KOREA UNIVERSITY

# Reinforcement Learning and Applications

Seung Hyun Oh

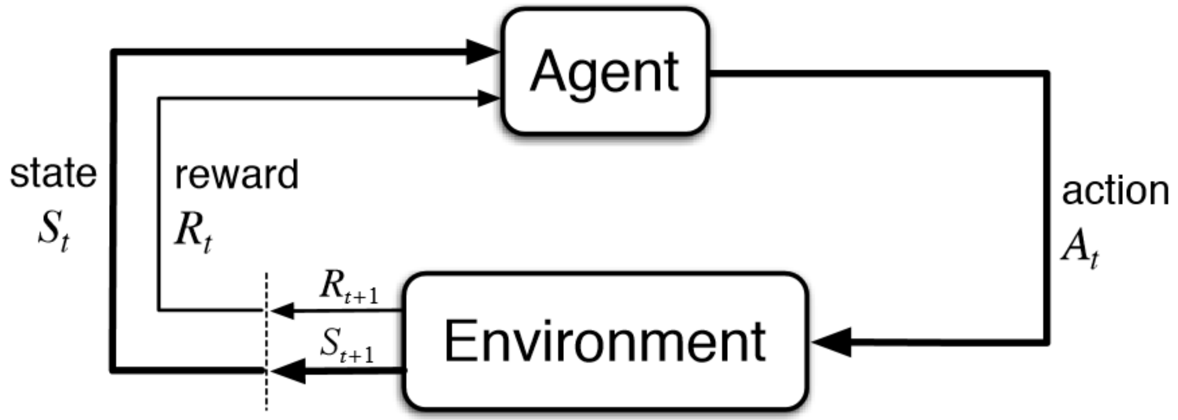
20240305

Lab for Informatics, communications and system

# Outlines

- What is Reinforcement Learning?
- Q-Learning
- DQN/DDPG/PPO
- Recent RL Algorithms(ex : MOPPO, TD3 … )
- Applications (ex : Vessel) // Vessel Dynamics (Action, State), 적용되는지

# Reinforcement Learning

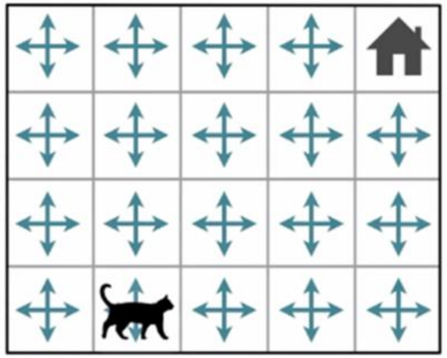



- Find optimal action sequence for expected reward maximization

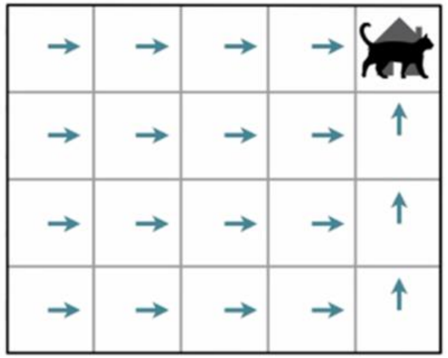
- The agent starts
in a given state within its environment $s_0 \in S$
by gathering an initial observation $\omega_0 \in \Omega$

- At each time step $t$,
The agent has to take an action $a_t \in A$

# Q-learning



Behavior policy



Target policy

평가하고 업데이트하고자 하는 policy

실제로 행동해서 다음 state를 얻는 것

Q-value

$$Q(s,a) \underset{\text{Update}}{\leftarrow} (1-\alpha)Q(s_t,a_t) + \underset{\text{Damping}}{\alpha}(R_t + \sigma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}))$$

$$= \int_{s_{t+1}a_{t+1}} (R_t + \sigma Q(s_{t+1},a_{t+1}))\underbrace{p(a_{t+1}|s_{t+1})}_{\text{Target Policy}}p(\underbrace{s_{t+1}}_{\text{R.V}}|s_t,a_t)\, ds_{t+1}, a_{t+1}$$

$$\fallingdotseq Q_N = (1-\alpha)Q_{N-1} + \alpha(R_t^N + \sigma Q(s_{t+1}^n, a_{t+1}^n))$$

# DQN (Deep Q-learning Network)

**Classic Q-learning**

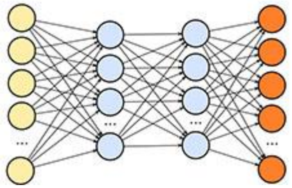
State $s_t$ →
Action $a_t$ →

| | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| $s_1$ | $Q(s_1, a_1)$ | $Q(s_1, a_2)$ | $Q(s_1, a_3)$ |
| $s_2$ | $Q(s_2, a_1)$ | $Q(s_2, a_2)$ | $Q(s_2, a_3)$ |
| $s_3$ | $Q(s_3, a_1)$ | $Q(s_3, a_2)$ | $Q(s_3, a_3)$ |

→ Q-value $Q(s_t, a_t)$

**Deep Q-learning**

State $s_t$ →
Action $a_t$ →



→ Q-value $Q(s_t, a_t)$

**Algorithm 1** Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
**for** episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  **for** $t = 1, T$ **do**
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  **end for**
**end for**

- Q-function

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q^*(s', a') \middle| s, a \right]$$

탐험율 감가율

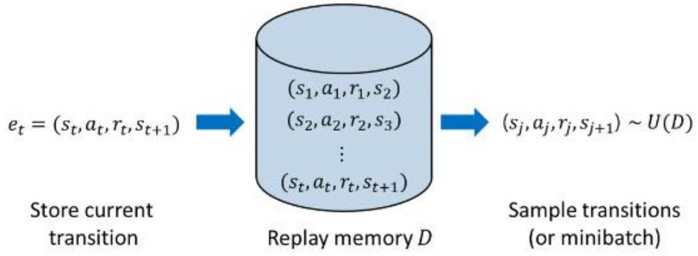
- Loss Function & Update (SGD)

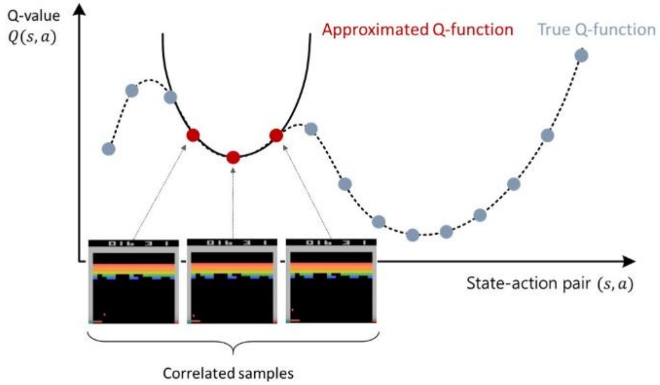
$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[ \left( y_i - Q(s, a; \theta_i) \right)^2 \right]$$

가중치 미래 보상 추정치 예측값

$$\boxed{y_i = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a]}$$

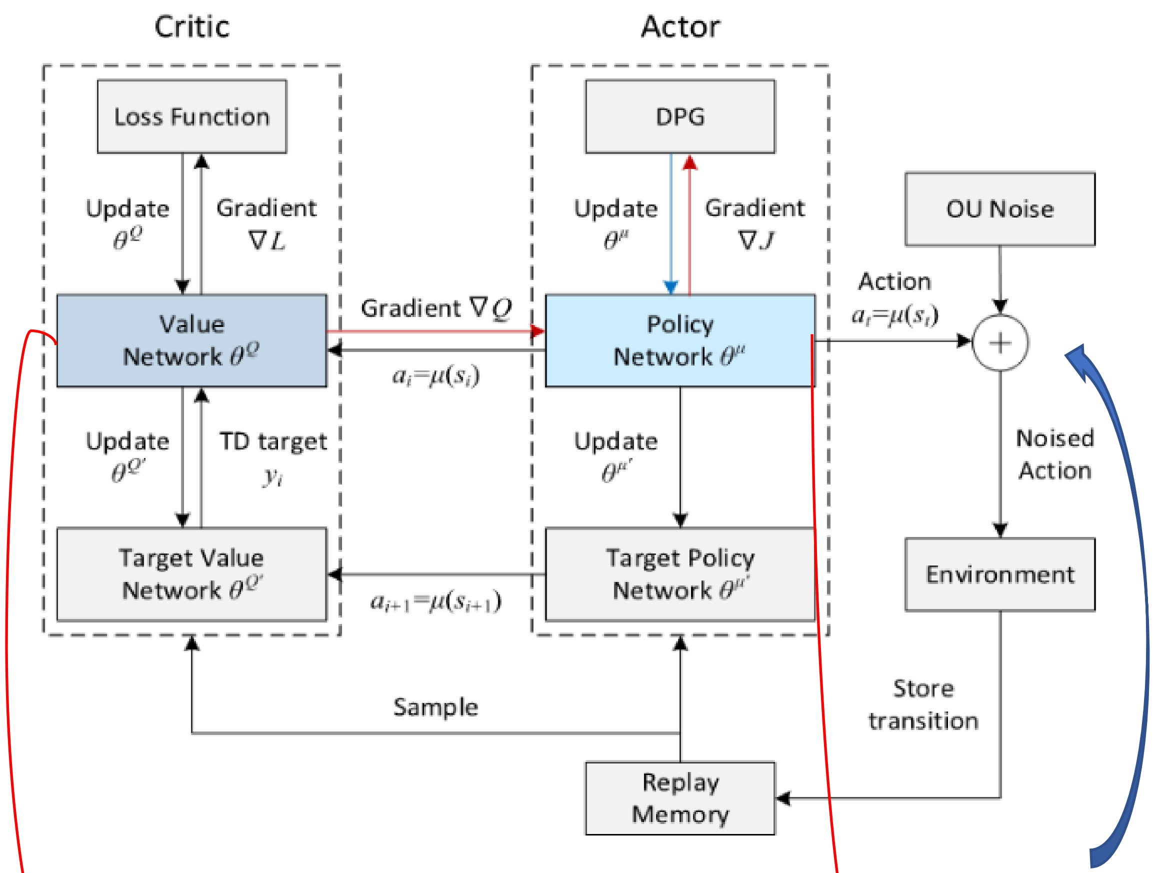
$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right].$$



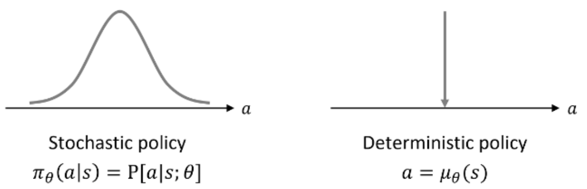




[1] Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). https://arxiv.org/abs/1509.02971

# DDPG(Deep Deterministic Policy Gradient)




- Q-function




$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1} \sim \pi}\left[Q^\pi(s_{t+1}, a_{t+1})\right]\right]$$

$$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))\right]$$

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right]$$

Loss Function

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right]$$
$$= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t}\right]$$

Policy Gradient

[2] Continuous control with deep reinforcement learning Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra https://arxiv.org/abs/1509.02971

# DDPG(Deep Deterministic Policy Gradient)

**Algorithm 1** Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
**for** episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  **for** $t = 1, T$ **do**
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  **end for**
**end for**

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
**for** episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  **for** t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

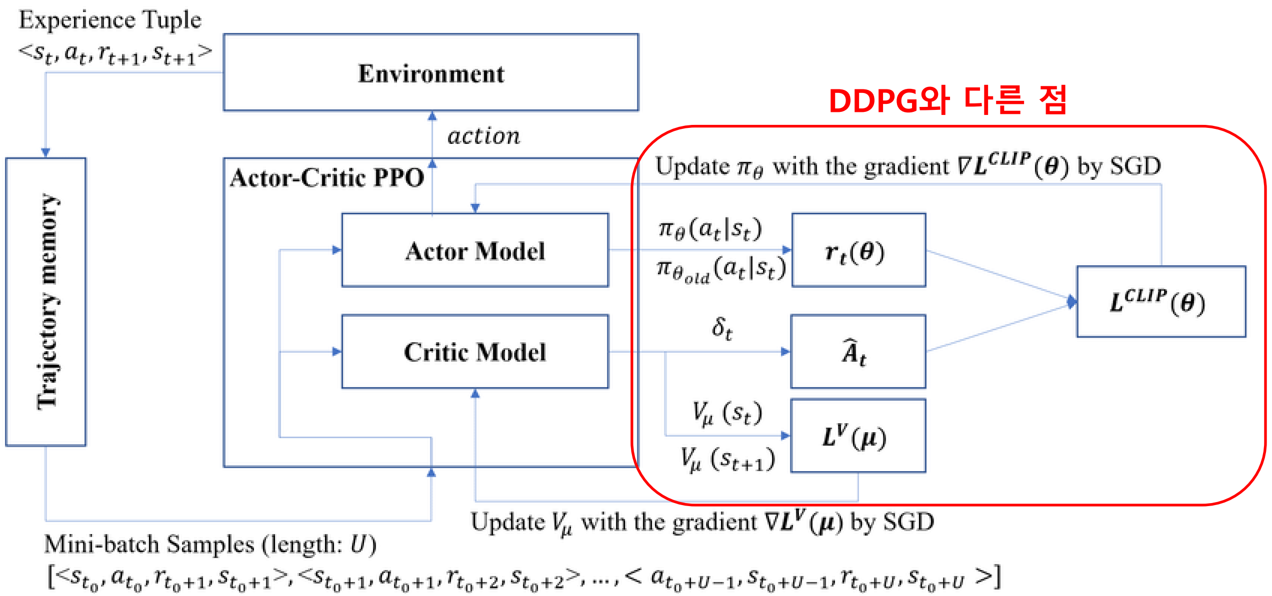
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  **end for**
**end for**

- Discrete -> Continuous action
- Simple CNN structure -> Actor & Critic

[2] Continuous control with deep reinforcement learning Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra https://arxiv.org/abs/1509.02971

# PPO (Proximal Policy Optimization)




$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

전후 state 확률 비율

$$= L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$
$$\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

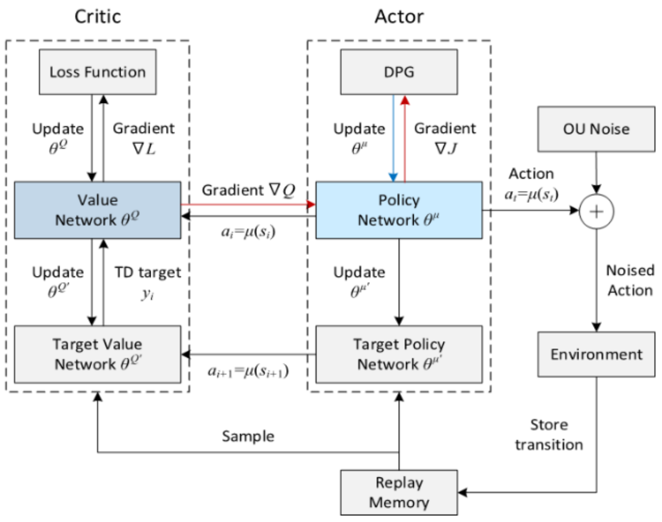
**Algorithm 1** PPO, Actor-Critic Style

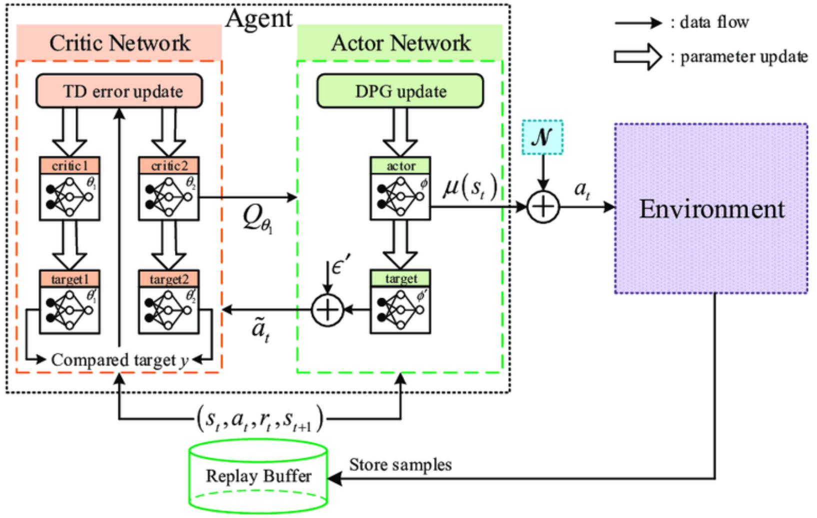
**for** iteration$=1, 2, \dots$ **do**
  **for** actor$=1, 2, \dots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ timesteps
    Compute advantage estimates $\hat{A}_1, \dots, \hat{A}_T$
  **end for**
  Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \le NT$
  $\theta_{\text{old}} \leftarrow \theta$
**end for**

[3] Proximal Policy Optimization Algorithms John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov https://doi.org/10.48550/arXiv.1707.06347

# Recent RL Algorithms

- <DDPG>에서 <TD3>로 발전 (4)




- PPO 변형 알고리즘

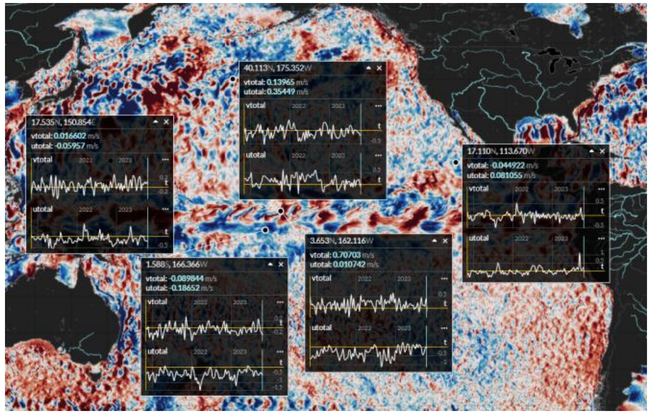
-> Adaptive PPO (학습 과정 중에 학습률 변경)

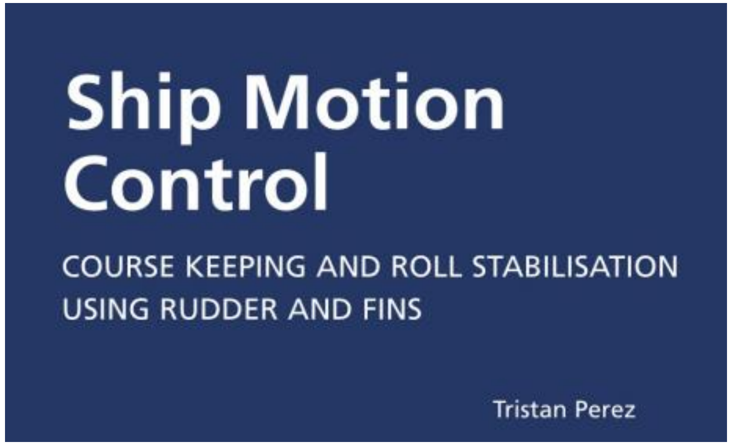
-> MoPPO (Modifed Actor-Critics) (5)


[4] Modified Actor-Critics Erinc Merdivan, Sten Hanke, Matthieu Geist https://doi.org/10.48550/arXiv.1907.01298

[5] **Addressing Function Approximation Error in Actor-Critic Methods** Scott Fujimoto, Herke van Hoof, David Meger https://doi.org/10.48550/arXiv.1802.09477

# Applications





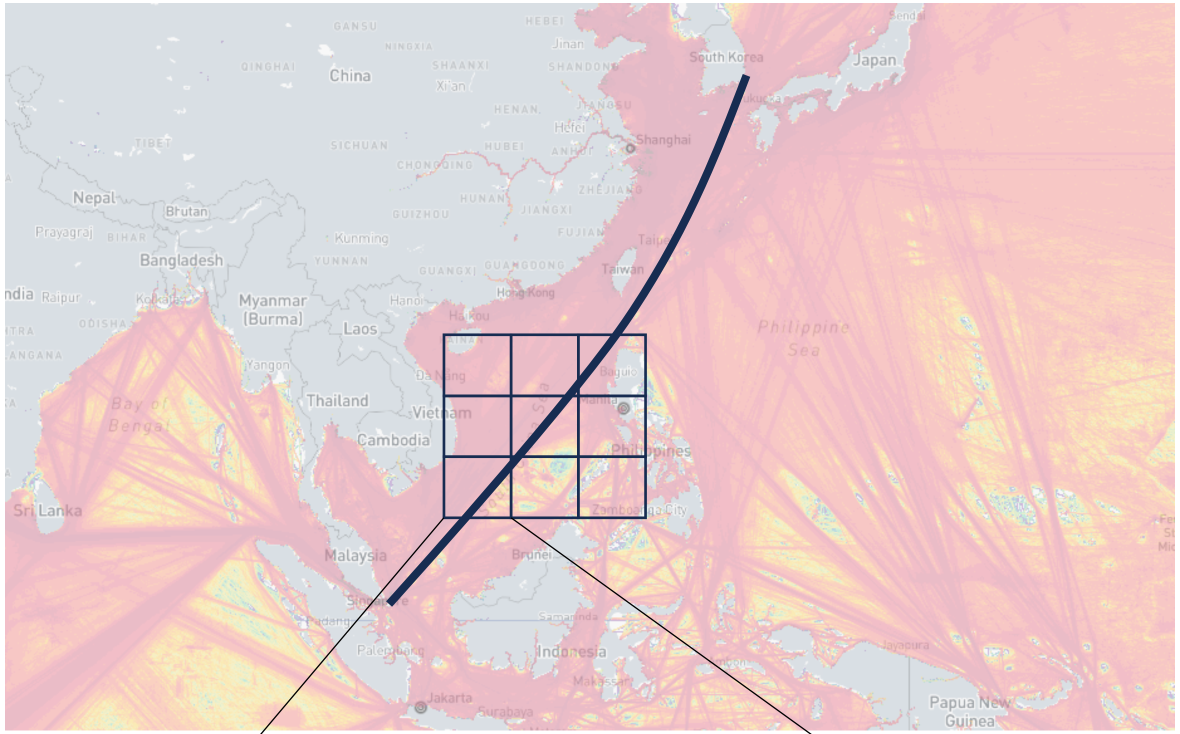

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V\cos\psi + C_x(x,y) \\ V\sin\psi + C_y(x,y) \\ r_c \end{bmatrix}$$

**Vessel Motion**
**(Action)**

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -m(x_g r + \nu) \\ 0 & 0 & mu \\ m(x_g r + \nu) & -mu & 0 \end{bmatrix}$$

**Drag, Interference Coeff**
**(State)**

Lab for Informatics, communications and system navigation



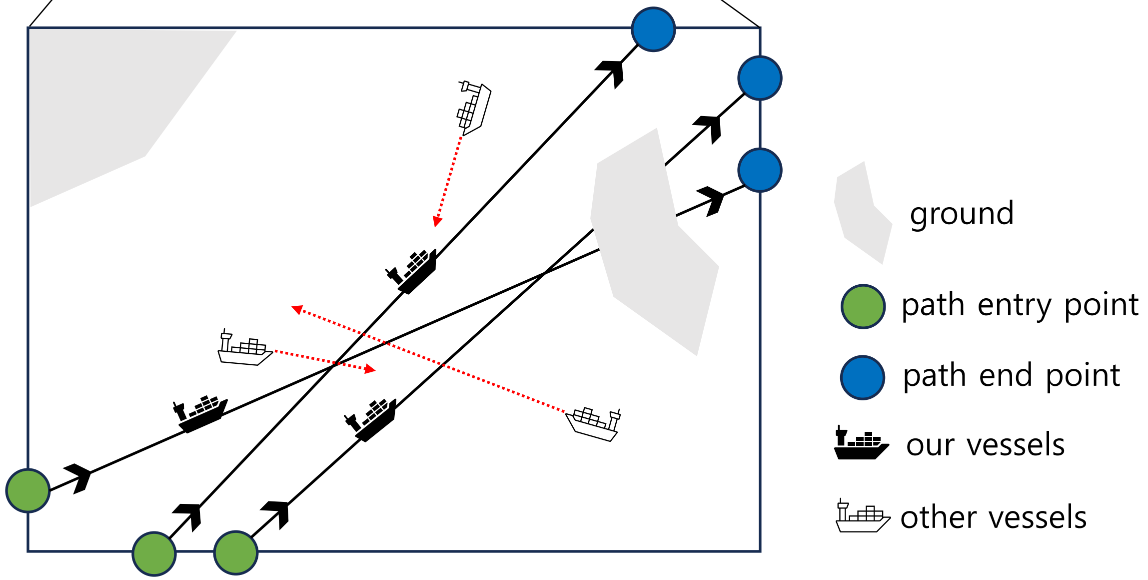

4606 km
(busan -> singapore)

6.1 days
(17knot)

31.484km/h
1knot= 1.852km/h

our system

path planning




15 ~ 20 km
each grid

0.5 ~ 1 hour

https://www.marinetraffic.com/en/ais/home/centerx:114.5/centery:20.3/zoom:4

# 감사합니다.