DEVIEW 2020

Reinforcement Learning meets the real-world: Industrial RL applications

MakinaRocks 주성호

CONTENTS

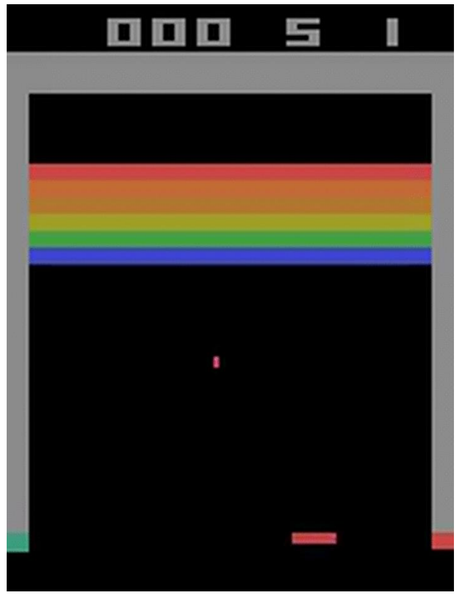
1. Real-world에서 강화학습 적용의 어려움

1.1 강화학습의 활용 영역

DEVIEW 2020

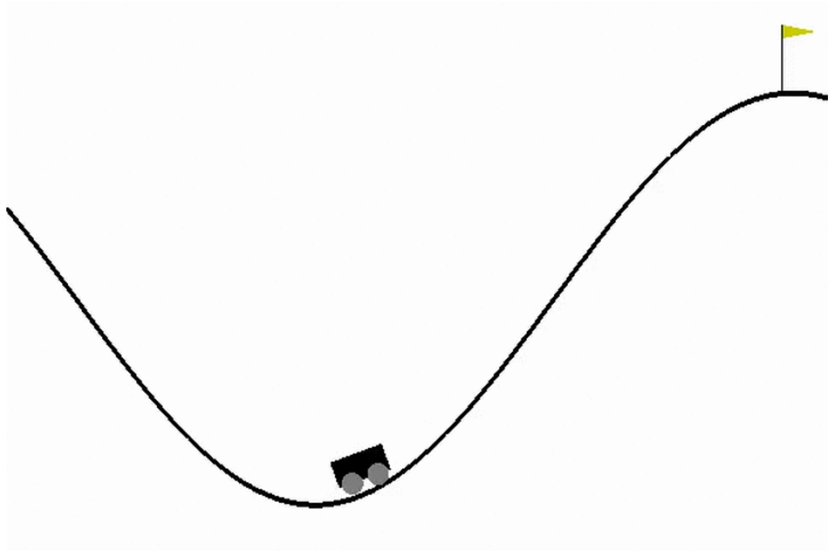
Environment

RandomAgent on Breakout-v0.



- Bellemare, M., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research, 47,* 253-279
- Mott, B., Anthony, St., & The Stella Team. (1990). Stella. Retrieved October 13, 2020, from https://stella-emu.github.io/

MountainCar-v0.



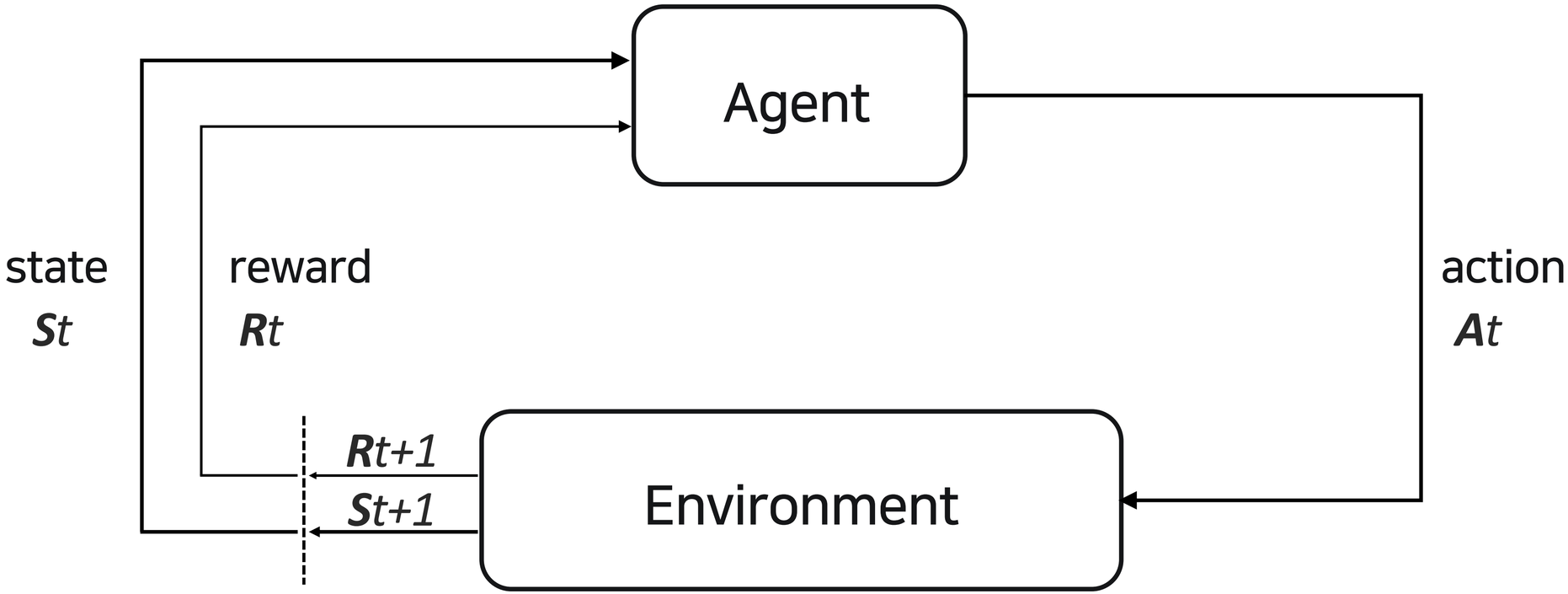
- Moore, A. (1990). *Efficient Memory-Based Learning for Robot Control* (Doctoral dissertation, University of Cambridge, Cambridge, United Kingdom).

Available from https://gym.openai.com/envs/

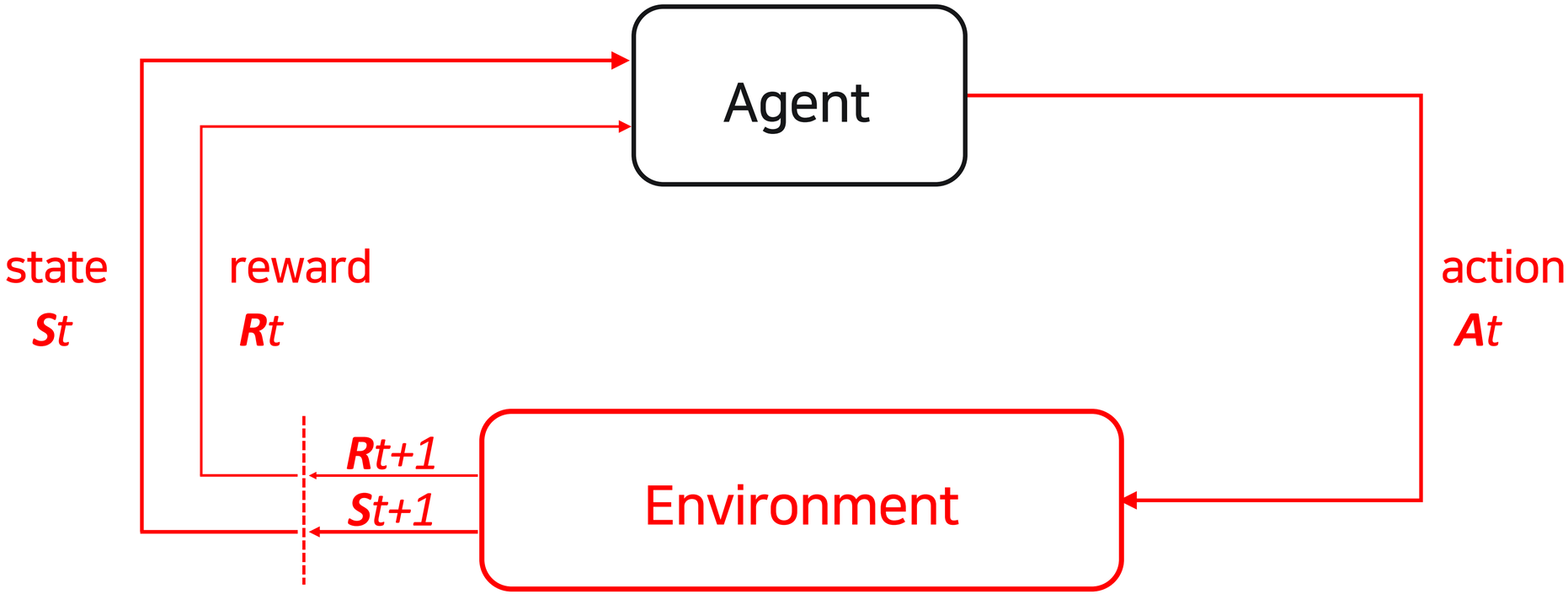
- PPO
- SAC
- HIRO
- Curiosity
- etc.

Agent

1.2 강화학습의 구성 요소



1.2 강화학습의 구성 요소



1.3 강화학습 적용의 어려움

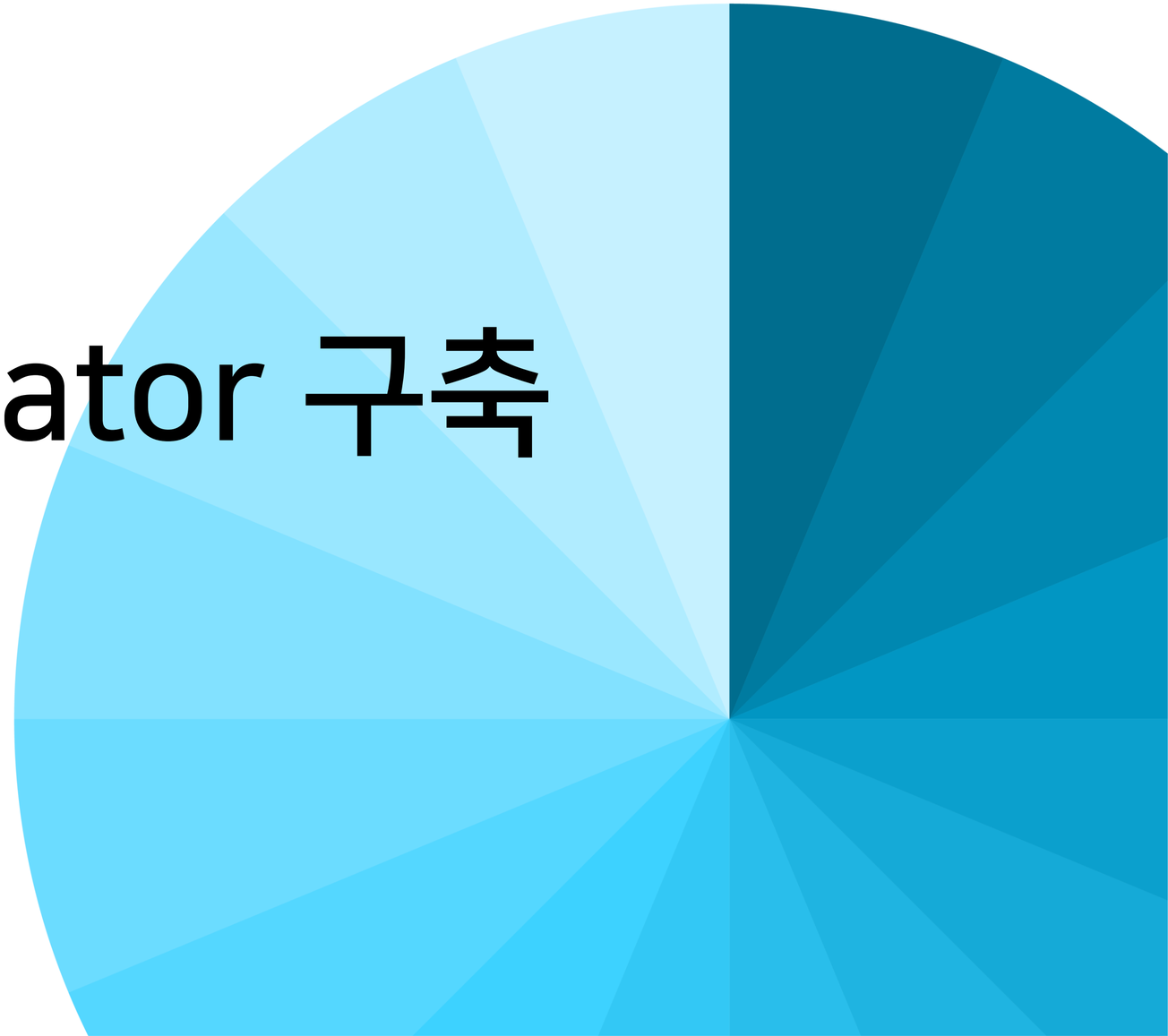
Agent
(강화학습 모델)

Simulator

Simulator 외

- Reward 설계
- Action 구현
- State Representation

2. Simulator 구축

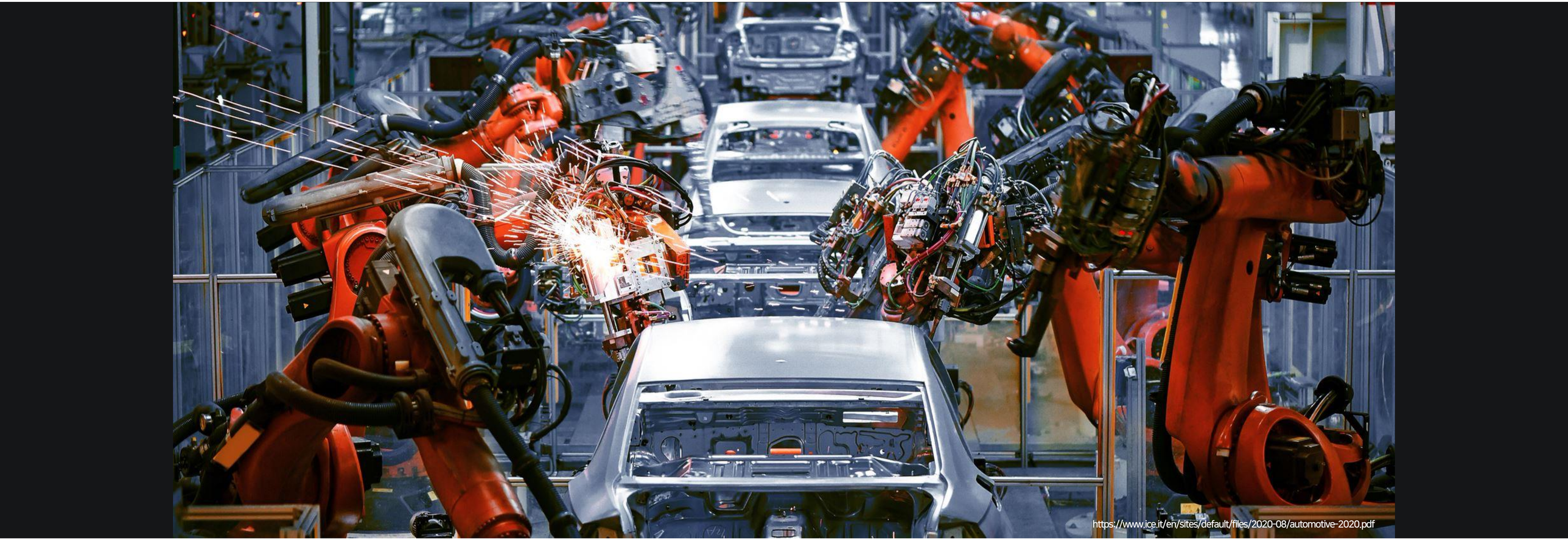


2.1 Simulator 구현 방법

2.2 Simulator 제작

DEVIEW 2020

사례: 로봇팔 Path Planning



2.2 Simulator 제작

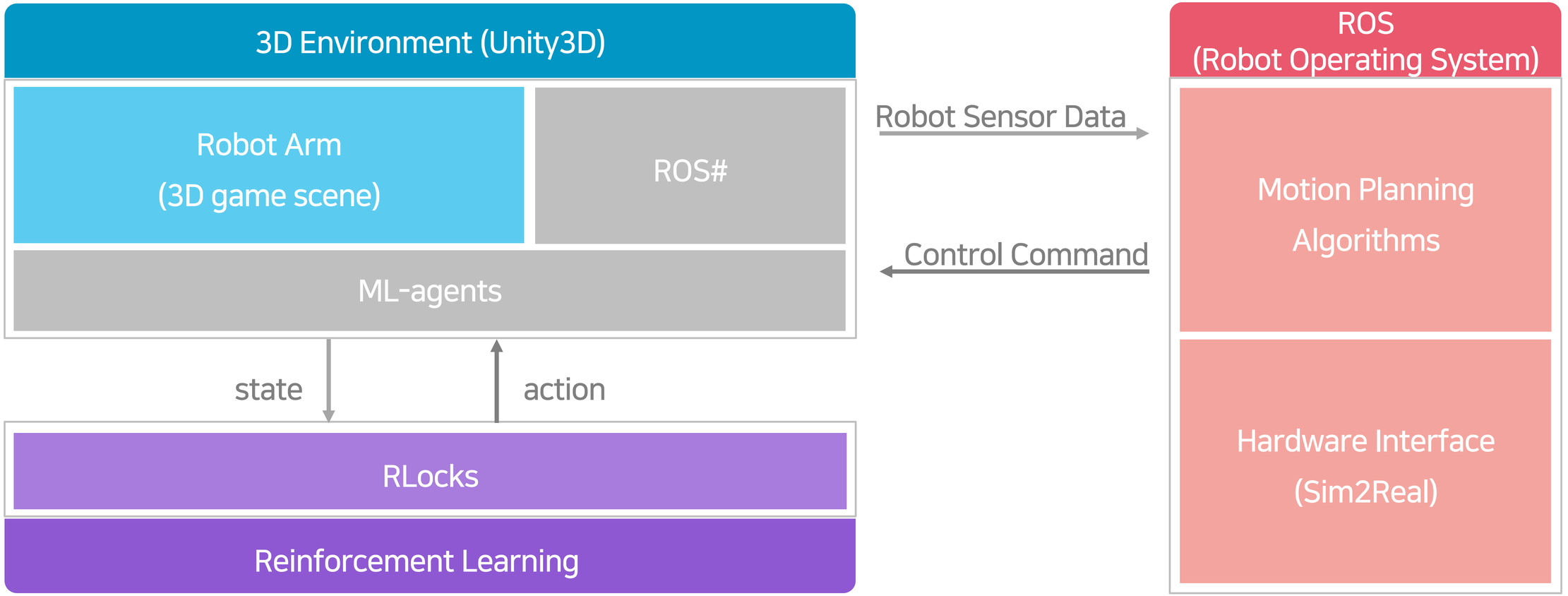
DEVIEW 2020

Simulator를 제작한 이유?

- 실제 환경 반영의 어려움
- 다양한 알고리즘 접목의 어려움
- 제한적인 구동 환경

2.2 Simulator 제작

실제 환경을 반영할 수 있는 Simulator 구성



2.2 Simulator 제작

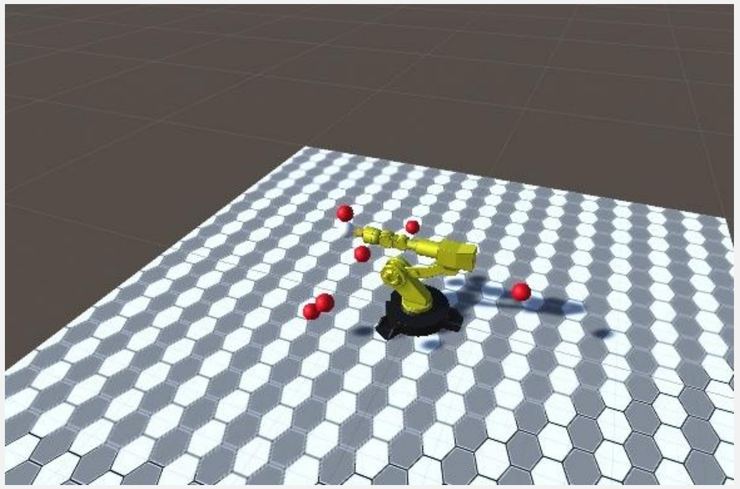
DEVIEW 2020

실제 환경을 반영할 수 있는 Simulator 구성

2.2 Simulator 제작

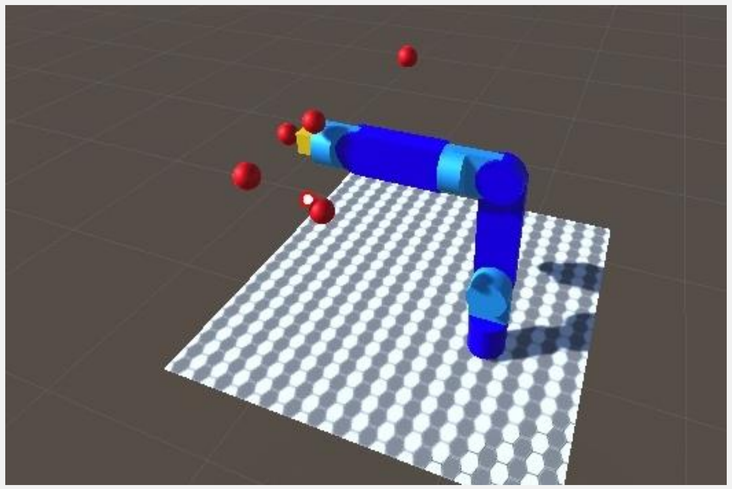
실제 환경을 반영할 수 있는 Simulator 구성

1차 실험 환경



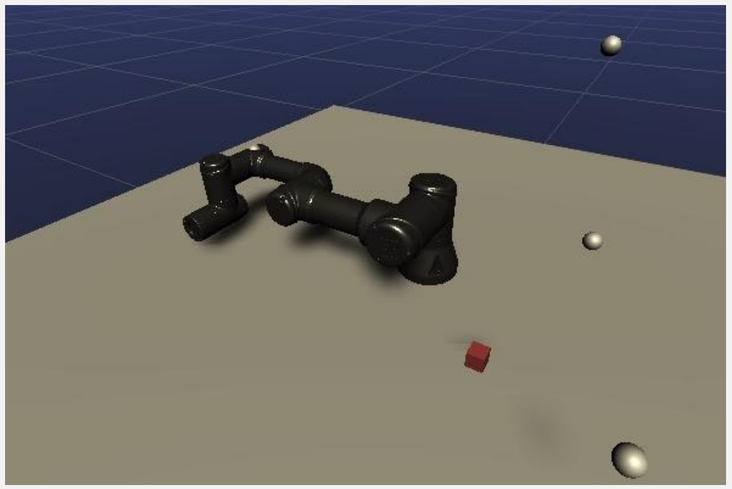
- 4 DOF
- 조인트 토크 제어

2차 실험 환경(IK)



- 6 DOF
- Inverse Kinematics

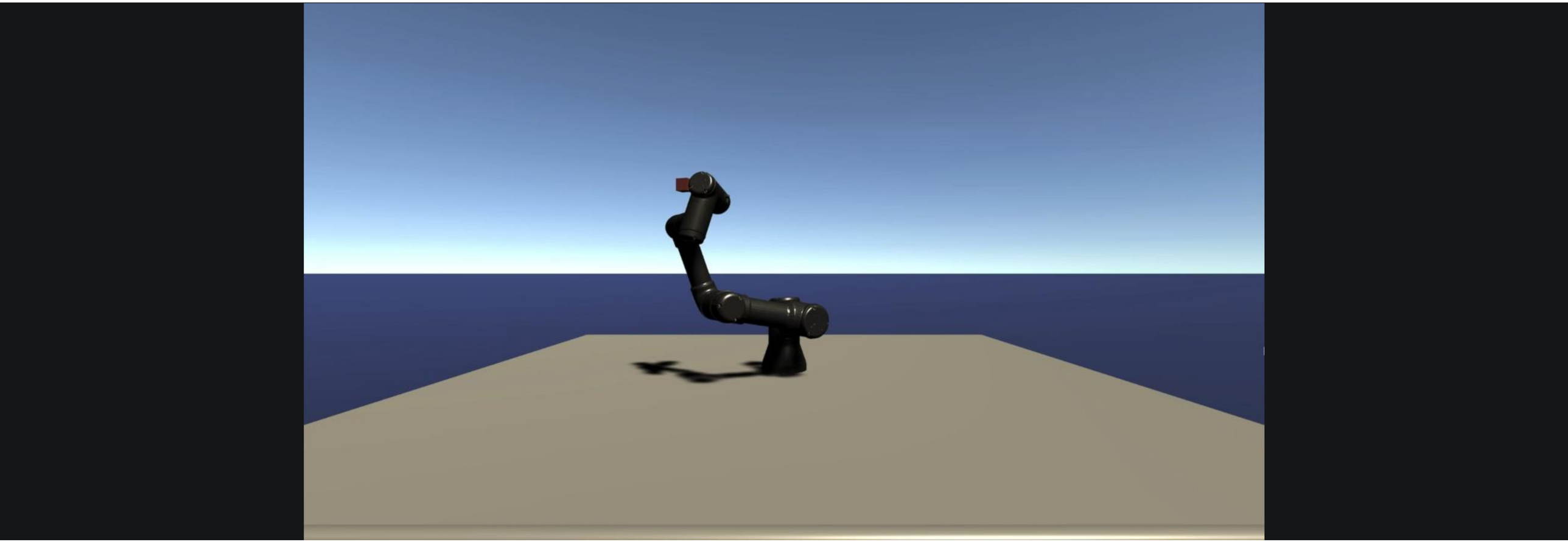
3차 실험 환경 (UR3 model + ROS)



- 6 DOF
- Real robot
- 조인트 각도 및 각속도 제어

2.2 Simulator 제작

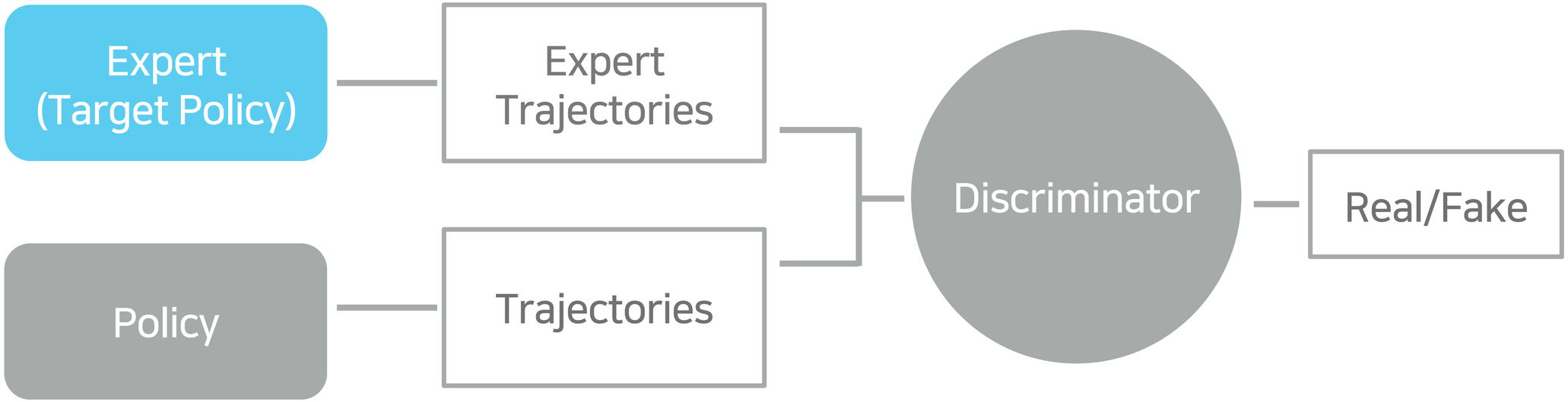
실제 환경을 반영할 수 있는 Simulator 구성



2.2 Simulator 제작

다양한 강화학습 알고리즘 접목

GAIL(Generative Adversarial Imitation Learning) Trajectory 생성

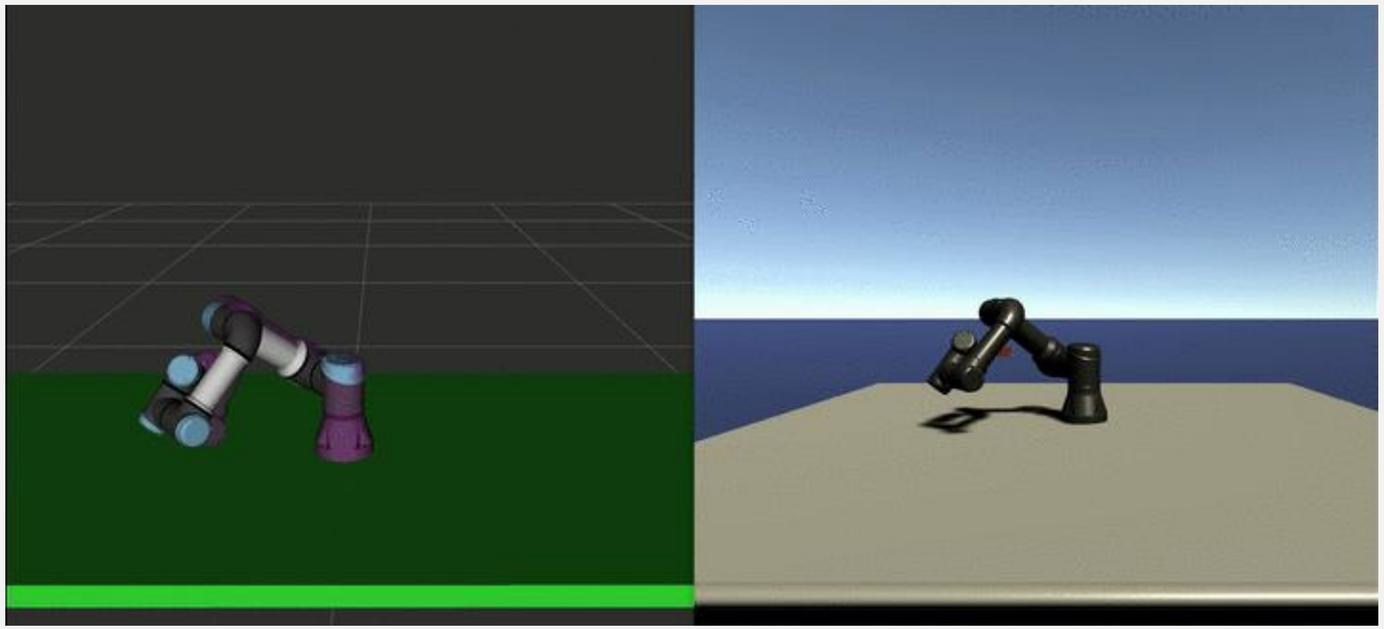


- 강화학습에서 바로 사용할 수 있는 형태의 Handmade Expert Trajectories는 현실적으로 얻기 어려움
- Moveit라는 Planning 오픈소스를 이용하여 Expert Trajectories 생성

2.2 Simulator 제작

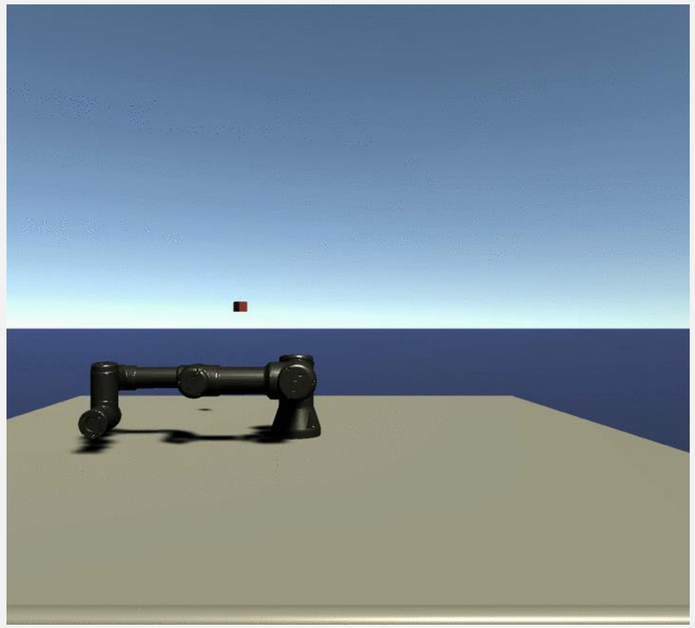
다양한 강화학습 알고리즘 접목

ROS + Moveit | Unity 3D



Trajectory Sampling

Unity 3D



Demo with GAIL

2.2 Simulator 제작

제한적인 구동 환경

속도 저하의 이유

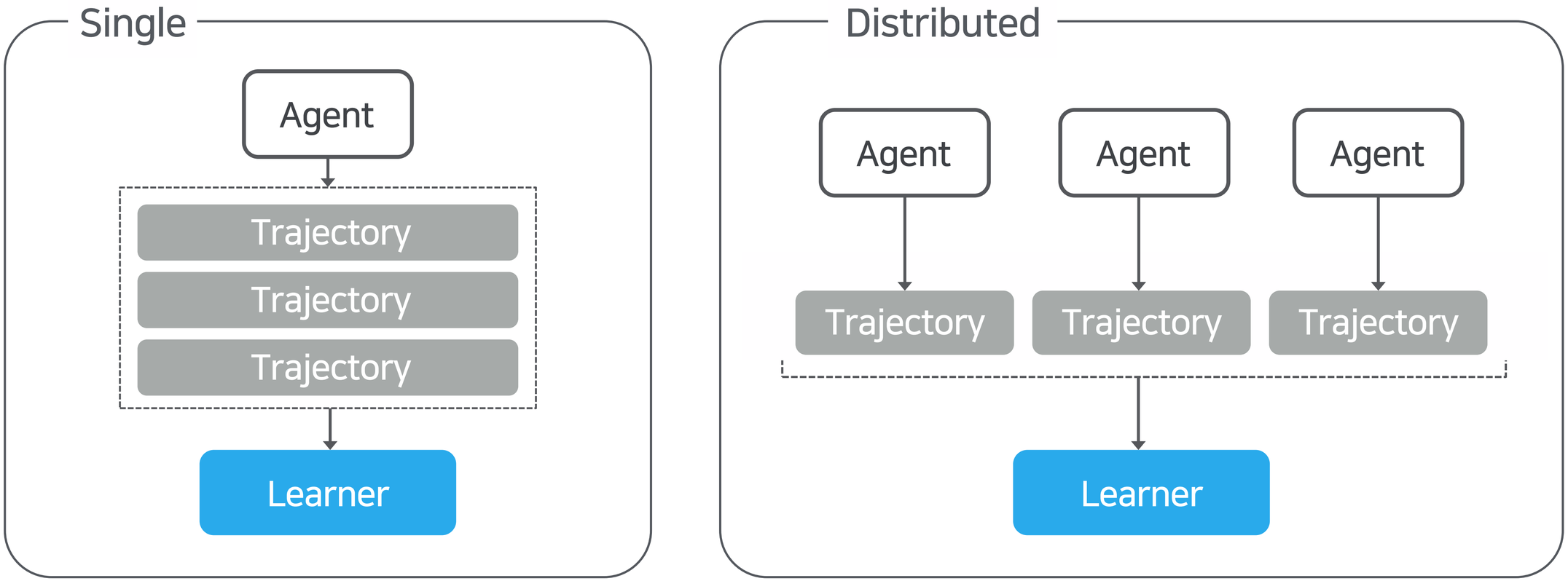
Simulator와의 상호작용
과정에서 병목 발생

해결 방안

Parallel Learning Algorithm
Distributed Learning Env.
(Ray)

2.2 Simulator 제작

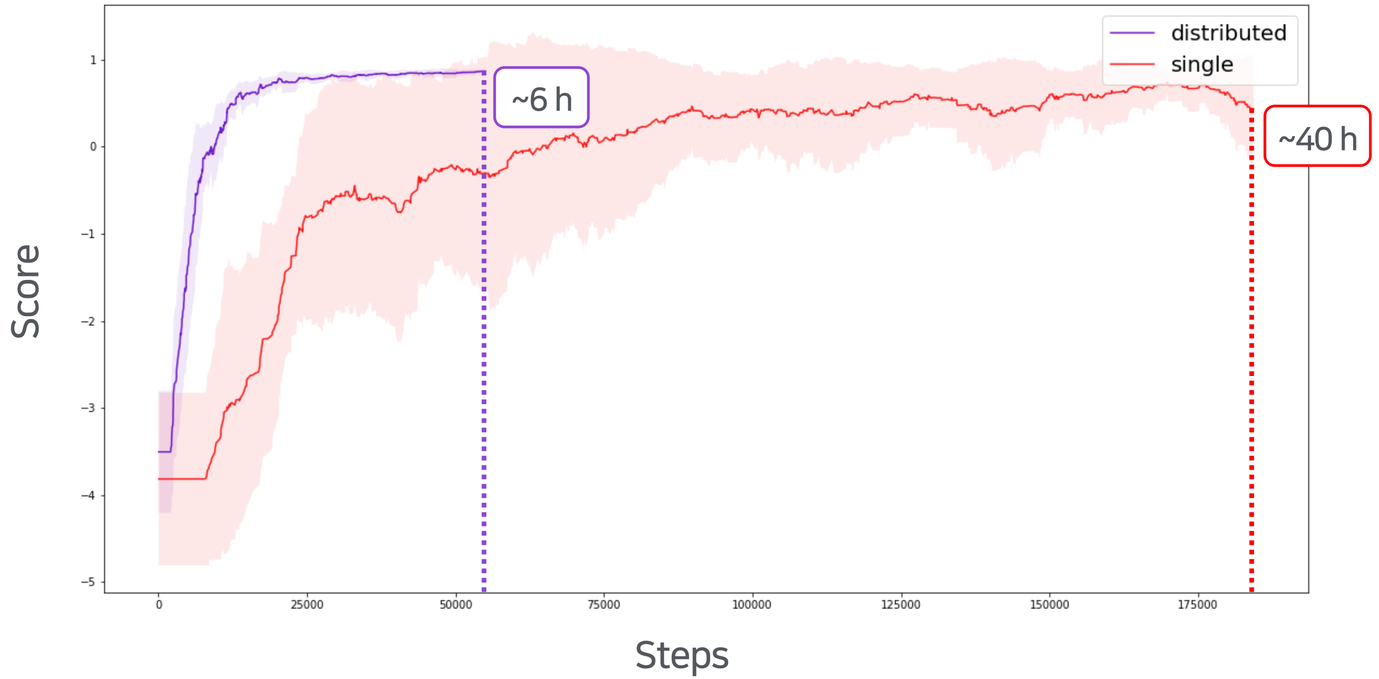
제한적인 구동 환경



2.2 Simulator 제작

학습 속도 개선

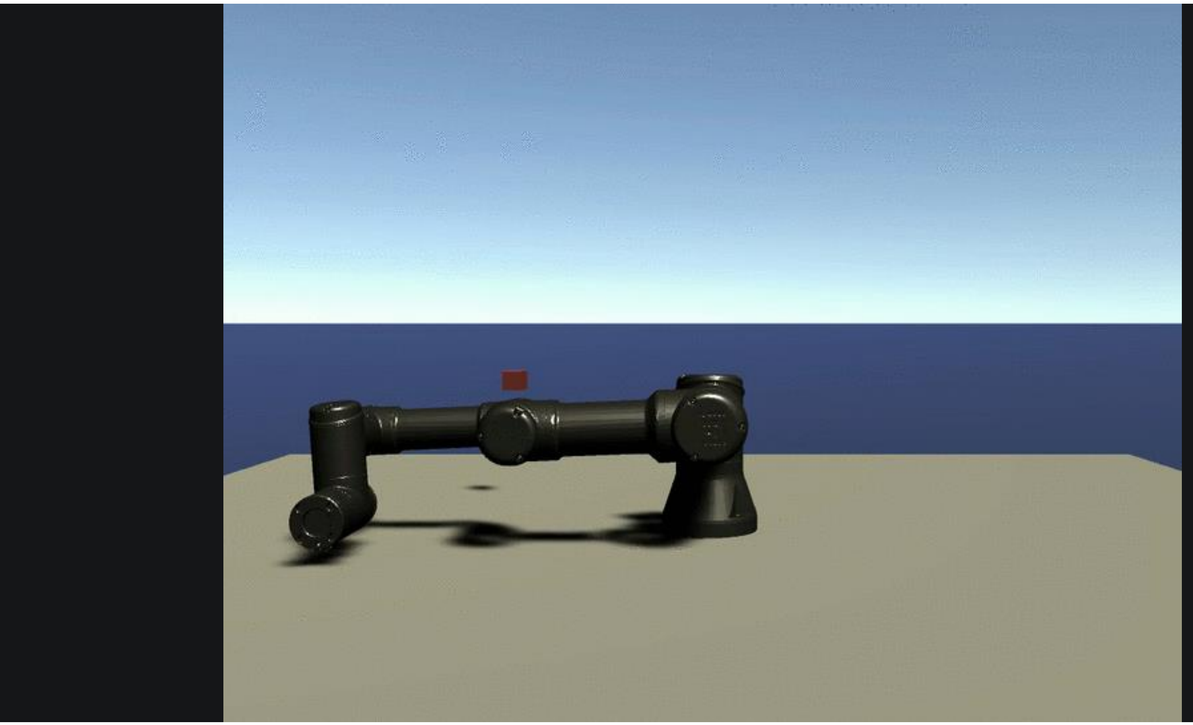
- Learning in distributed environment result
- Algorithm: PPO (MakinaRocks RLocks library)
- Distributed communication: Ray library (8 agents)



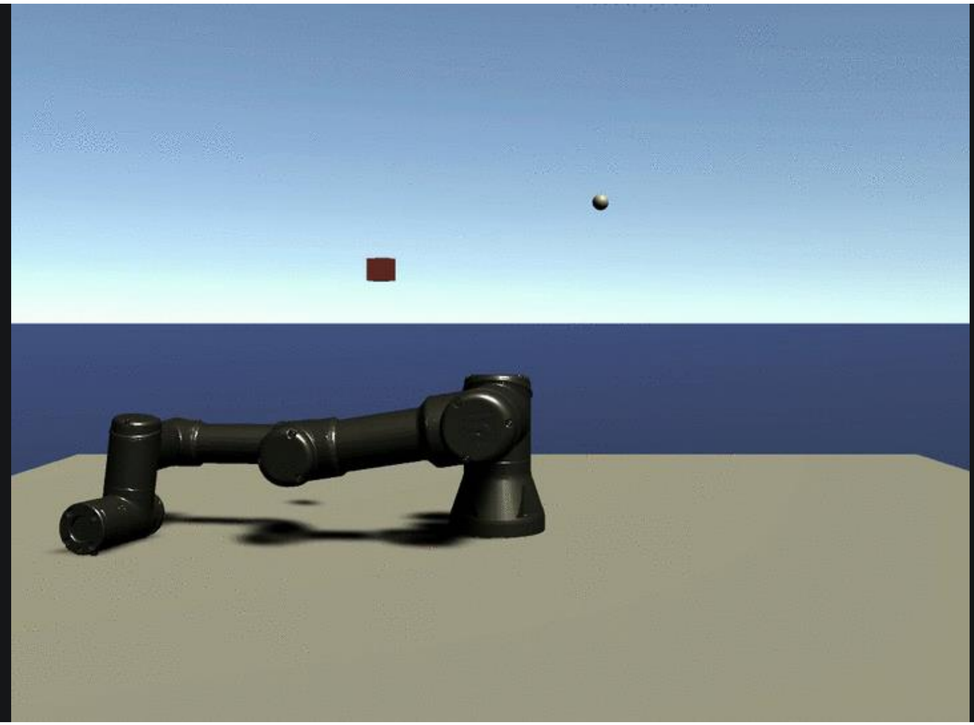
2.2 Simulator 제작

학습 속도 개선

- Training efficiency: 6 hours



Single Agent Demo



Distributed Demo

2.3 산업용 Simulator 사용

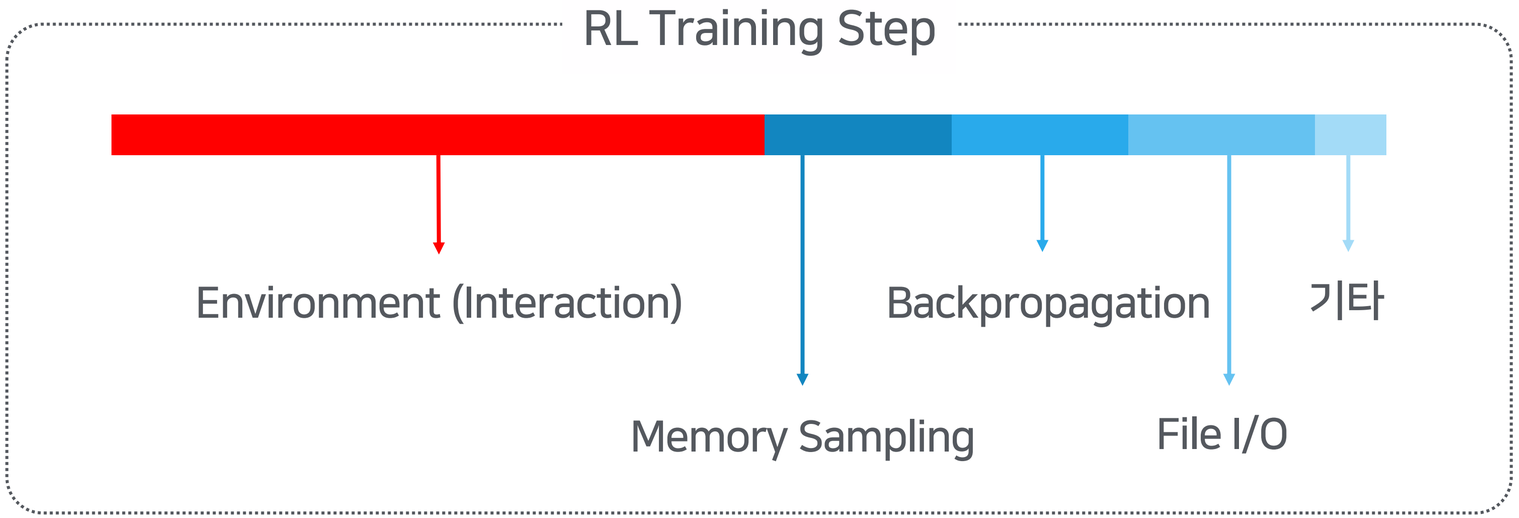
산업용 Simulator를 이용할 때 가장 큰 Bottleneck은?

- 미분방정식 기반 Simulator의 연산 속도
- 높은 불확실성
- 머신러닝 학습 연동

2.3 산업용 Simulator 사용

경험 사례

- Simulation 시간 1초 -> 실제 소요 시간 7~10초
- 한 번의 학습에 1주일 소모
- 라이센스 문제로 병렬 처리의 한계 존재



3. Simulator 외 환경 구축

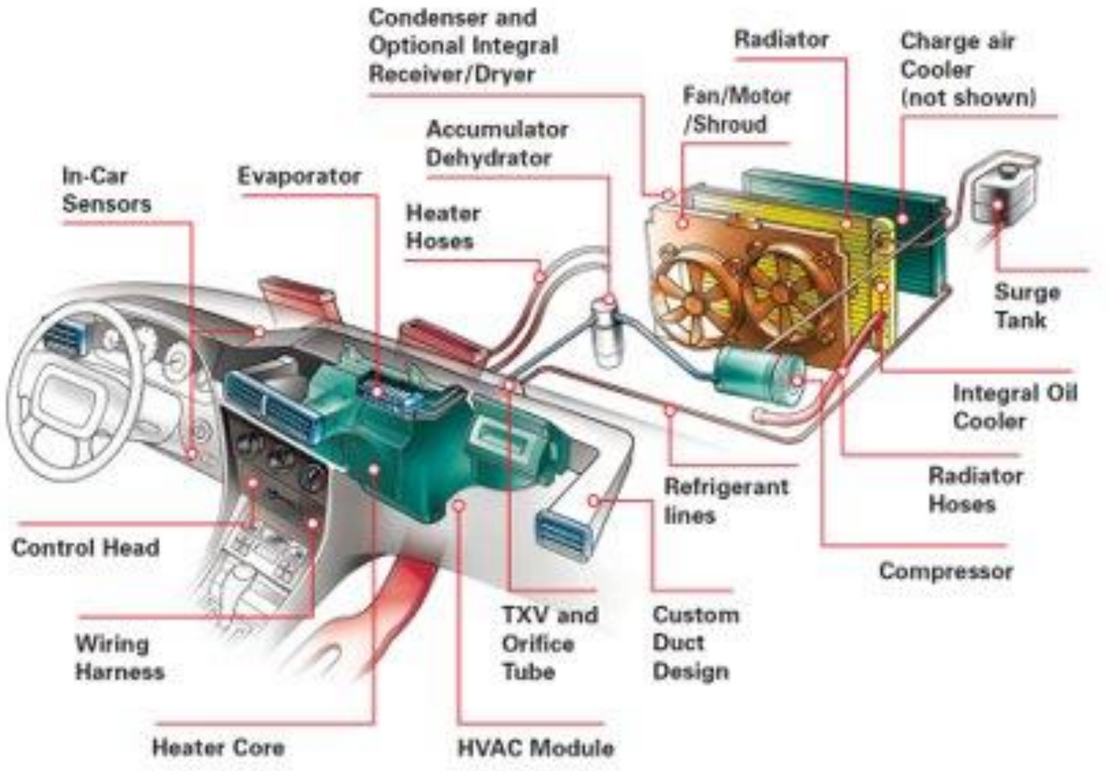
3.1 EMS 최적 제어

전기차 에너지 관리 시스템(EMS)

https://pixabay.com/ko/photos/%EC%B0%A8%EB%9F%89-%EC%9E%90%EB%8F%99%EC%B0%A8-%EC%97%90-%EC%96%B4-%EC%BB%A8%EB%94%94%EC%85%94%EB%84%88-3872550/

3.1 EMS 최적 제어란?

Compressor, Fan 등 복수의 Components를 제어해 안전하고 효율적으로 온도를 제어하는 것



Han, T., Huang, L., Kelly, S., Huizenga, C., & Hui, Z. (2001). Virtual Thermal Comfort Engineering. *SAE Transactions, 110,* 582-591. Retrieved October 22, 2020, from http://www.jstor.org/stable/44730913

3.1 EMS 제어 시 문제들

DEVIEW 2020

- Simulator 속도
- 동시에 충족해야 하는 목표들
- 온도에 따른 요구사항의 변동

Reward 설계

- 실제 하드웨어의 제약

Action 구현

- 사용하지 못한 Feature
- 제어의 진동하는 경향

State Representation

3.2 Reward 설계

설계 시 고려사항

문제점

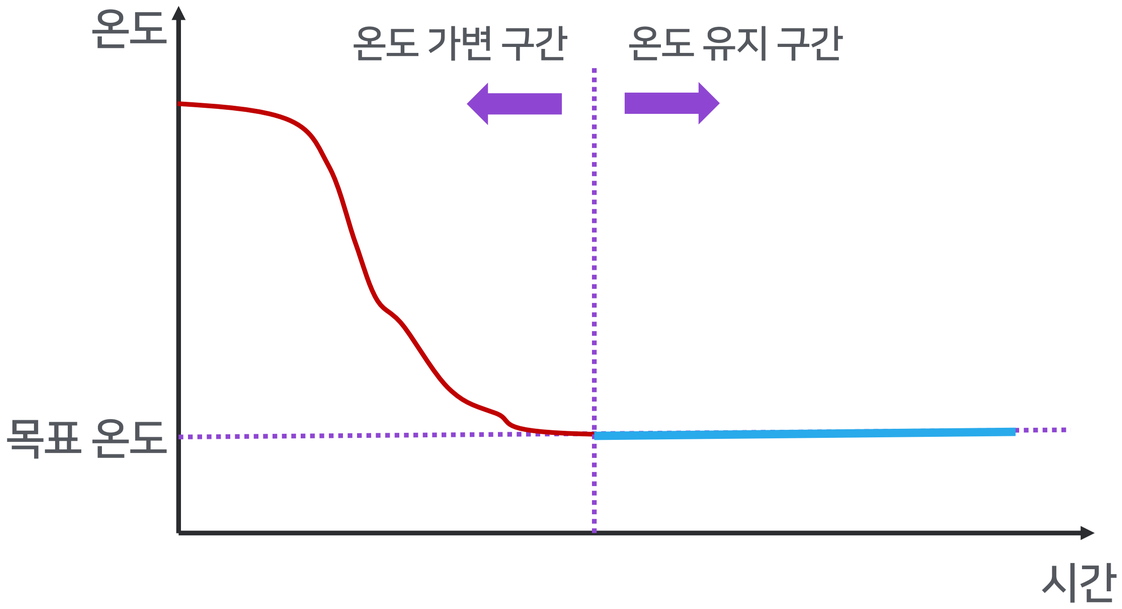
- Simulator 속도
- 동시에 충족해야 하는 목표들
- 온도에 따른 요구사항의 변동

해결 방안

- Sparse -> Shaped reward
- Relative size of rewards
- Negative/Positive

3.2 Reward 설계

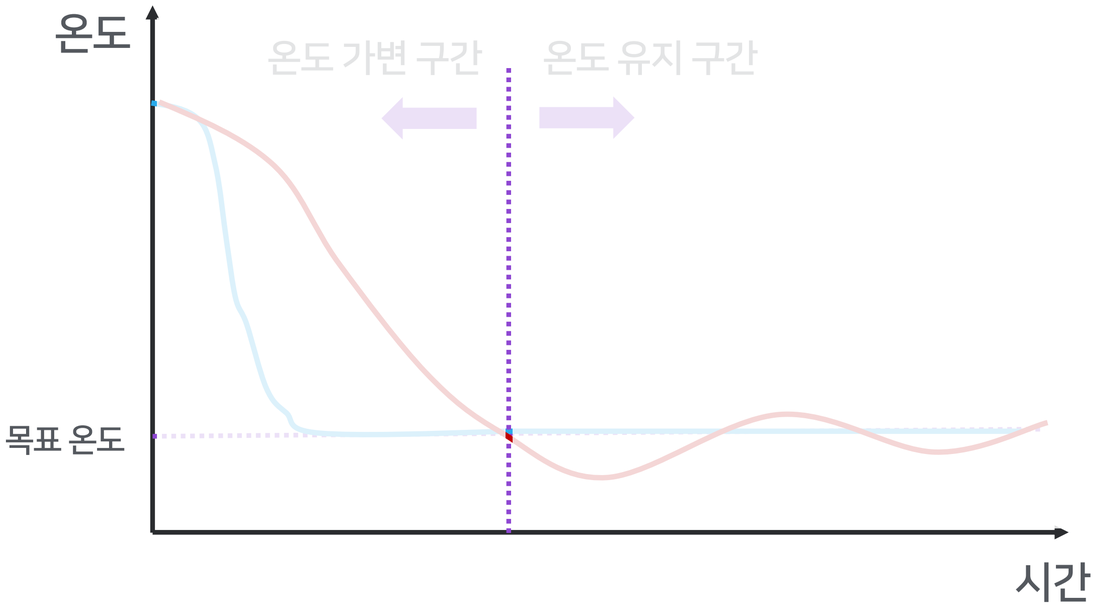
Negative / Positive Reward 반영

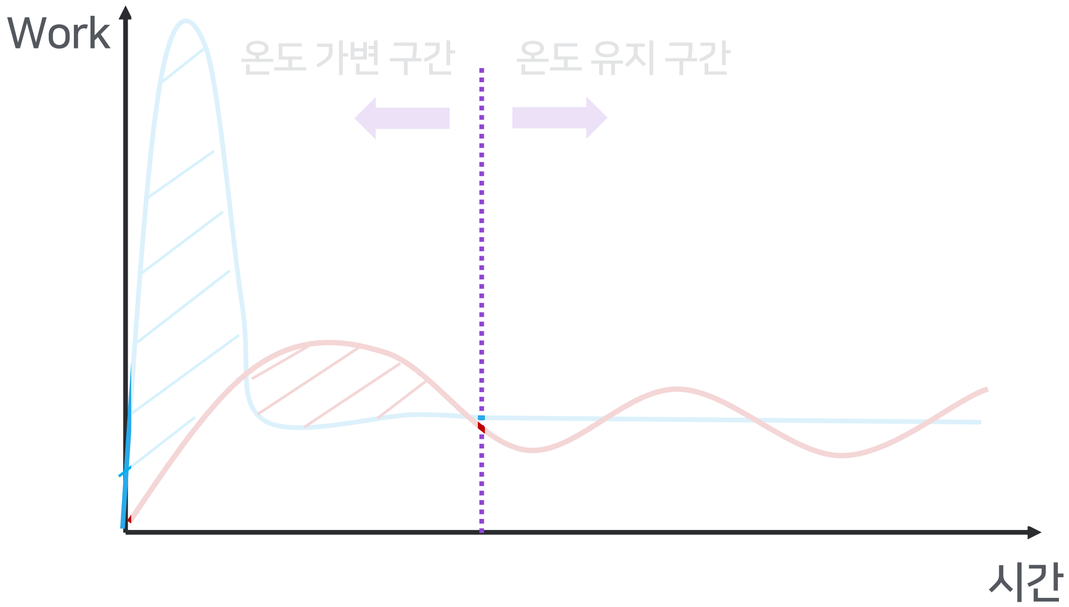


	기대 결과
온도 가변 구간 (Negative reward)	온도 수렴 + 효율 + 빠른 수렴 속도 Action trajectory
온도 유지 구간 (Positive reward)	온도 유지 + 최대 효율 Action 조합

3.2 Reward 설계

Negative / Positive Reward 반영





- Positive Reward

1. 온도 수렴 후 유지 능력
2. 온도 가변 구간 work 소모 ↑
3. 온도 수렴 구간에 유리

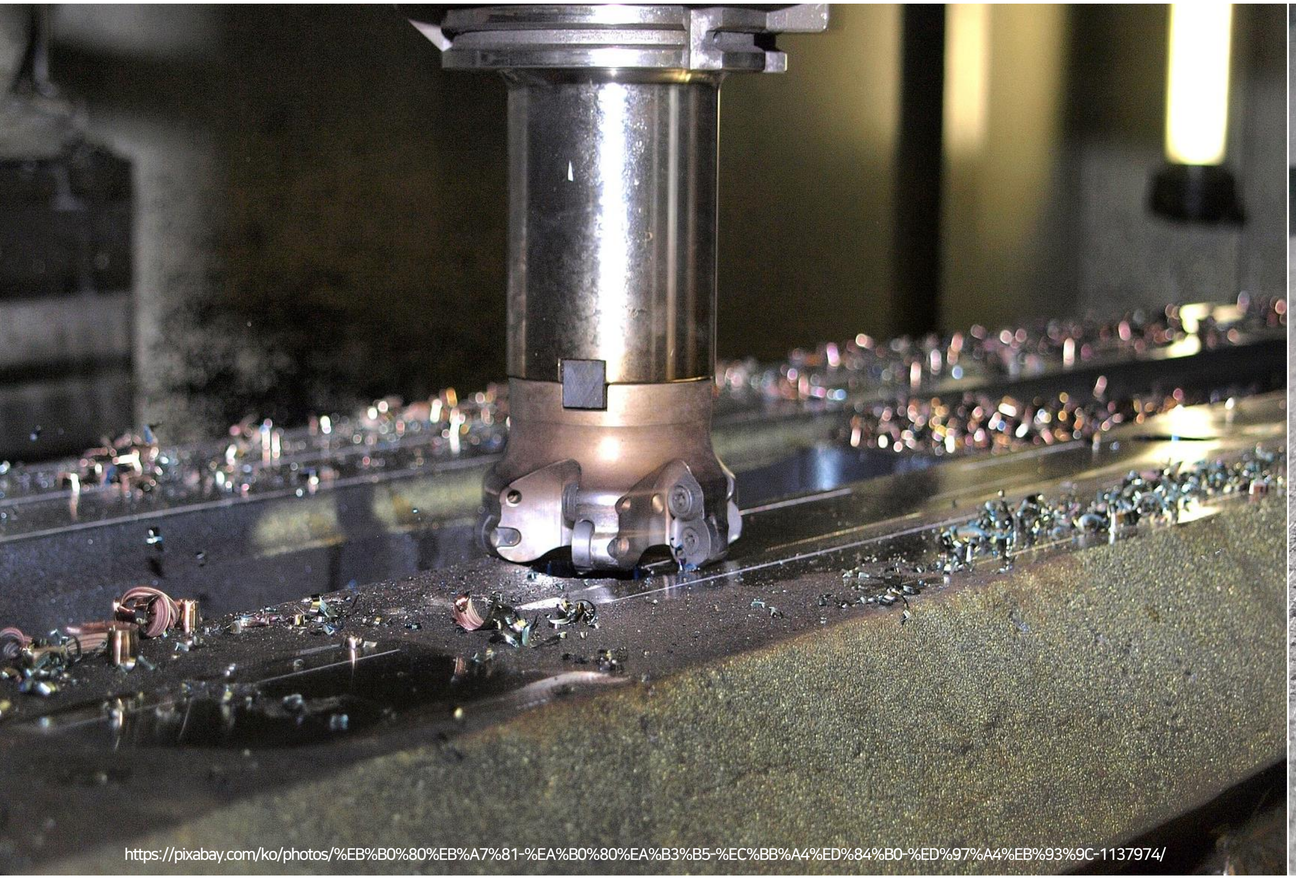
- Negative Reward

1. 목표 온도 주변에서 진동하는 경향
2. 온도 가변 구간 work 소모 ↓
3. 온도 가변 구간에 유리

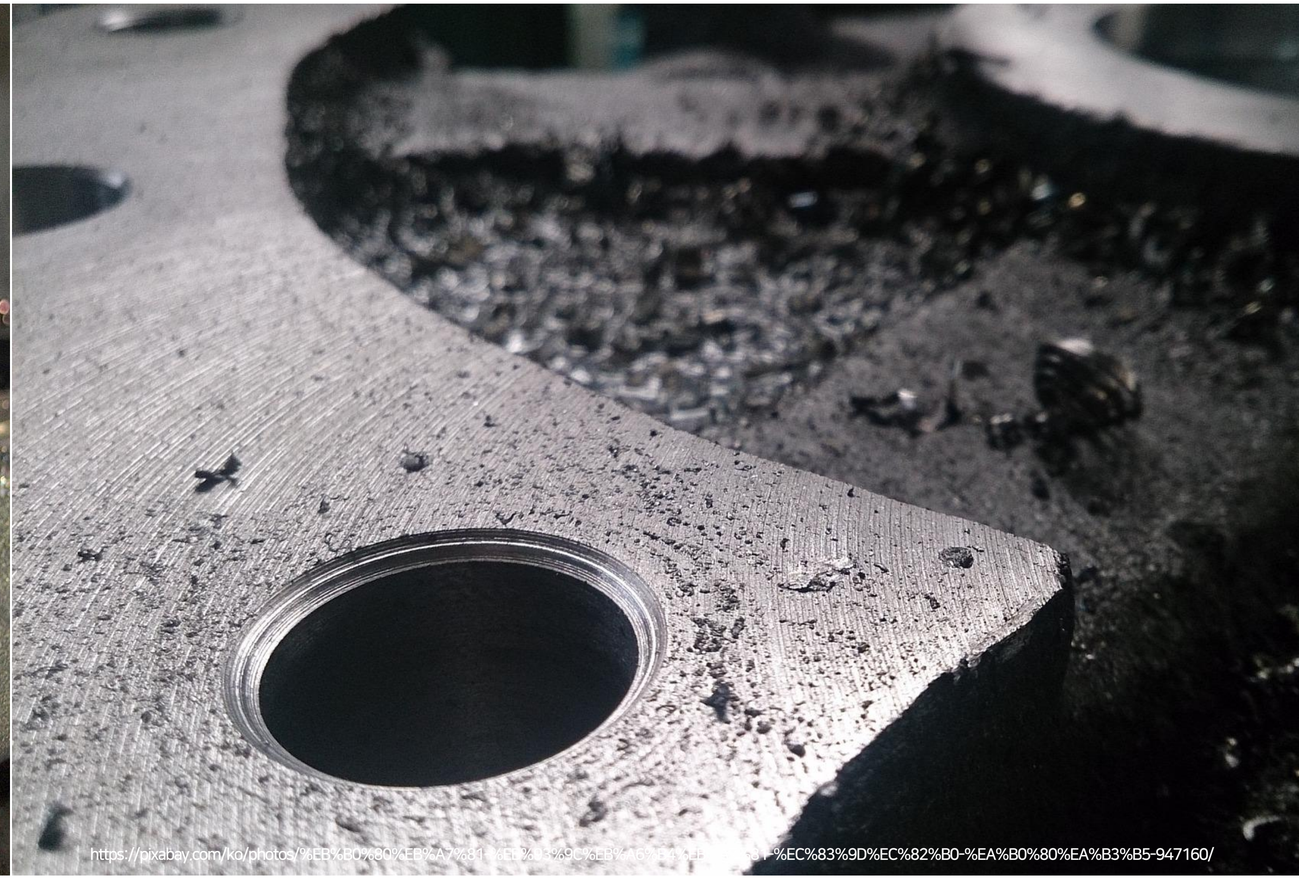
3.2 Reward 설계

DEVIEW 2020

Constrained MDP



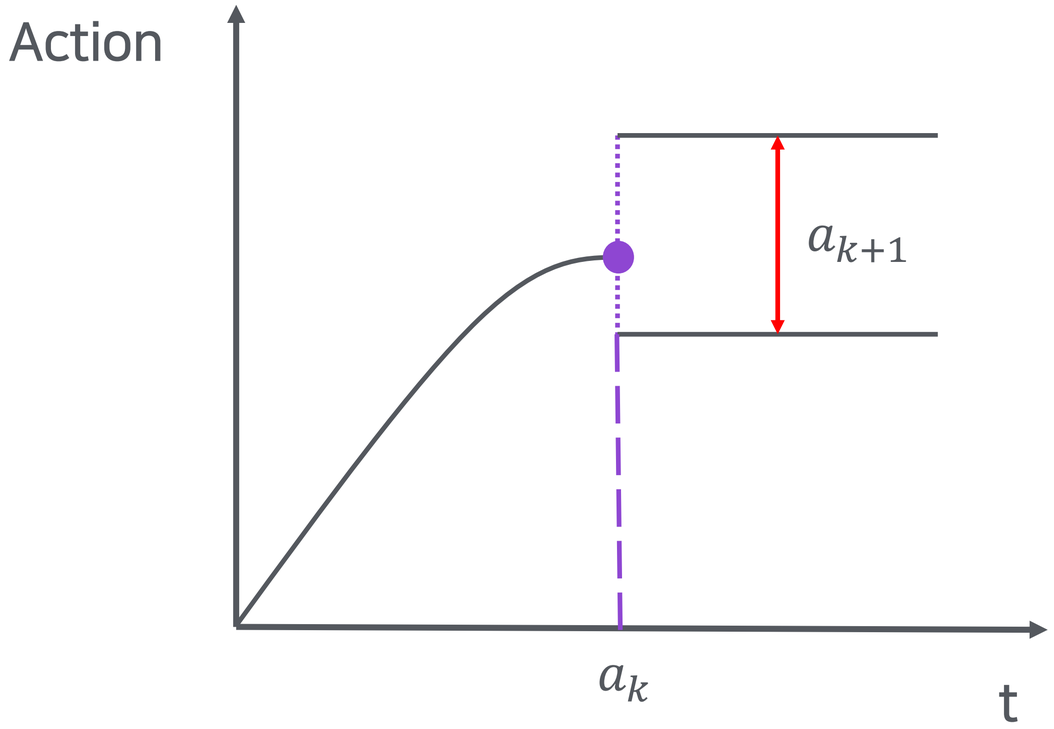
https://pixabay.com/ko/photos/%EB%B0%80%EB%A7%81-%EA%B0%80%EA%B3%B5-%EC%BB%A4%ED%84%B0-%ED%97%A4%EB%93%9C-1137974/



https://pixabay.com/ko/photos/%EB%B0%80%EB%A7%81-%EA%B3%B5%EC%9E%91%EA%B8%B0%EA%B3%84-%EC%83%9D%EC%82%B0-%EA%B0%80%EA%B3%B5-947160/

3.3 Action 구현

제어 시 고려해야 하는 Hardware Constraint



3.3 Action 구현

DEVIEW 2020

제어 시 고려해야 하는 Hardware Constraint

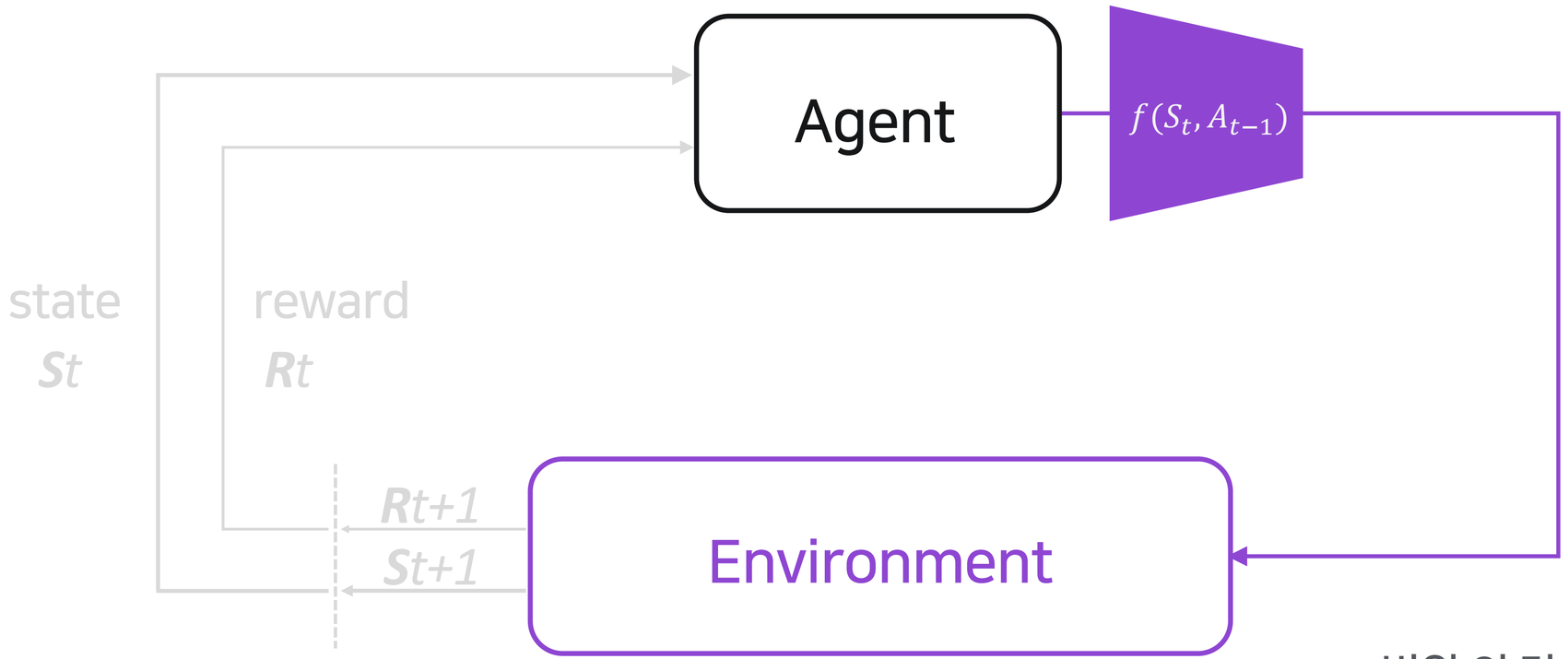
$100\ km/h$

Δt

$80{\sim}130\ km/h$

3.3 Action 구현

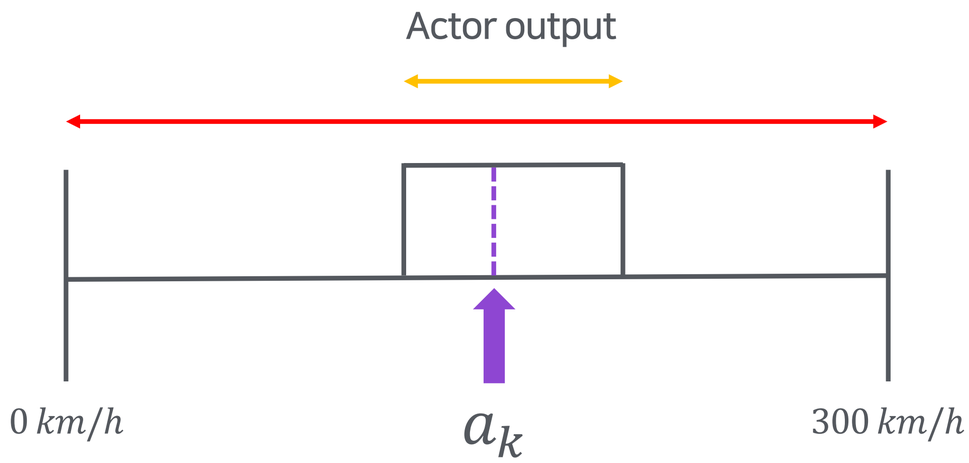
Constrained MDP 구현 방식



- 범위 외 값 제거 (Clipping)
- 범위 재조정 (Rescaling)

3.3 Action 구현

Constrained MDP 구현 방식



- Clipping

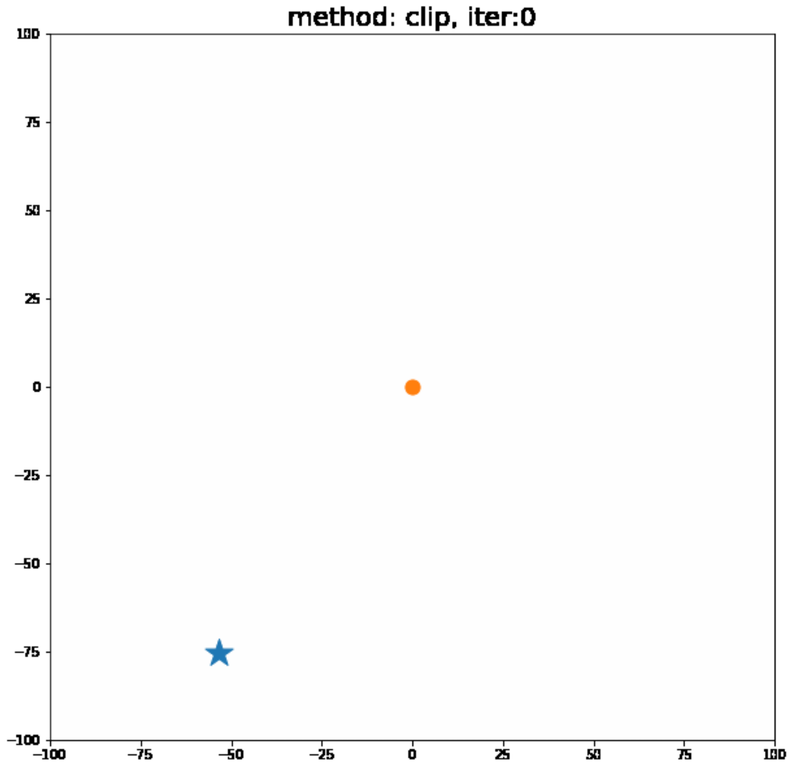
Actor output : $[-1, 1] \rightarrow [0, 300] \rightarrow [a_k - \lambda_l,\ a_k + \lambda_h]$

- Rescaling

Actor output : $[-1, 1] \rightarrow [a_k - \lambda_l,\ a_k + \lambda_h]$

3.3 Action 구현

Constrained MDP 적용 결과



State : [이전 위치, 현재 위치, 타겟과의 차이, 이전 action]

Action : 공의 속도(x_{vel}, y_{vel})

Reward : $reward_{dist} + reward_{energy}$

1. $reward_{dist} \propto -distance$

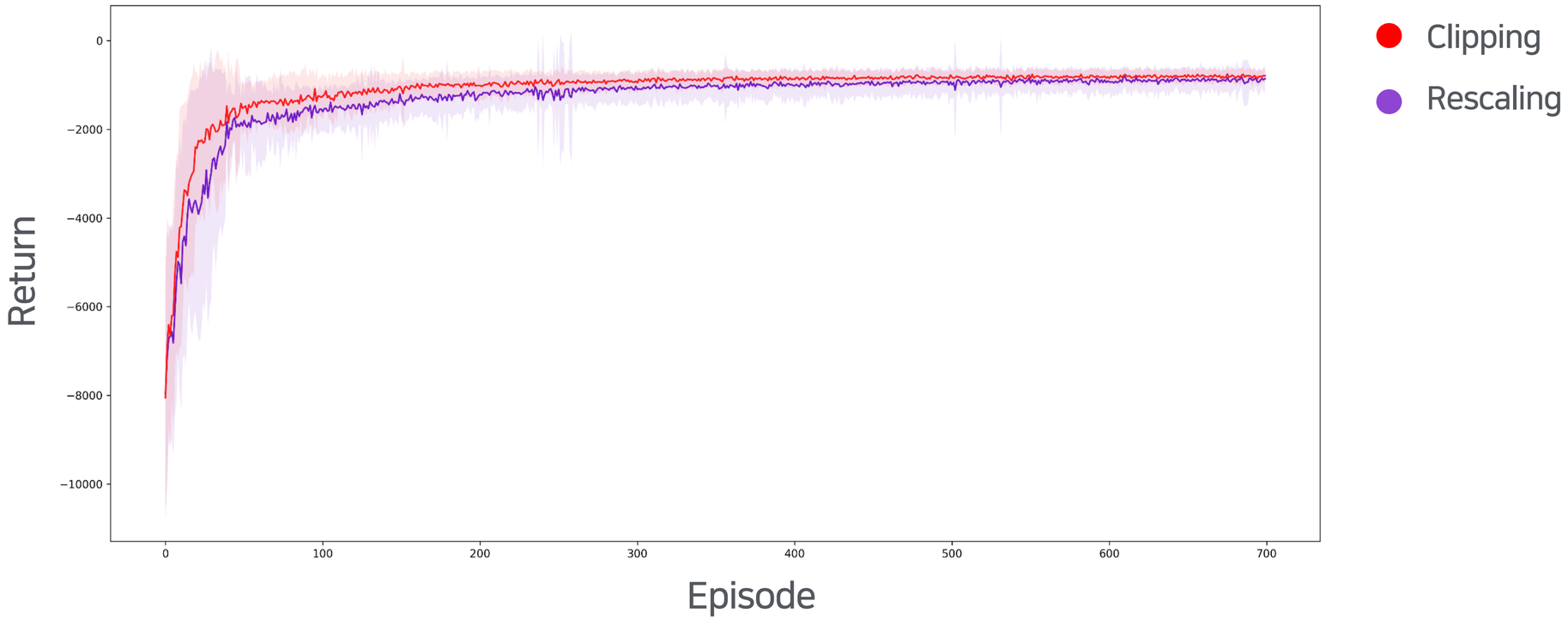
2. $reward_{energy} \propto -v^2$

Training algorithm : SAC

3.3 Action 구현

DEVIEW 2020

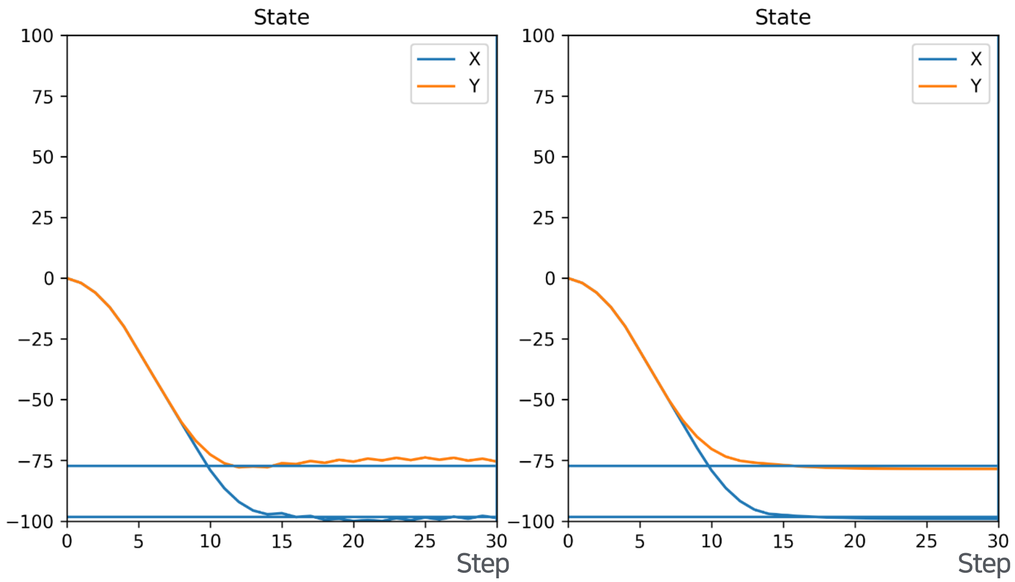
Constrained MDP 적용 결과



3.3 Action 구현

Constrained MDP 적용 결과

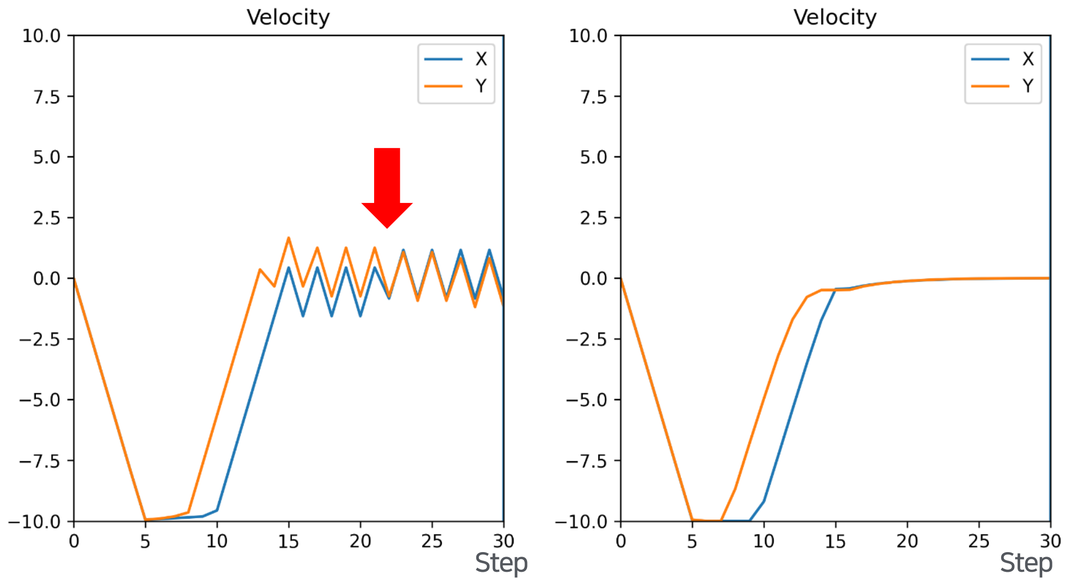
좌표



Clipping

Rescaling

Action

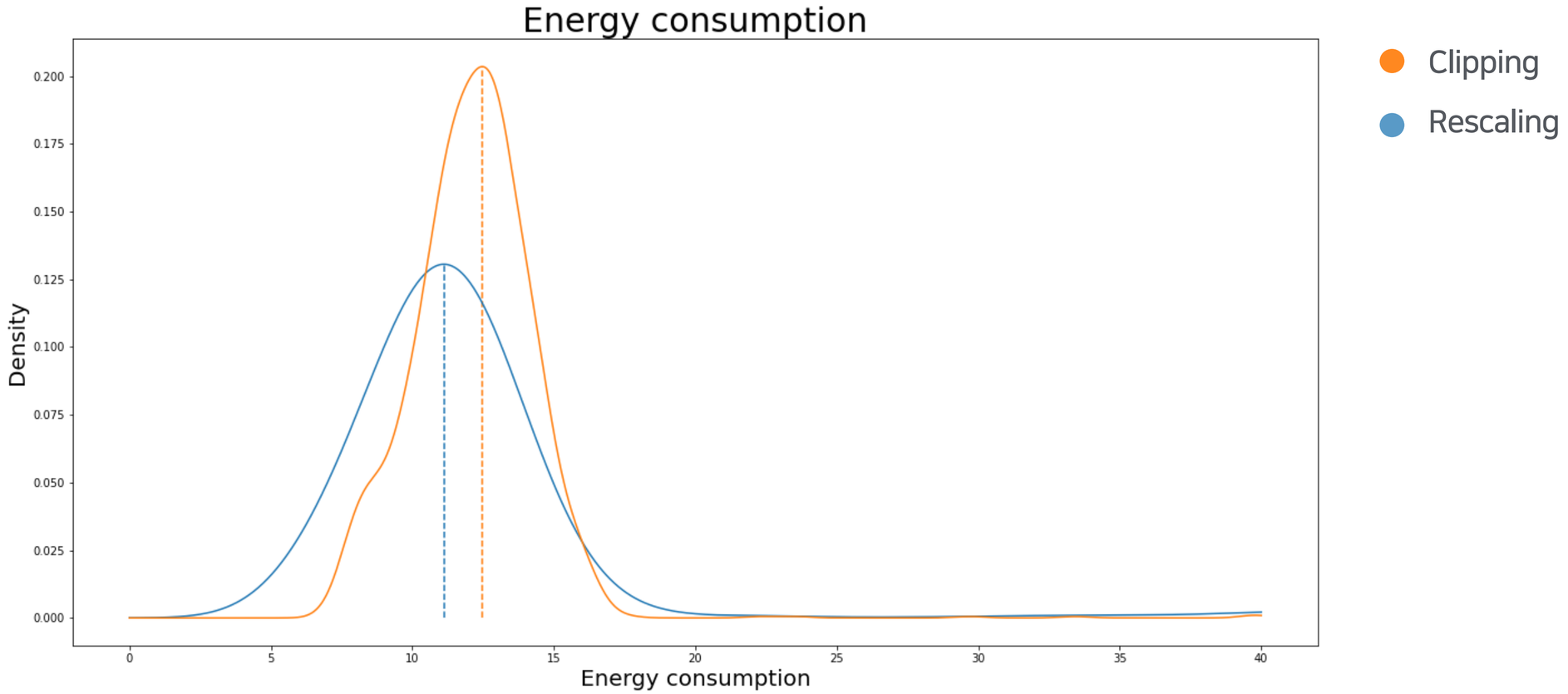


Clipping

Rescaling

3.3 Action 구현

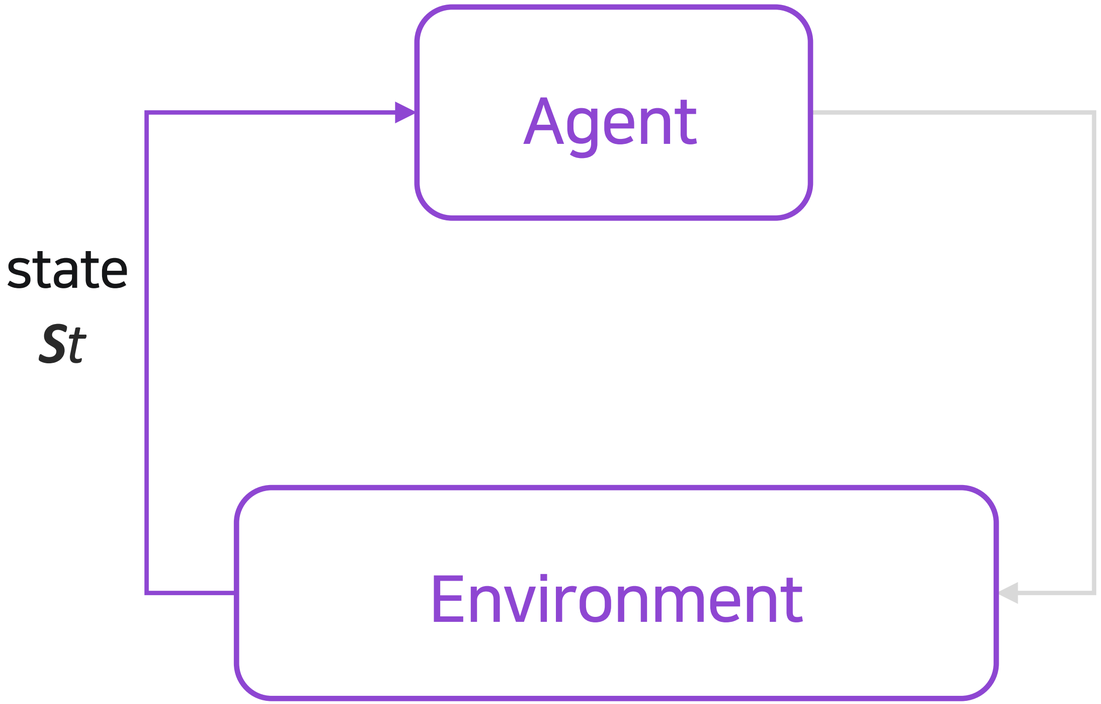
Constrained MDP 적용 결과

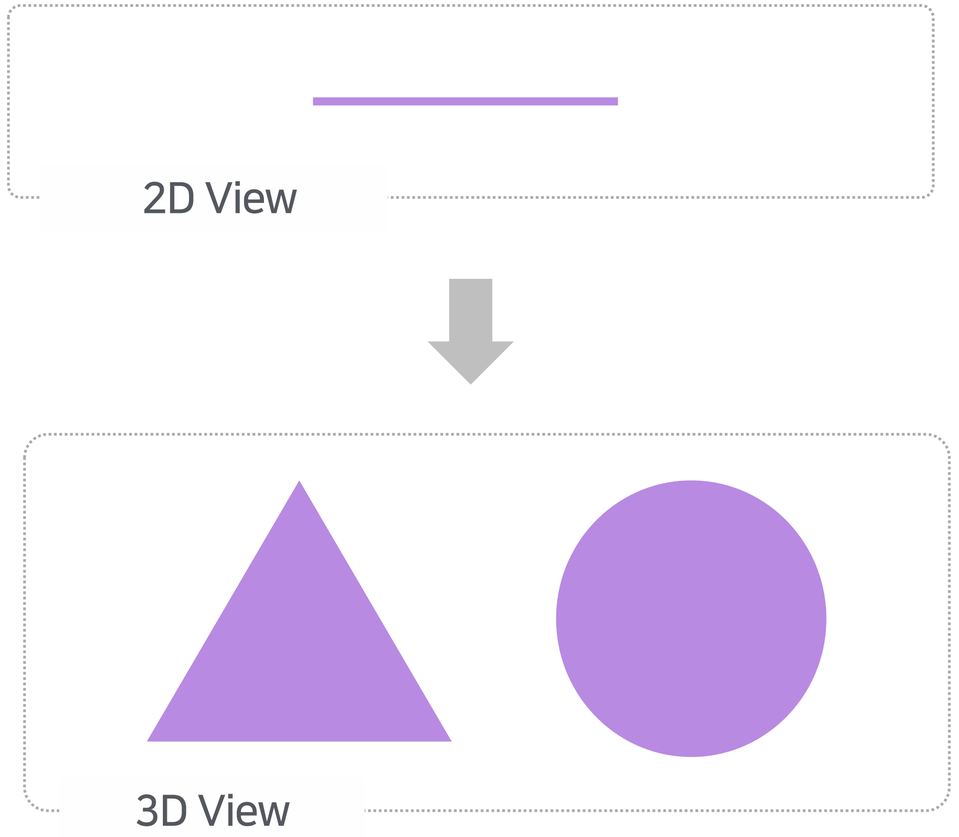


3.4 State Representation

DEVIEW 2020

왜 중요한가?



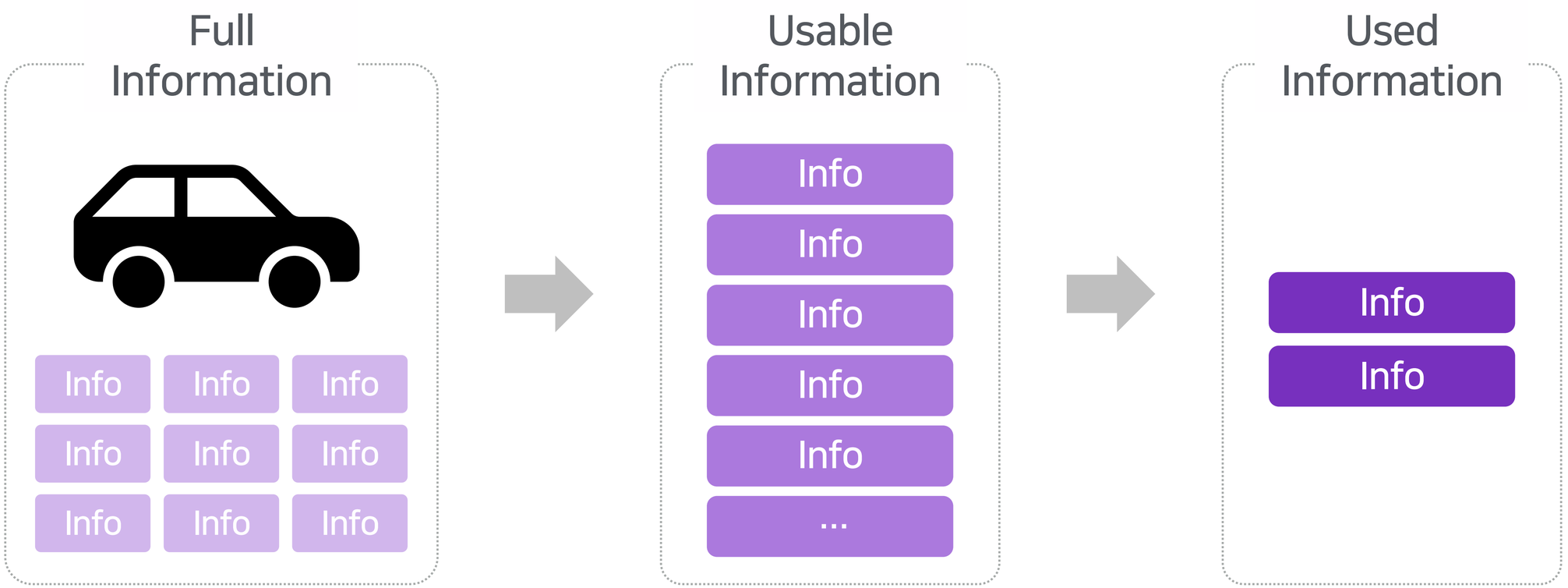


3.4 State Representation

DEVIEW 2020

왜 어려운가?

- Vertical Information

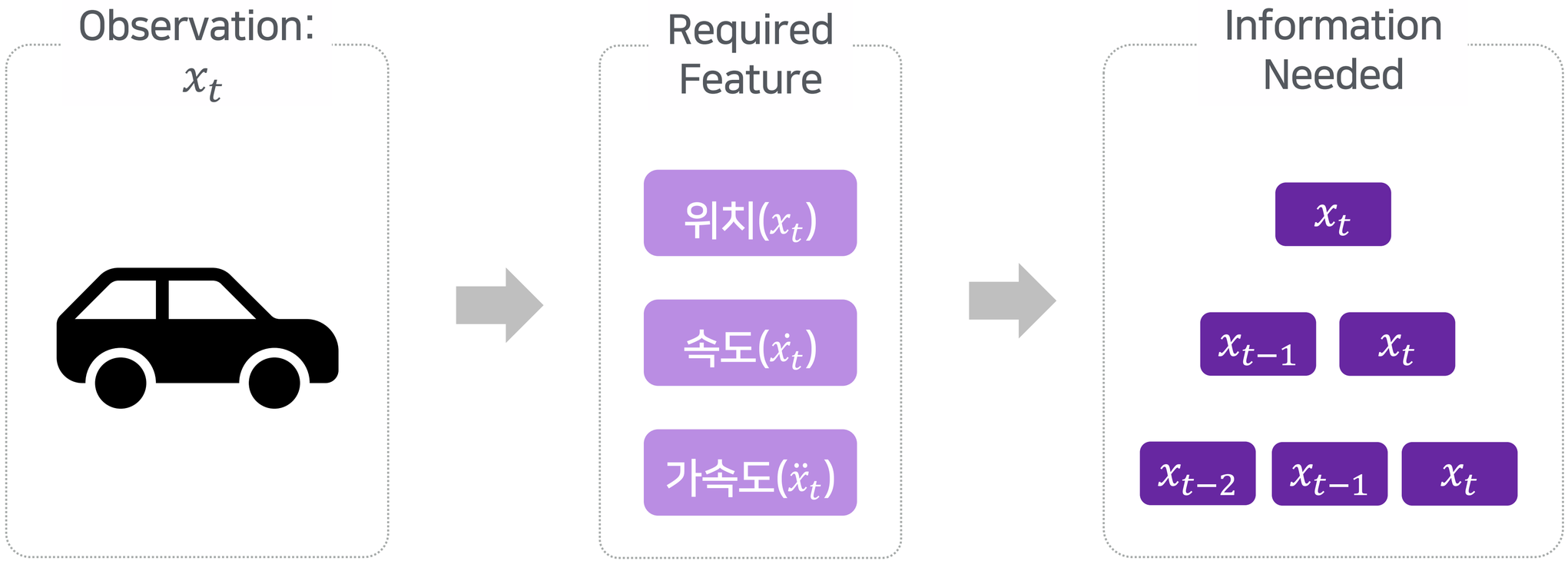


3.4 State Representation

DEVIEW 2020

왜 어려운가?

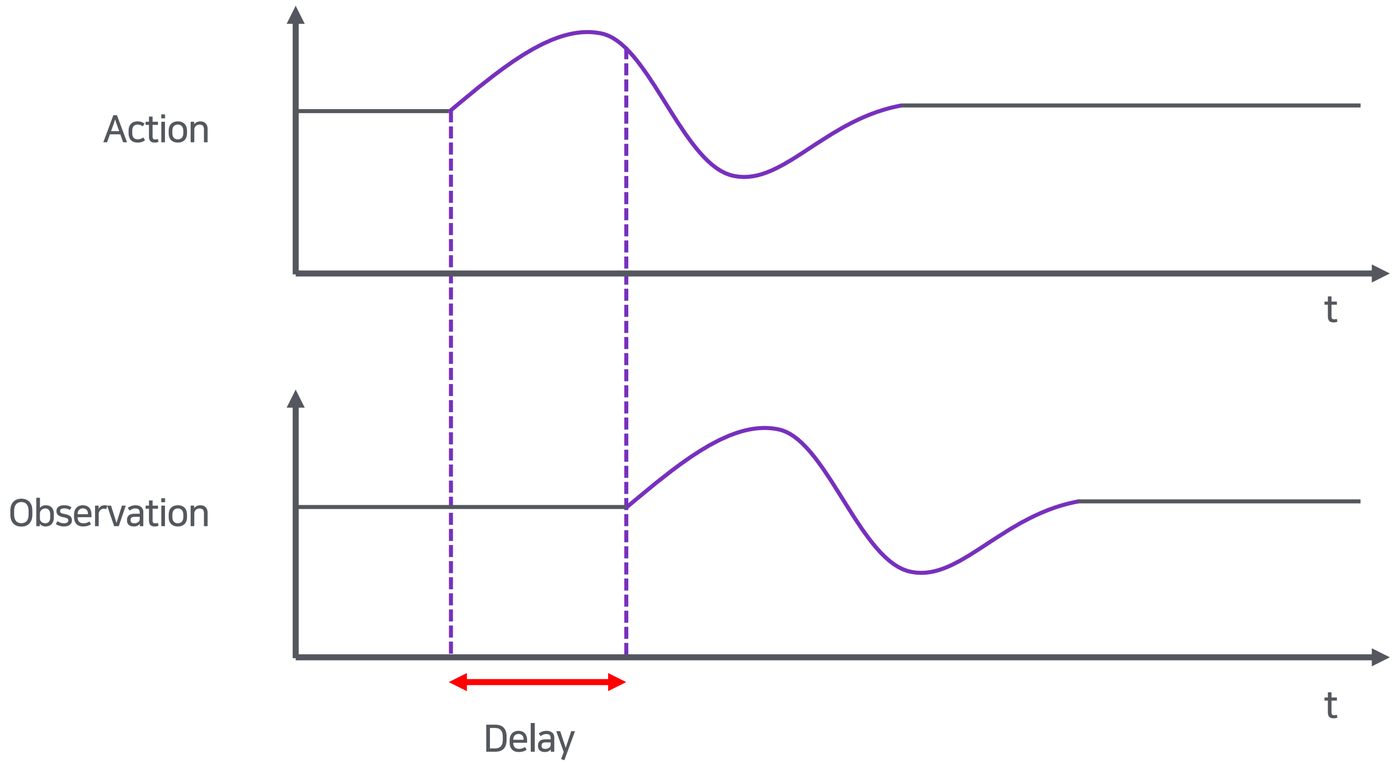
- Horizontal Information



3.4 State Representation

왜 어려운가?

- System Delay

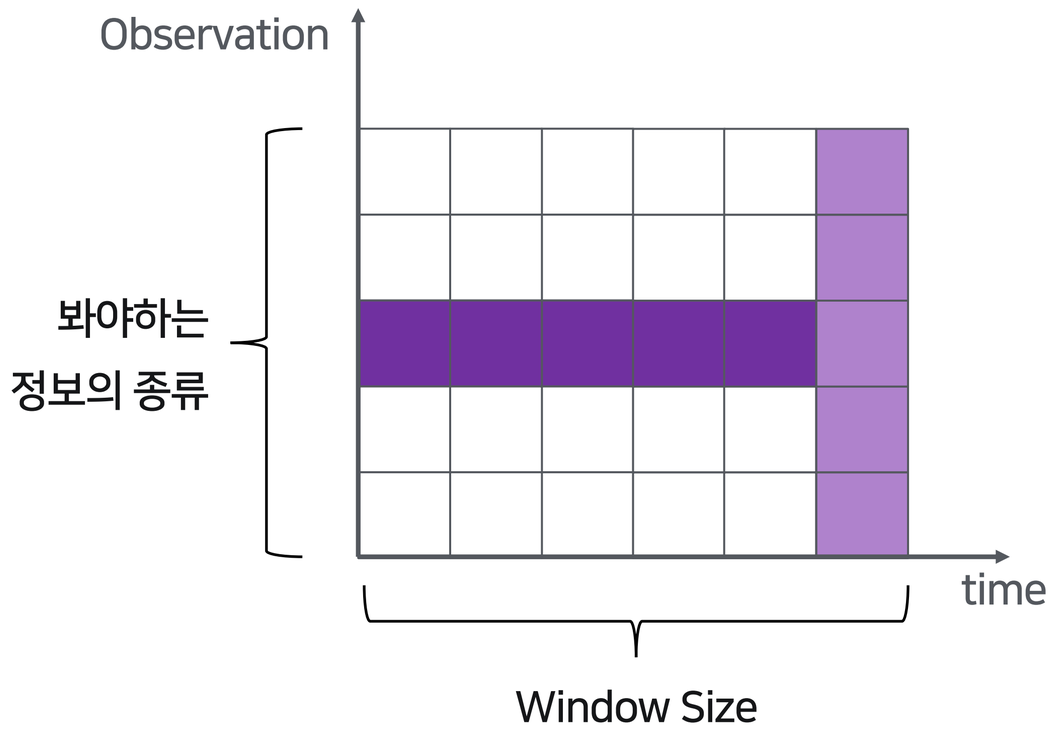


3.4 State Representation

기존 방식의 한계

도메인 전문가 의견을 반영한 주요 Feature 선정

- 사용되지 못하는 Feature는?
- 시스템 Delay 등을 고려한 적절한 Window Size는?



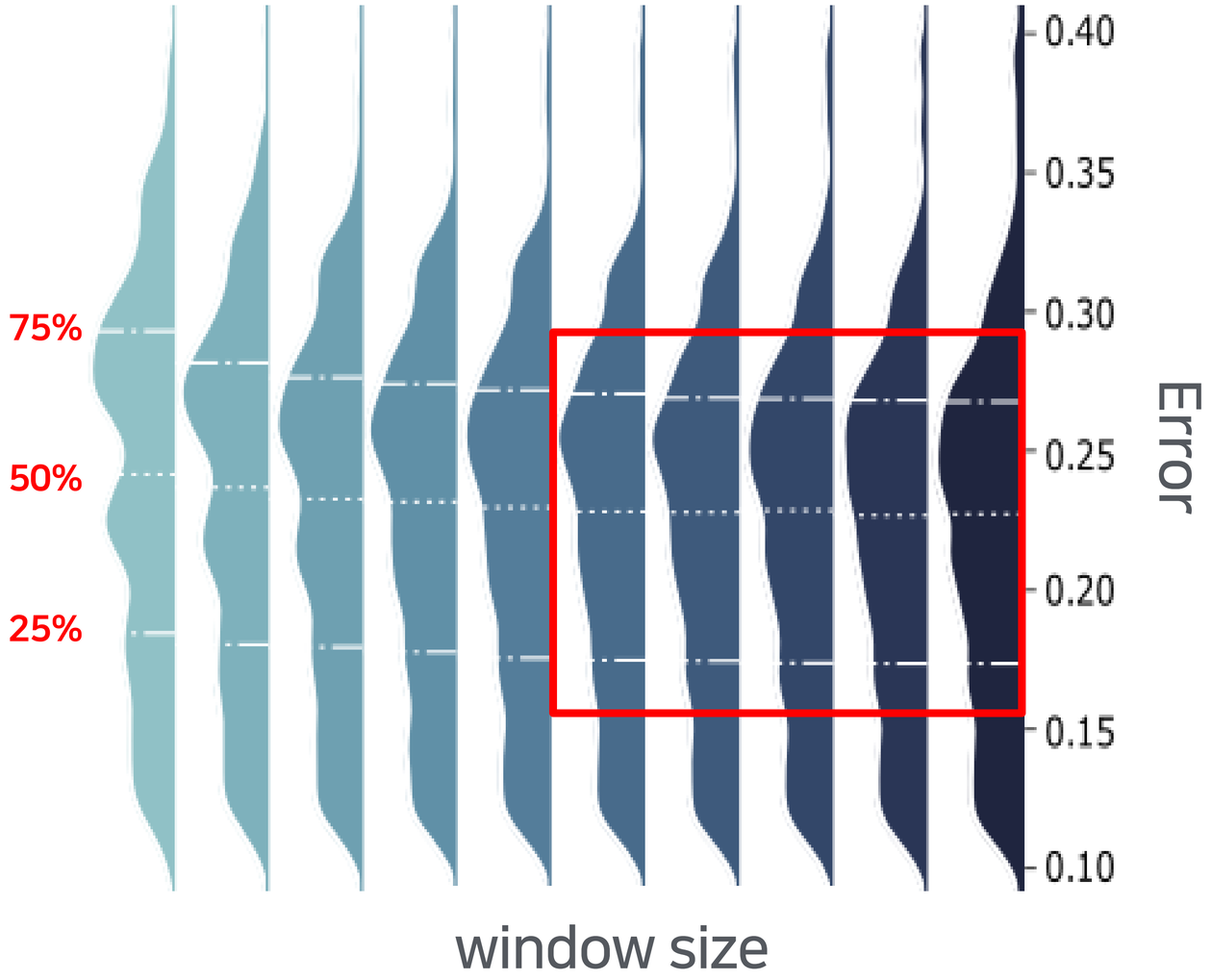
3.4 State Representation

DEVIEW 2020

Data-driven Analysis

1) Finding Window Size

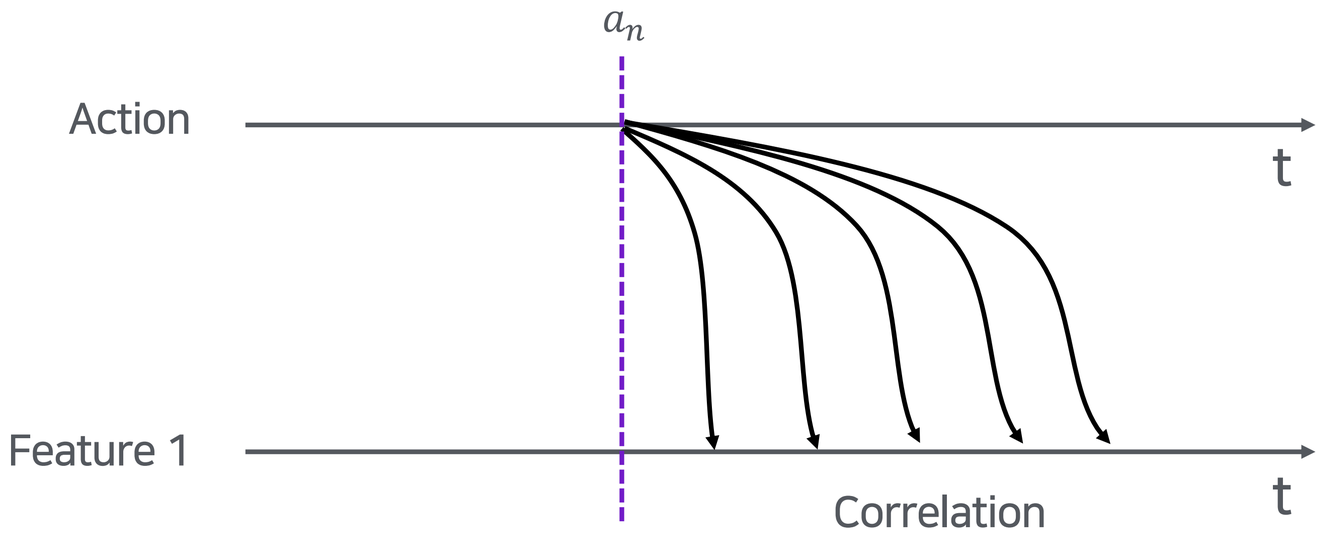
- Regression
- Input: observations of k timesteps
- Output: predictions of observation at n+1 timestep
- Ridge plot of MSE of predictions



3.4 State Representation

Data-driven Analysis

2) Finding delay time for action vs. observation



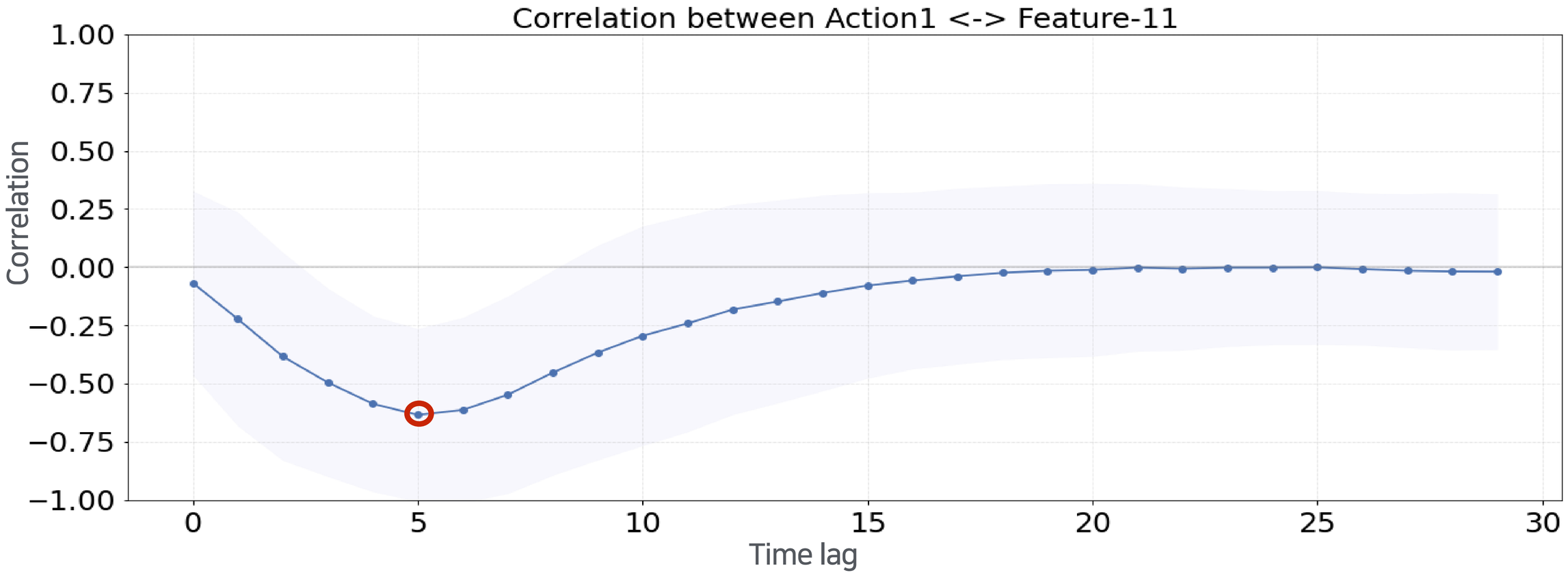
얼마나 Delay된 Feature가 현재 Action과 가장 Correlation이 높을까?

3.4 State Representation

DEVIEW 2020

Data-driven Analysis

2) Finding delay time for action vs. observation

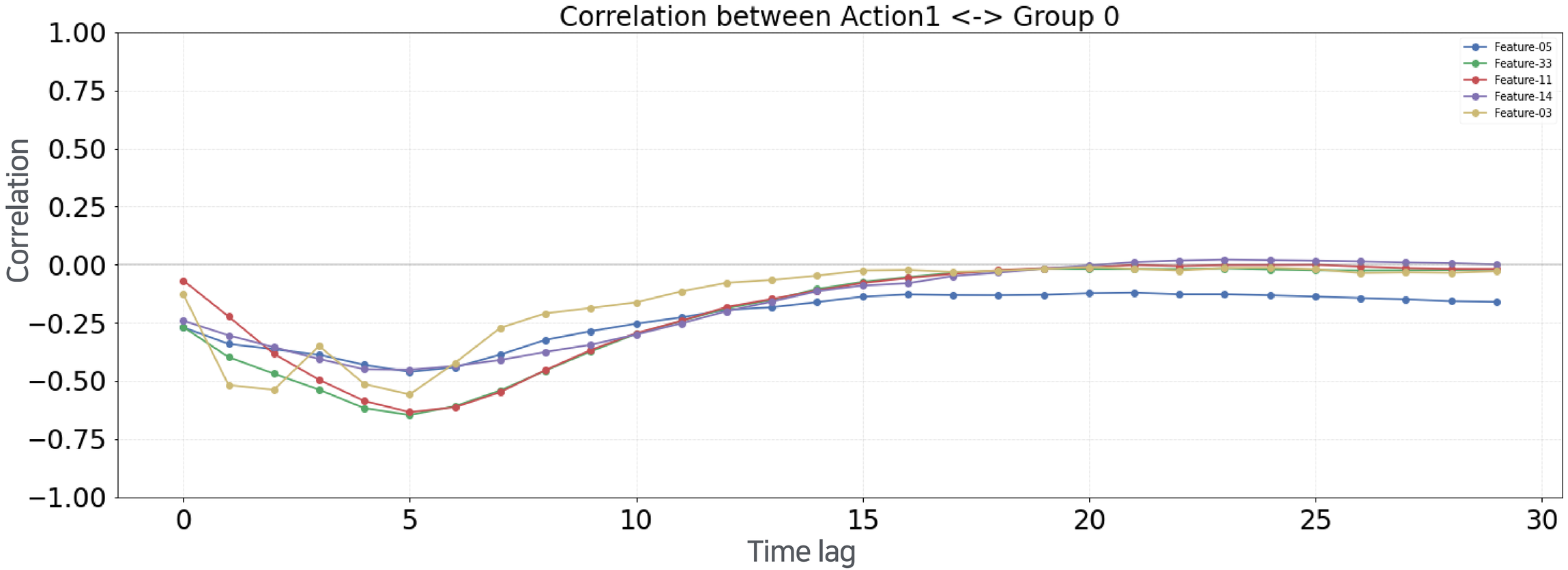


3.4 State Representation

DEVIEW 2020

Data-driven Analysis

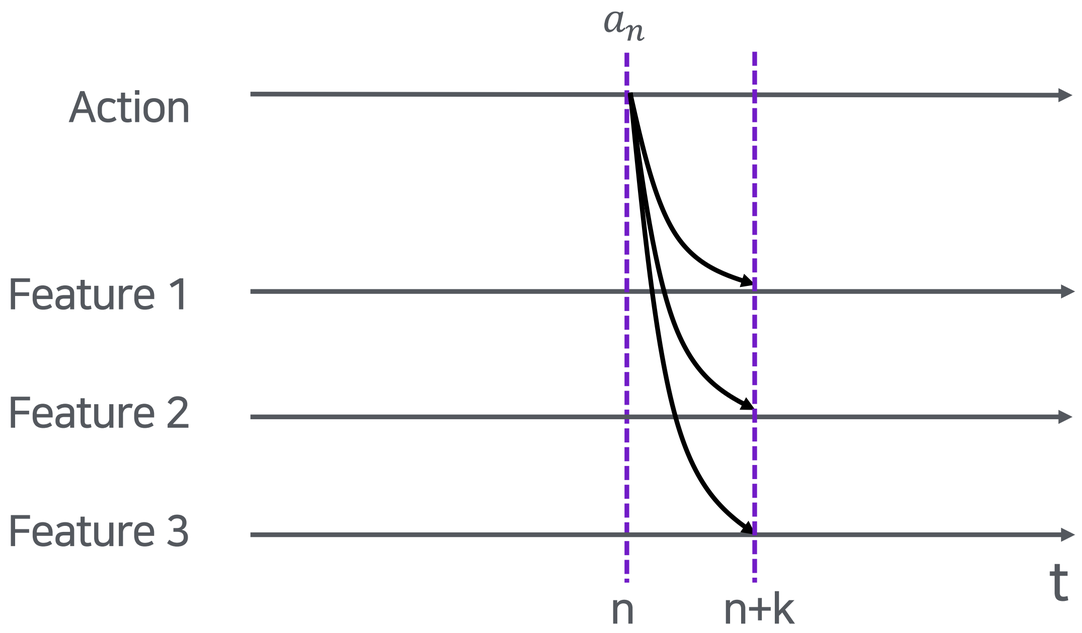
2) Finding delay time for action vs. observation



3.4 State Representation

Data-driven Analysis

2) Finding delay time for action vs. observation



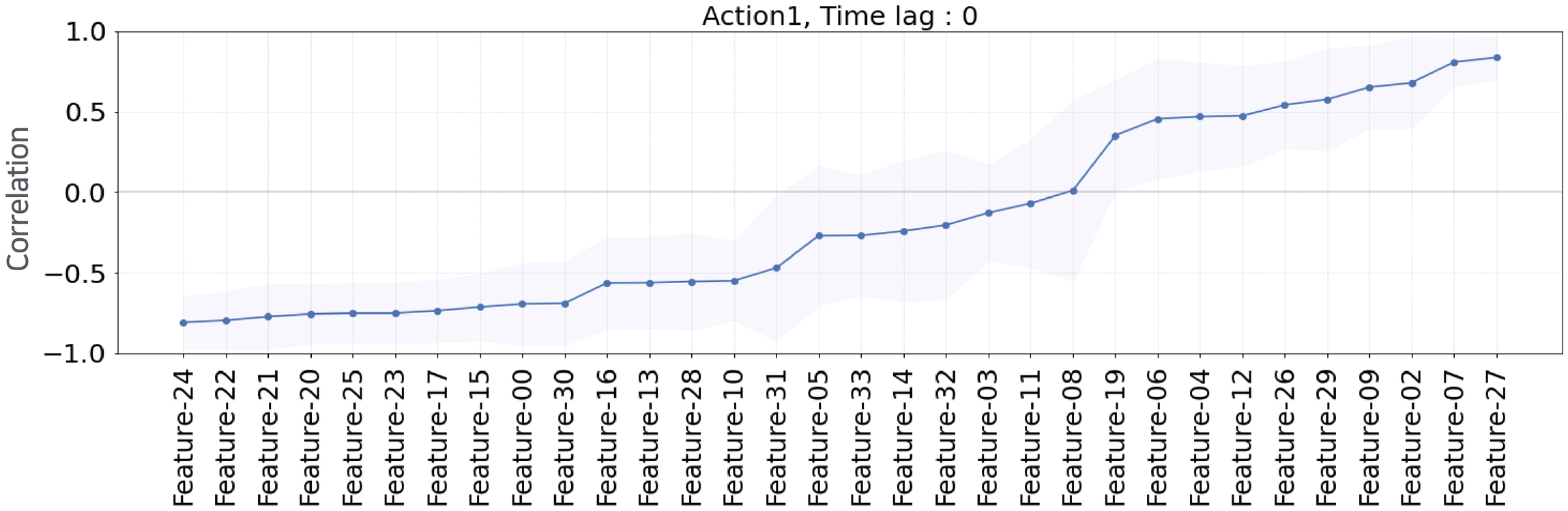
k만큼 Delay된 Feature들 중 어떤 Feature가 현재 Action과 가장 Correlation이 높을까?

3.4 State Representation

DEVIEW 2020

Data-driven Analysis

2) Finding delay time for action vs. observation

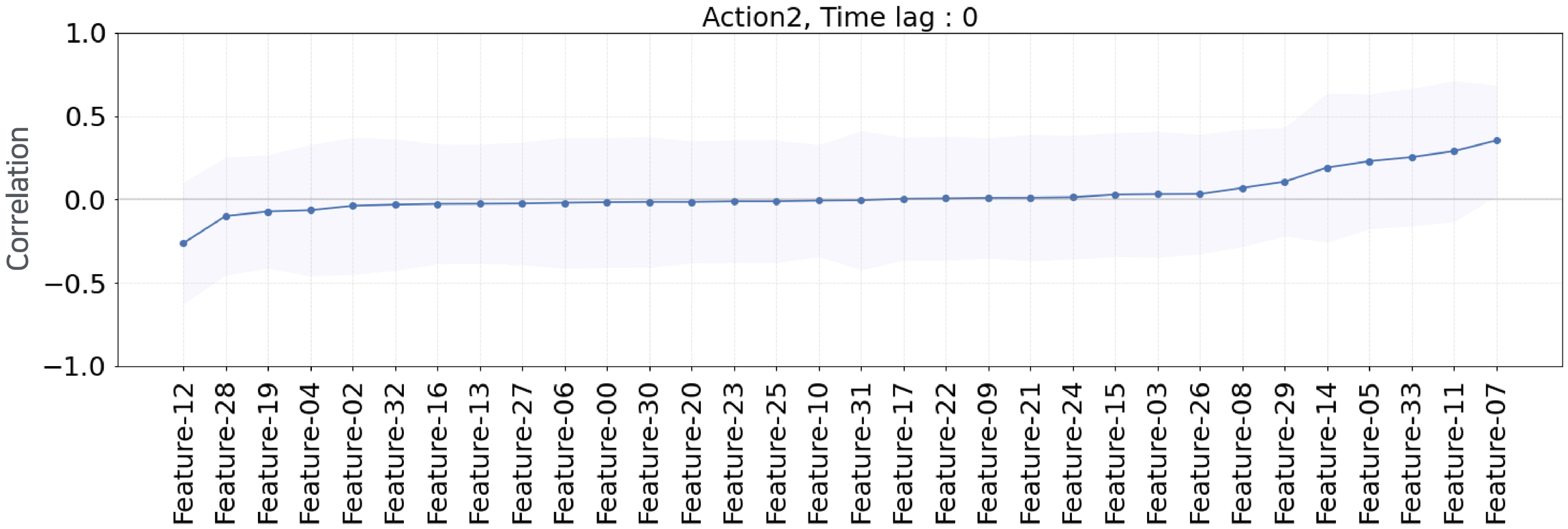


3.4 State Representation

DEVIEW 2020

Data-driven Analysis

2) Finding delay time for action vs. observation

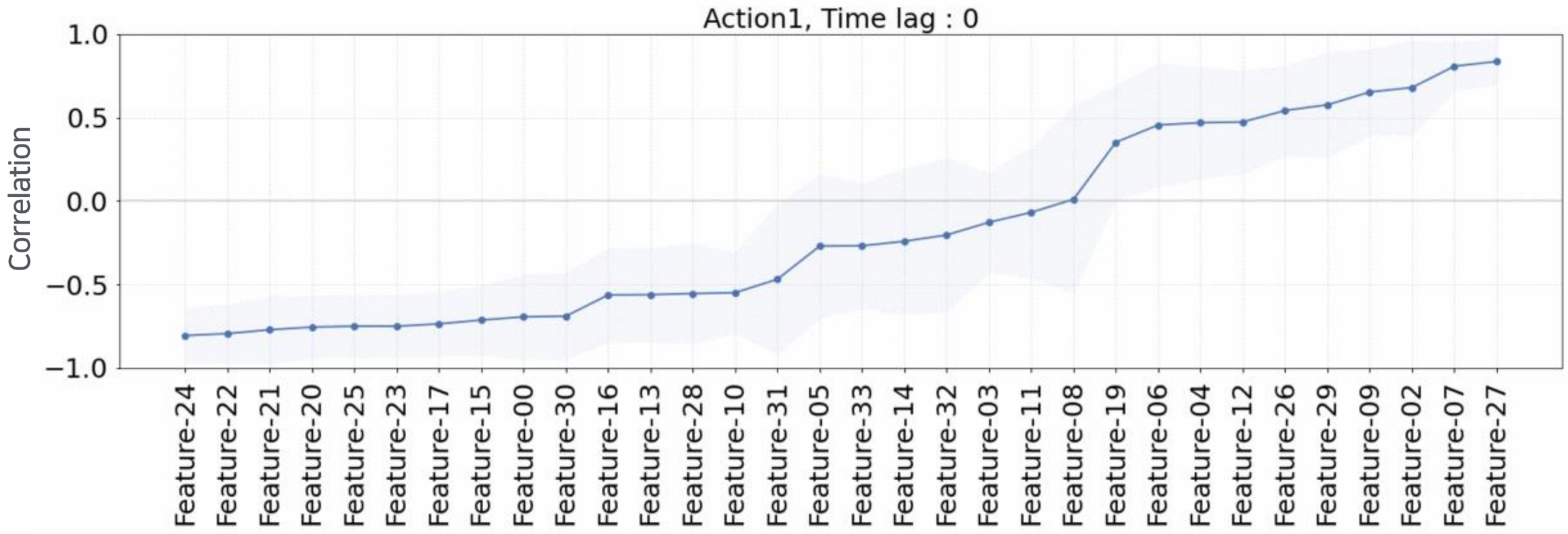


3.4 State Representation

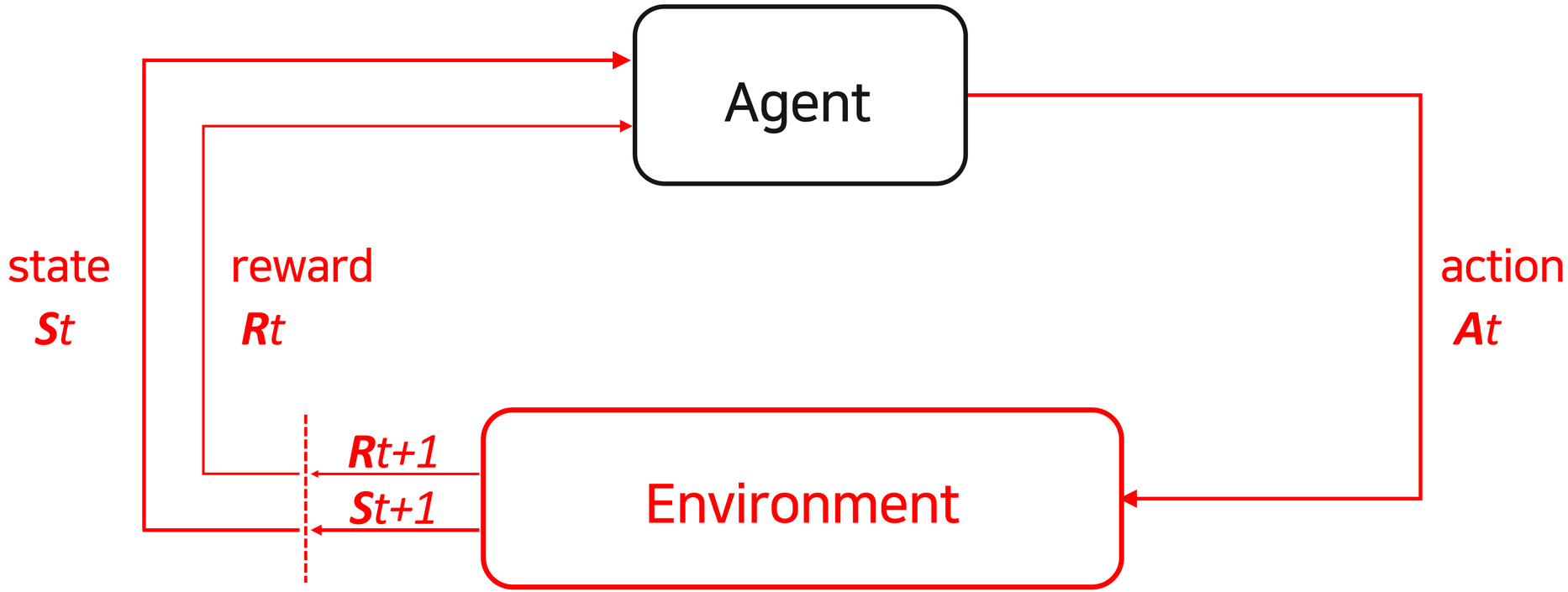
DEVIEW 2020

Data-driven Analysis

2) Finding delay time for action vs. observation



3.5 Summary



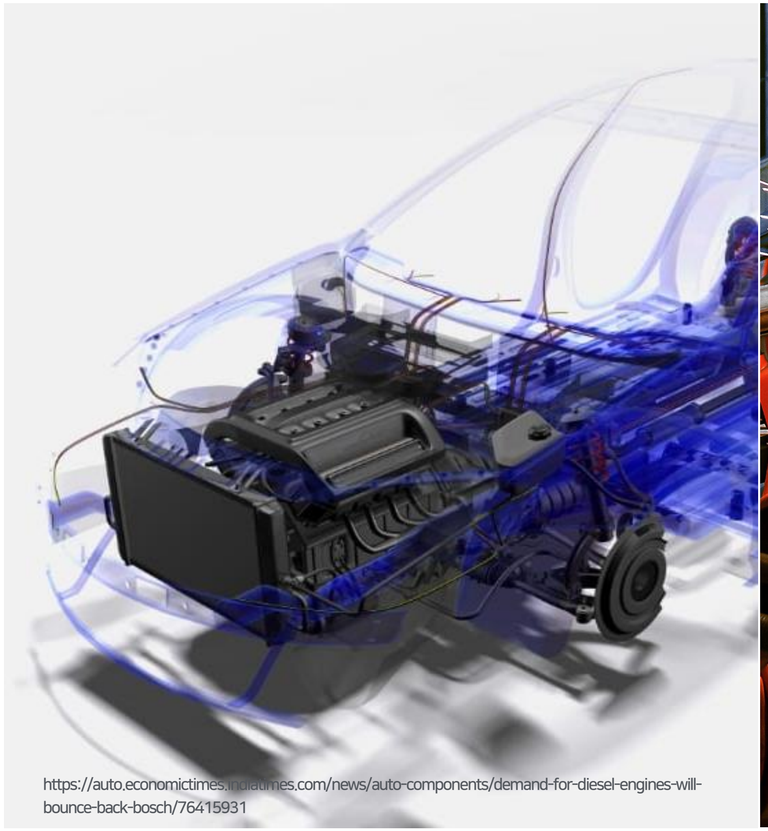
DEVIEW 2020

4. What’s next?

4.1 강화학습 프로젝트 현황

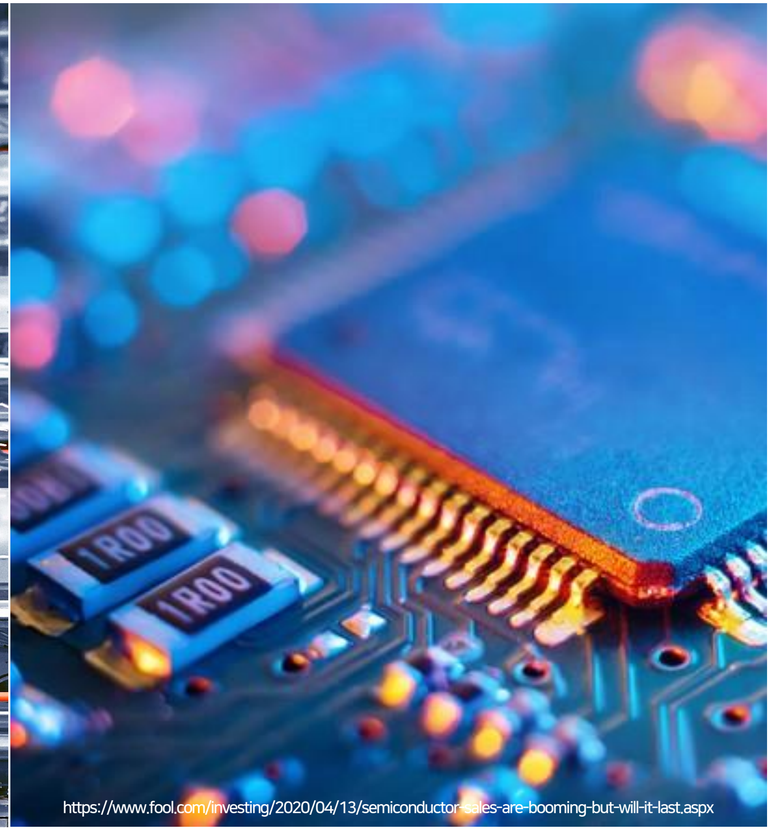
DEVIEW 2020

EMS, 로봇팔, AI Chip



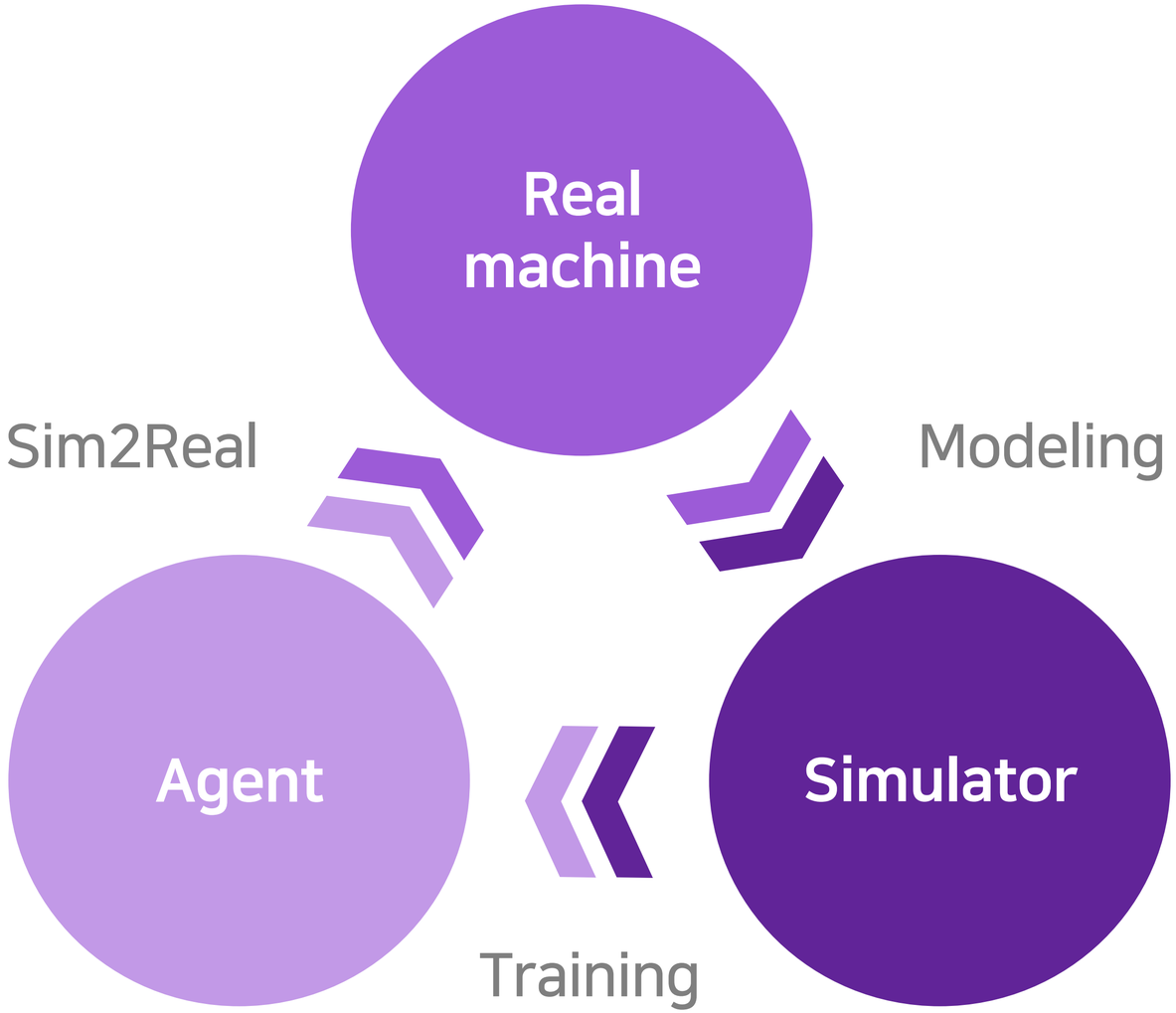
https://auto.economictimes.indiatimes.com/news/auto-components/demand-for-diesel-engines-will-bounce-back-bosch/76415931

https://www.ice.it/en/sites/default/files/2020-08/automotive-2020.pdf



https://www.fool.com/investing/2020/04/13/semiconductor-sales-are-booming-but-will-it-last.aspx

4.2 What's next?



Thank You

Q & A

References

References

Bellemare, M., Naddaf, Y., Veness, J., Bowling, M. (2013). The arcade learning environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research, 47,* 253-279.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., Rassa, Y. (2018). Safe exploration in continuous action paces. *arXivLabs*. arXiv:1801.08757v1 [cs.AI]

Mott, B., Anthony, St., & The Stella Team. (1990). *Stella*. Retrieved October 13, 2020, from https://stella-emu.github.io/

Appendix

Appendix A. Glossary

Curiosity	Technique which uses an intrinsic reward function that returns higher values when the agent explore new areas of the environment.
DOF	Degrees of Freedom
EMS	Energy Management System
GAIL	Generative Adversarial Imitation Learning
HIRO	HIerarchical Reinforcement learning with Off-policy correction.
MDP	Markov Decision Process
ML-Agents	Open-source library that enables games and simulations on Unity3D to serve as environments for training intelligent agents.
MoveIt	Open source software for robot manipulation. Can perform motion planning, manipulation, collision avoidance, control, inverse kinematics and 3D perception.
MSE	Minimum Square Error
POMDP	Partially Observable Markov Decision Process
POV	Point of View
PPO	Proximal Policy Optimization. On-Policy reinforcement learning algorithm.
Ray	Universal API for building distributed applications
RLocks	MakinaRocks Reinforcement Learning framework.
ROS	Robot Operating System is an open-source meta-operating system for robots. It contains a set of tools, software and hardware abstractions such as low-level drivers, inter-process communication, package management. etc.
ROS#	Set of open source software libraries and tools in C# for communicating with ROS from .NET applications.
SAC	Soft Actor-Critic. Off-policy reinforcement learning algorithm
Sim2Real	Area of research that aims to transfer the knowledge learned on simulation to real world systems.
Unity3D	Cross-platform game engine developed by Unity Technologies
UR3	Compact table-top robot produced by Universal Robots

Appendix B. Simulator 구성

DEVIEW 2020

구성 요소	
3D Environment	▪ Unity3D (game engine) ▪ ML-Agents (Unity 3D < - > RL Interface) ▪ ROS# (Unity 3D < - > ROS Interface) ▪ 3D Game Scene (Robot Arm, goals, obstacles)
Reinforcement Learning	▪ Rlocks (MakinaRocks reinforcement learning framework) ▪ Agent, runner, utils, algorithms (HER, GAIL, etc.)
ROS	▪ ROS (Robot Operating System) ▪ Motion Planning ▪ Sim2Real