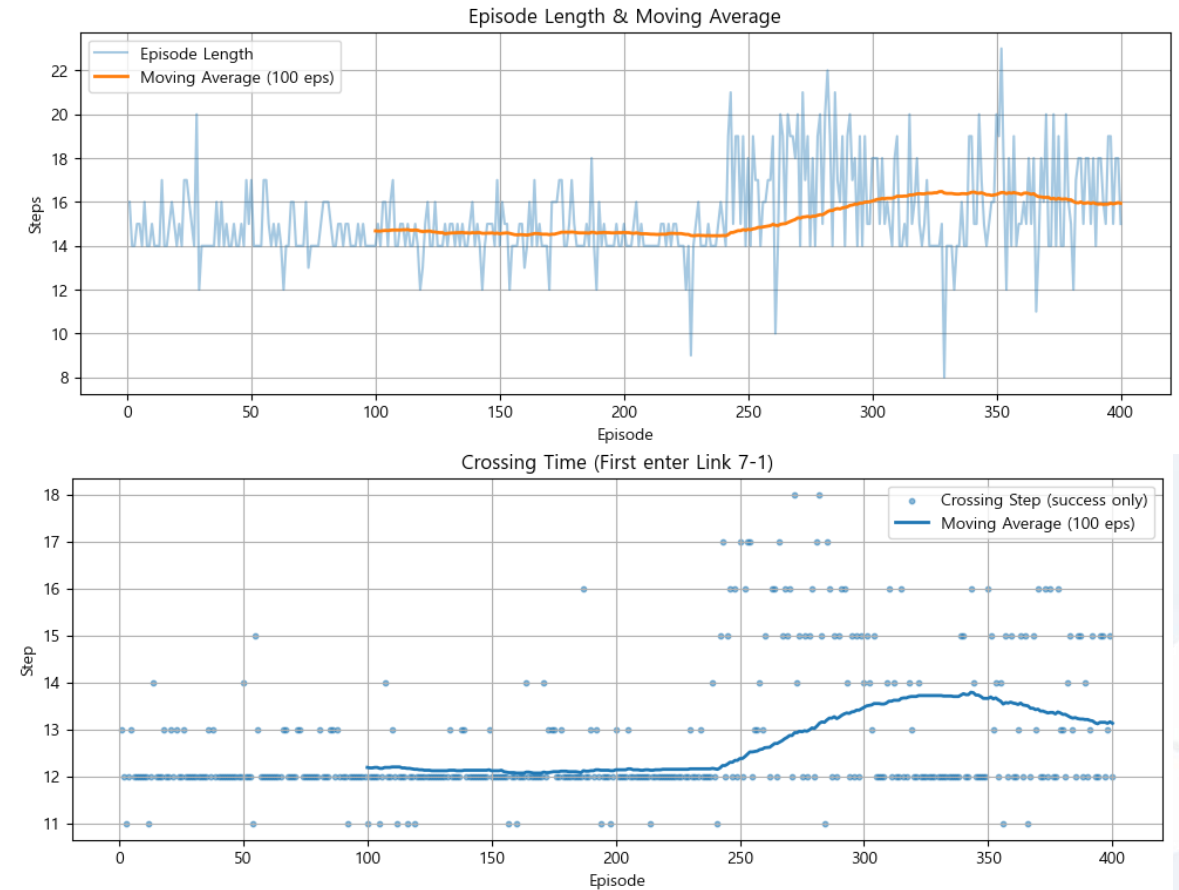
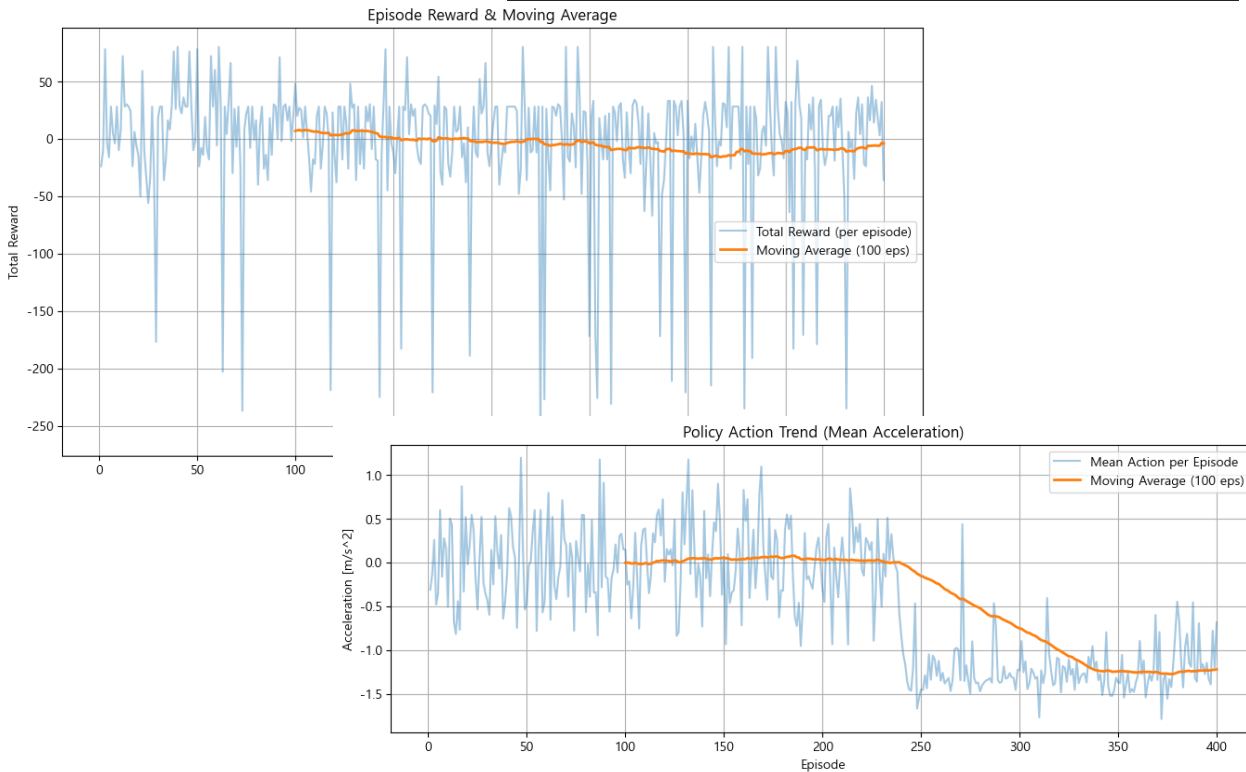
# 3. 적용 결과

## TD3 모델6

```
state = np.array([
    ego_speed,
    dtc_norm,
    time_to_Gn_norm,
    f1['spacing_norm'], # 가장 가까운 following
    f1['ttc_norm'],
    f1['rel_speed_norm'], # 가장 가까운 following 상대속도
    f2['spacing_norm'], # 두 번째 following
    f2['ttc_norm'],
    c1['spacing_norm'], # 가장 가까운 crossing
    c1['pet_norm'],
    c1['rel_speed_norm'], # 가장 가까운 crossing 상대속도
    c2['spacing_norm'], # 두 번째 crossing
    c2['pet_norm']
], dtype=np.float32)
```

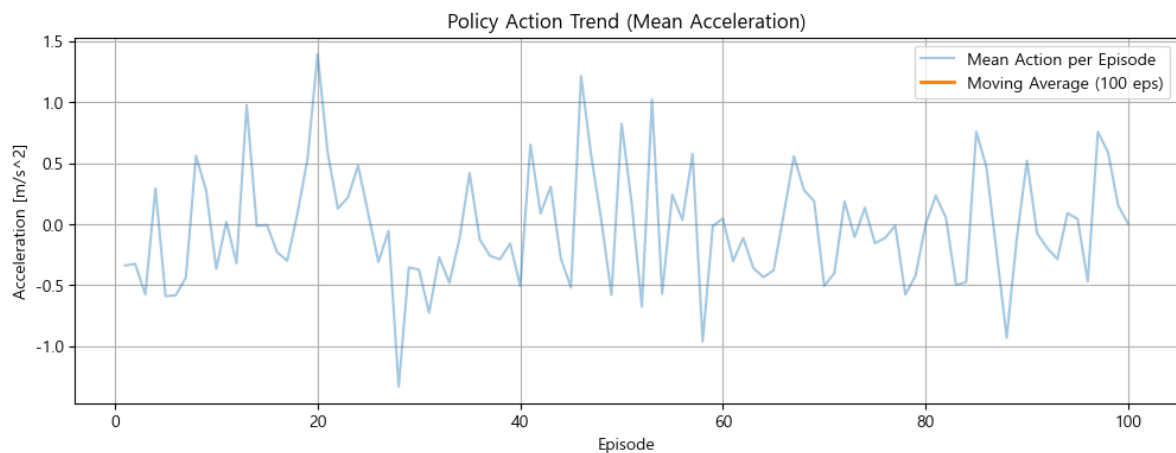
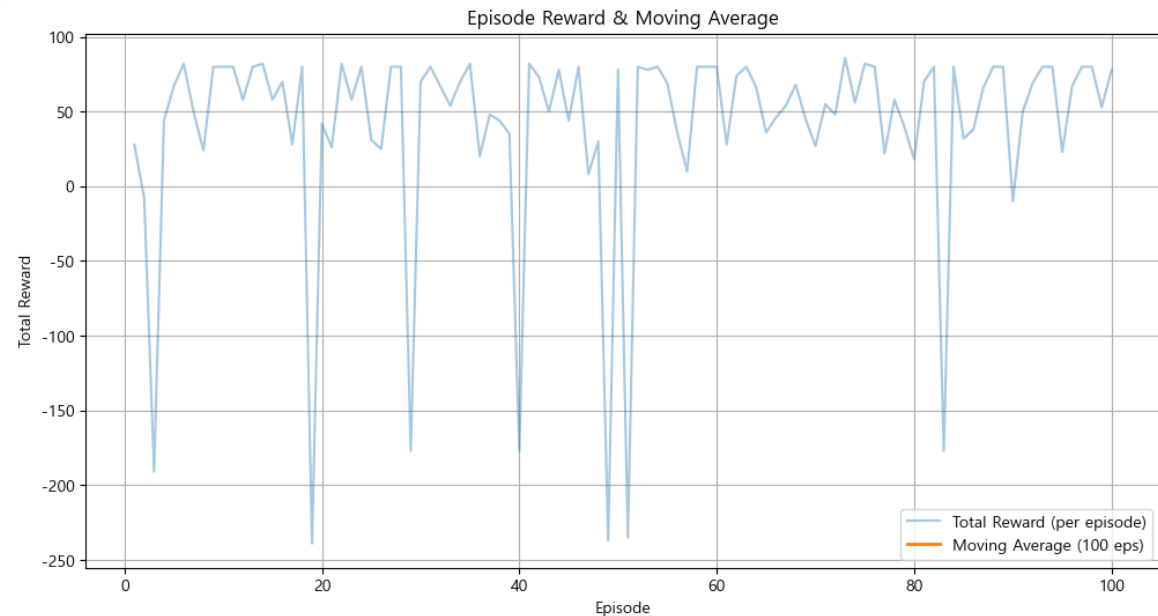
400Episodes 수행(150분)

특징: Episode terminal node(7-1구간 도착) reward 추가  
아직 수렴 못함



# 3. 적용 결과

## TD3 모델6

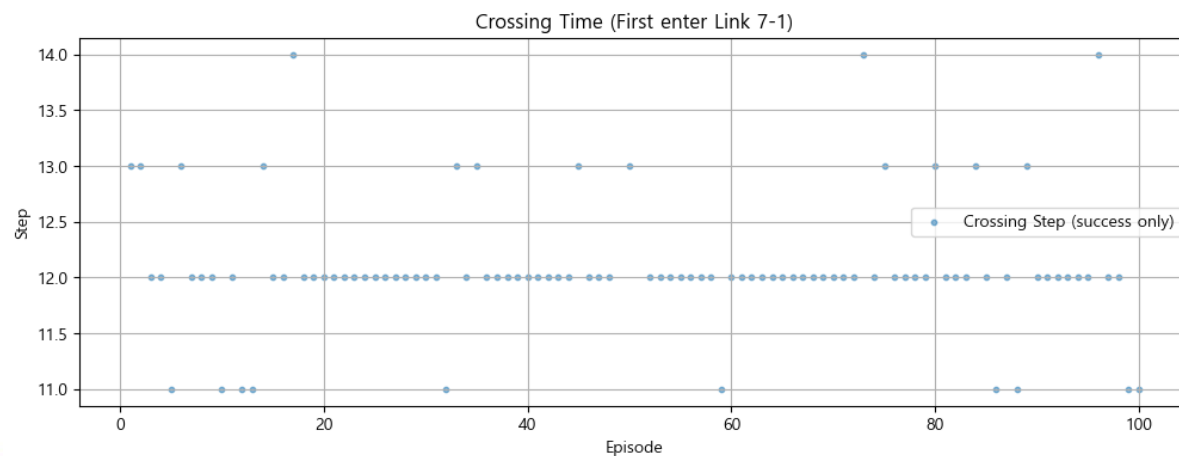
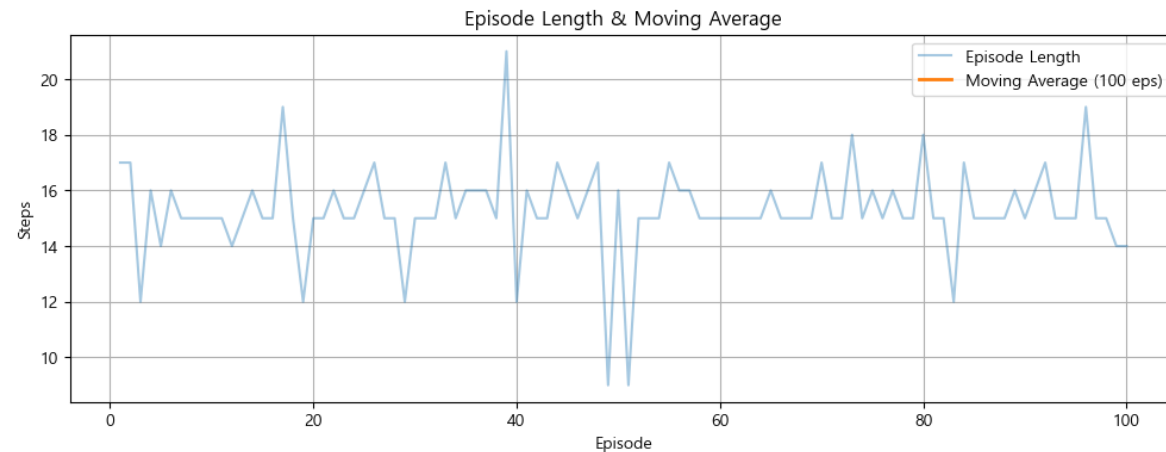


400Episodes 수행(150분)

특징: Episode terminal node(7-1구간 도착) reward 추가  
crossing, following 차량 정보 (최근접, 그다음 차량) 확대

State space = 13개

아직 수렴 못함



감사합니다.

Thank you