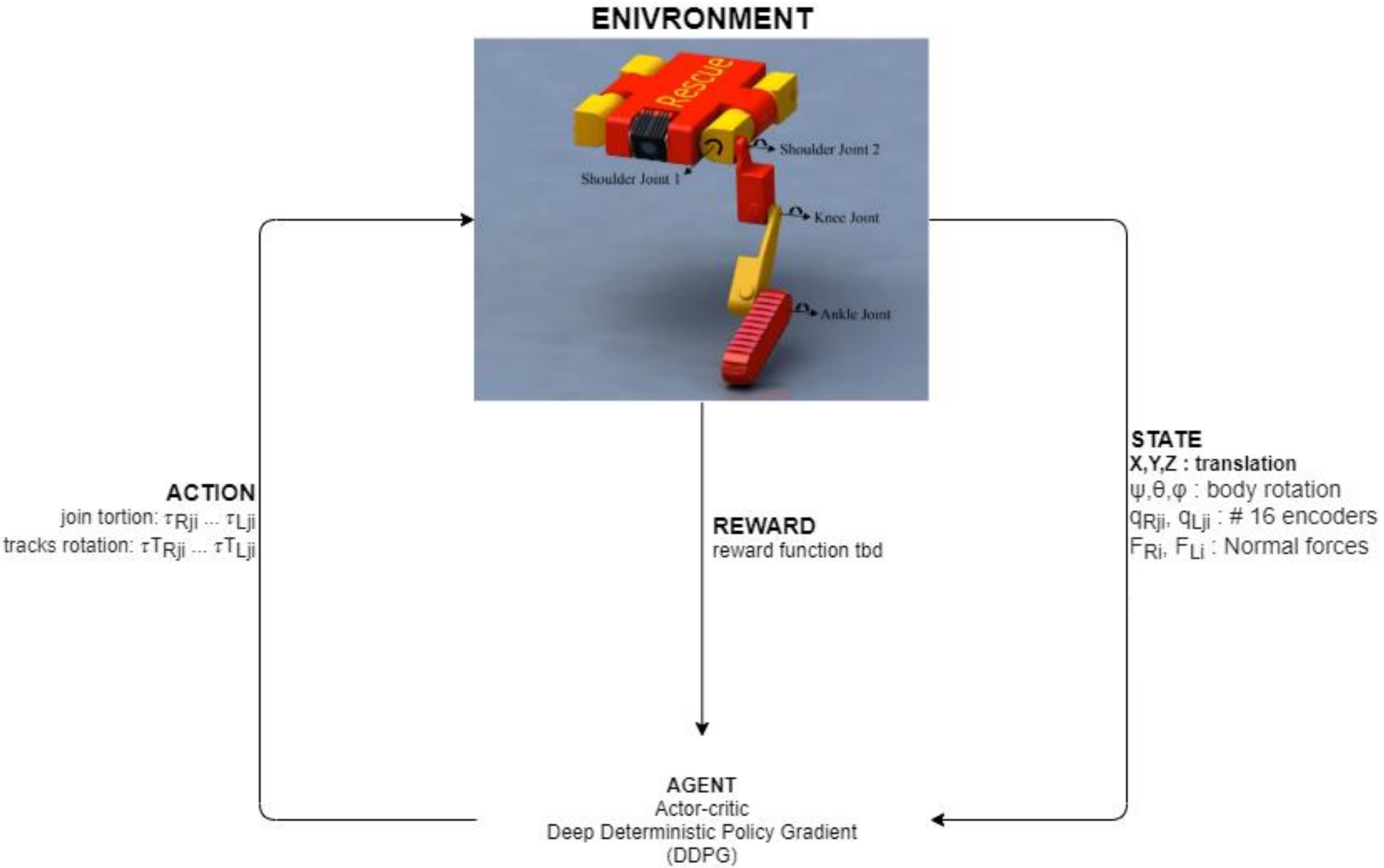
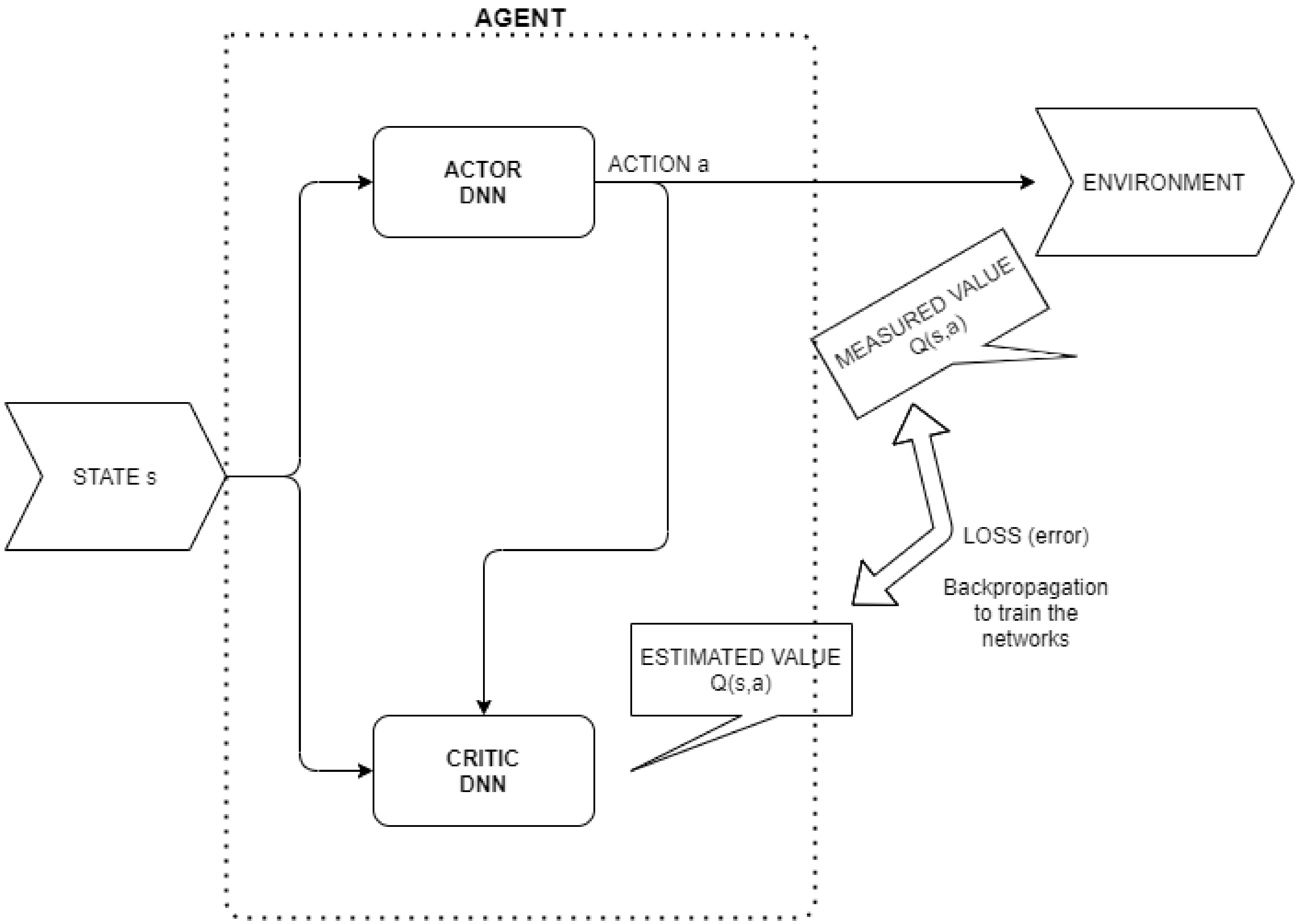
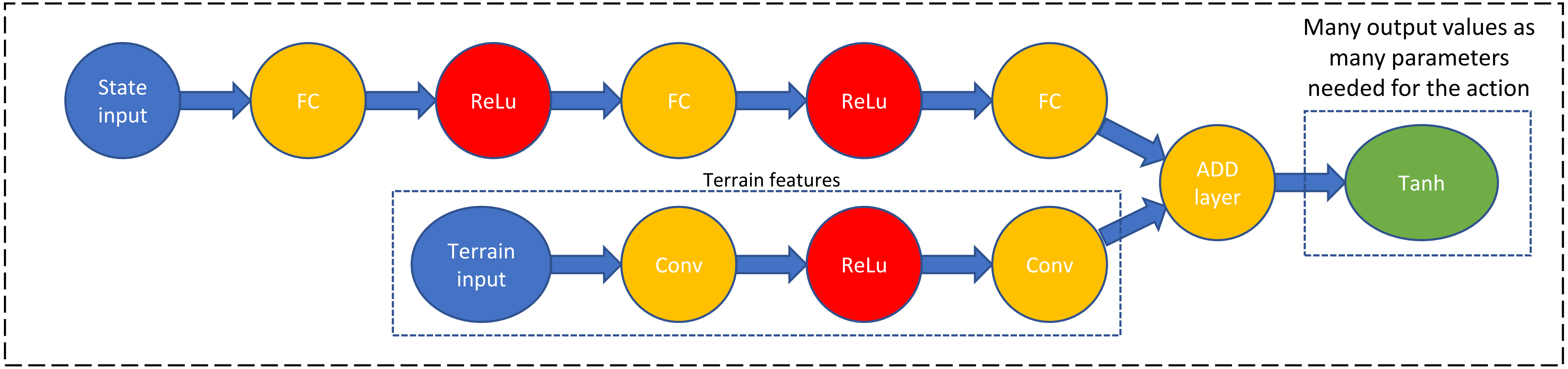
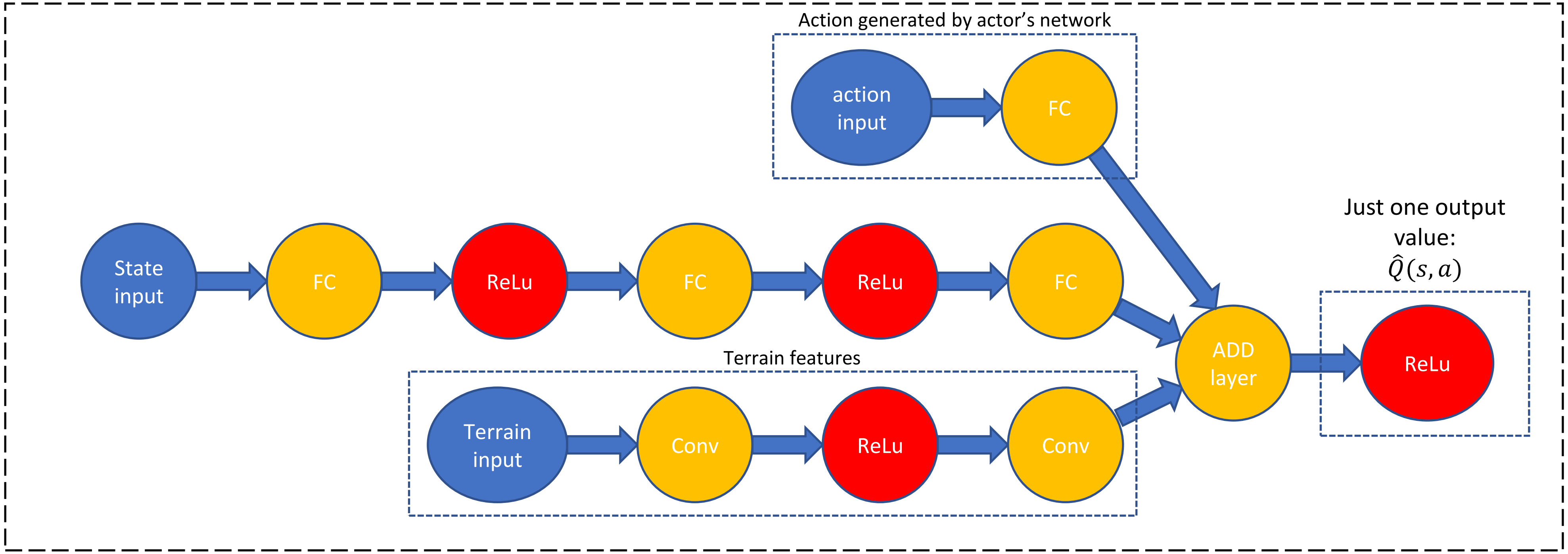

ENIVRONMENT
Rescue
Shoulder Joint 2
Shoulder Joint 1
Knee Joint
Ankle Joint
ACTION
join tortion: τ_{Rji} ... τ_{Lji}
tracks rotation: τT_{Rji} ... τT_{Lji}
REWARD
reward function tbd
STATE
X,Y,Z : translation
ψ,θ,φ : body rotation
q_{Rji}, q_{Lji} : # 16 encoders
F_{Ri}, F_{Li} : Normal forces
AGENT
Actor-critic
Deep Deterministic Policy Gradient
(DDPG)


AGENT
ACTOR
DNN
ACTION a
ENVIRONMENT
MEASURED VALUE
Q(s,a)
STATE s
LOSS (error)
Backpropagation
to train the
networks
ESTIMATED VALUE
Q(s,a)
CRITIC
DNN

Actor’s network



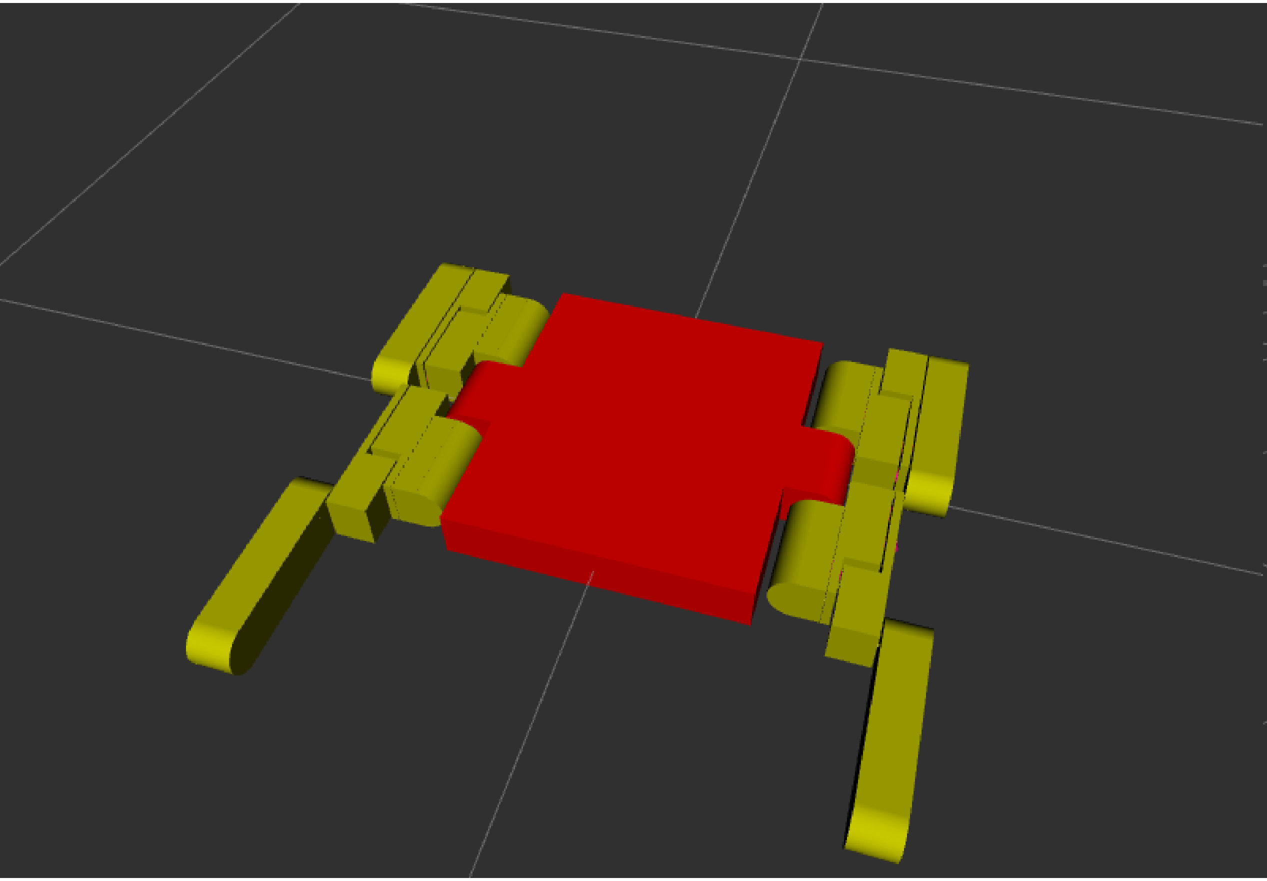
Critic’s network

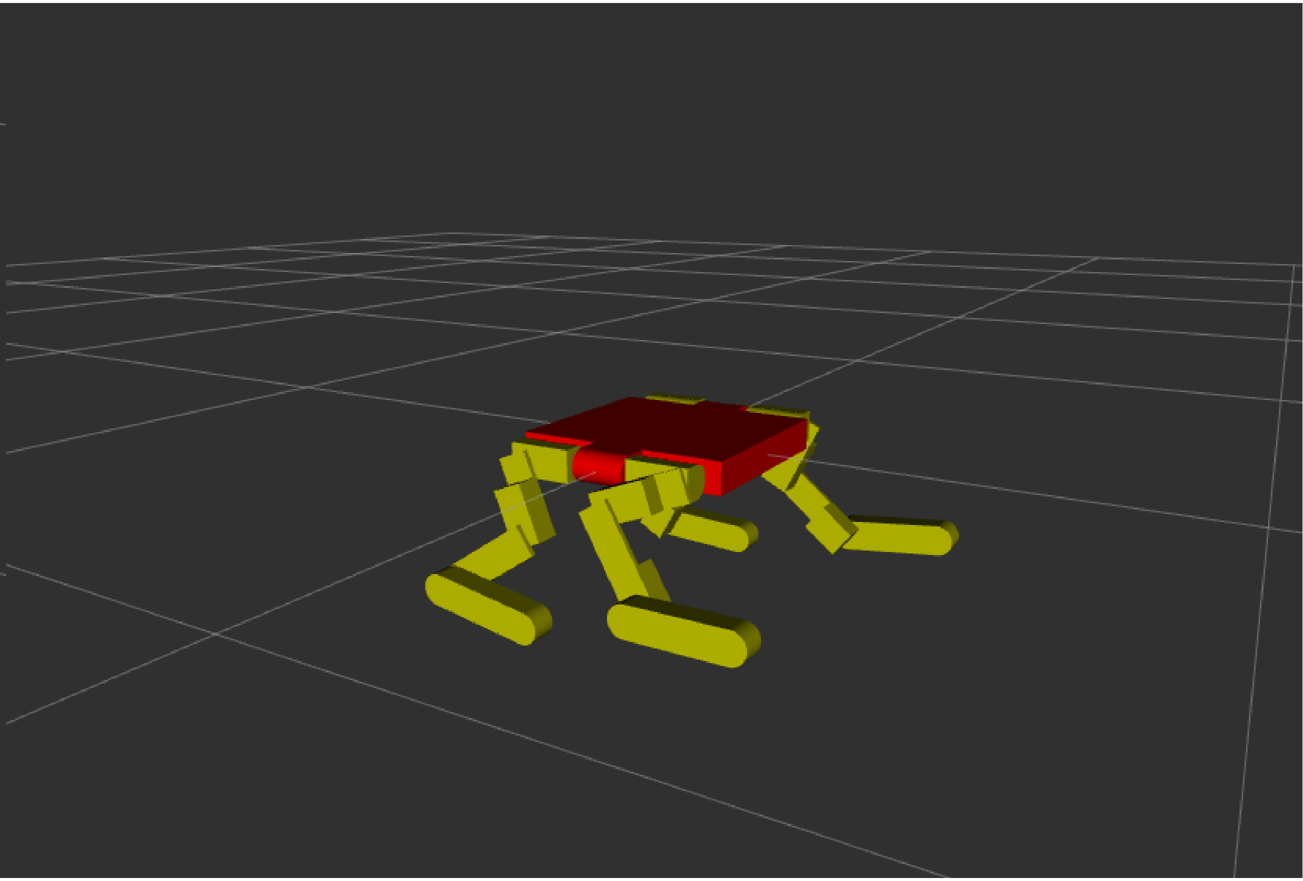


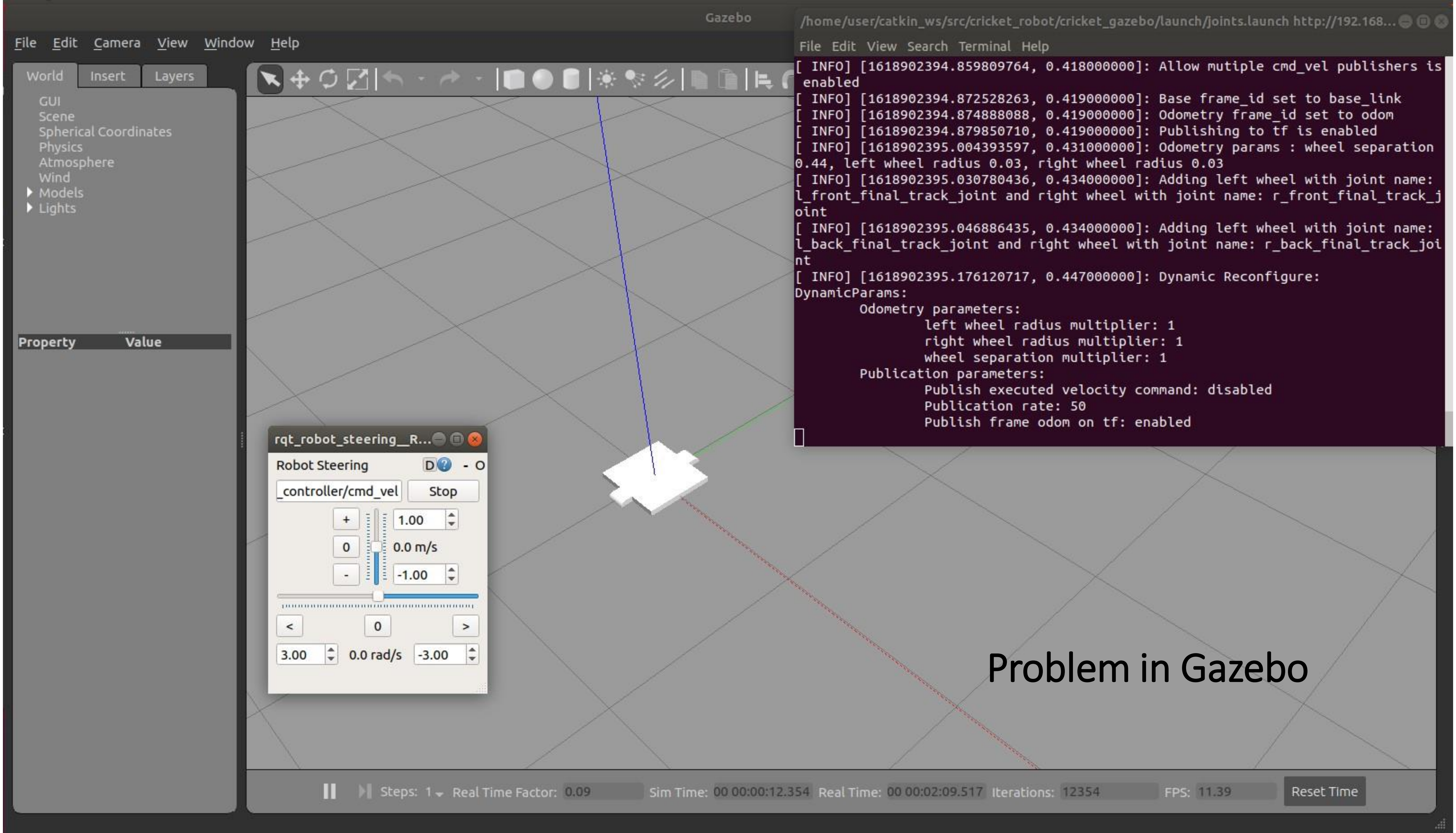
Reward parameters

Forward scaling → forward reward X	A reward for following the right direction to reach the goal
Penalty deviation Y	Total deviation penalty
Penalty deviation Z	
Torque penalty	Avoid that the robot makes useless movements
Duration reward/penalty	Optimize the time to reach the goal
(optional) energy consumption reward/penalty	Optimize the energy consumption

URDF model







Problem in Gazebo

It is a reward and not a penalty to avoid the robot decides to encounter stop conditions instead of going on

Avoid straying from decided path

The error ε represent the difference between the predicted $\hat{Q}(s, a)$ and the measured $Q(s, a)$.

Discount factor $0 < \gamma \leq 1$

$$\mathcal{R}_t = \boxed{v_x} + \boxed{w_t \cdot \mathrm{t}} - \boxed{\mathrm{z}^2} - \boxed{\mathrm{w_y y}^2} - \boxed{\sum_i (w^i q_{t-1}^i)^2} - \boxed{w_\varepsilon \left(\sum_{i=0}^{5} \gamma^i \varepsilon_{t-i} \right)^2}$$

Forward velocity

Distance from basic height

Minimize the encoders effort (considering the previous values of q)

This gives the robot a way to understand the error term and develop a policy accordingly

It is a reward and not a penalty to avoid the robot decides to encounter stop conditions instead of going on

Avoid straying from decided path

The error ε represent the difference between the predicted $\hat{Q}(s,a)$ and the measured $Q(s,a)$.

Discount factor
$0 < \gamma \leq 1$

$$\boldsymbol{\mathcal{R}}_t = \boxed{v_x} + \boxed{0.0625 \cdot \mathrm{t}} - \boxed{\mathrm{z}^2} - \boxed{10\mathrm{y}^2} - \boxed{0.02 \sum_i (q_{t-1}^i)^2} - \boxed{100\left(\sum_{i=0}^{5} \gamma^i \varepsilon_{t-i}\right)^2}$$

Forward velocity

Distance from basic height

Minimize the encoders effort (considering the previous values of q)

This gives the robot a way to understand the error term and develop a policy accordingly

Number of states (input layer dimension for the actor’s neural network)

Robot position	X, Y, Z	3
Robot velocity	v_x, v_y, v_z	3
Robot angle	ϕ, θ, ψ	3
Robot angle rate	$\omega_\phi, \omega_\theta, \omega_\psi$	3
4 legs	four joints per leg	16
4 tracks	4 track rotation values	4
Contact forces	6 normal forces per track	24
Computed errors	ε_t	5
Total		61

Difference from final position

The error ε represent the difference between the **predicted** $\widehat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})$ and the **measured** $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})$**.**

$\beta_i = 0 \text{ or } 1$
Penalty if the robot touches the environment

Discount factor
$0 < \gamma \leq 1$

Difference from final optimal position

$$\boldsymbol{\mathcal{R}}_t = -\sum_{i=1}^{no\ tracks} \left[\boxed{\alpha_i\left(q_{t-1}^i\right)^2} + \boxed{\beta_i\left(q_{t-1}^i\right)^2} + \boxed{\kappa_\tau^i\left|\tau_{t-1}^i\right|} + \boxed{w_q^i\left(\Delta q_t^i\right)^2} \right] - \boxed{w_\varepsilon\left(\sum_{i=0}^{5} \gamma^i \varepsilon_{t-i}\right)^2} - \boxed{w_t t} - \boxed{\Delta X - \Delta Y - \Delta Z}$$

$\alpha_i = 0 \text{ or } 1$
Penalty if the robot touches itself

$\kappa_\tau^i = -1 \text{ or } 1$
reward/penalty based on the direction of the last rotation

This gives the robot a way to understand the error term and develop a policy accordingly

Time passed

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

$$\boldsymbol{\mathcal{R}}_t = -\sum_{i=1}^{no\ tracks}\left[\alpha_i\left(q_{t-1}^i\right)^2 + \beta_i\left(q_{t-1}^i\right)^2 + \kappa_\tau^i\left|\tau_{t-1}^i\right| + w_q^i\left(\Delta q_t^i\right)^2\right] - w_\varepsilon\left(\sum_{i=0}^{5}\gamma^i\varepsilon_{t-i}\right)^2 - w_t t - w_{\mathrm{X}}(\Delta X)^2 - \mathrm{w}_{\mathrm{Y}}(\Delta Y)^2 - \mathrm{w}_{\mathrm{Z}}(\Delta Z)^2 - w_\psi(\Delta\psi)^2 - w_\theta(\Delta\theta)^2 - w_\phi(\Delta\phi)^2$$

The error ε represent the difference between the **predicted** $\widehat{\boldsymbol{Q}}(\boldsymbol{s},\boldsymbol{a})$ and the **measured** $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})$.

Difference with the joints final position

$\beta_i = 0 \text{ or } 1$
Penalty if the robot touches the environment

Discount factor
$0 < \gamma \leq 1$

$$\boldsymbol{\mathcal{R}}_t = -\sum_{i=1}^{no\ tracks}\left[\boxed{\alpha_i\left(q_{t-1}^i\right)^2} + \boxed{\beta_i\left(q_{t-1}^i\right)^2} + \boxed{\kappa_\tau^i\left|\tau_{t-1}^i\right|} + \boxed{w_q^i\left(\Delta q_t^i\right)^2}\right] - \boxed{w_\varepsilon\left(\sum_{i=0}^{5}\gamma^i\varepsilon_{t-i}\right)^2} - \boxed{w_t t} - \overbrace{w_{\mathrm{X}}(\Delta X)^2 - w_{\mathrm{Y}}(\Delta \mathrm{Y})^2 - w_{\mathrm{Z}}(\Delta \mathrm{Z})^2}^{\text{Difference with the robot's final centre of mass position}} - \underbrace{w_\psi(\Delta\psi)^2 - w_\theta(\Delta\theta)^2 - w_\phi(\Delta\phi)^2}_{\text{Difference with the robot's final body rotation}}$$

$\alpha_i = 0 \text{ or } 1$
Penalty if the robot touches itself

$\kappa_\tau^i = -1 \text{ or } 1$
reward/penalty based on the direction of the last rotation

This gives the robot a way to understand the error term and develop a policy accordingly

Time passed

Deep Deterministic Policy Gradient

[1] Lillicrap, Timothy P., et al. ”*Continuous control with deep reinforcement learning*” (2015)
[2] Zhipeng L., Hao C., et al. “*Adversarial Deep Reinforcement Learning in Portfolio Management*” (2018)

Initialize actor and critic networks $\mathcal{A}(s, e|\theta^{\mathcal{A}})$ and $\mathcal{C}(s, a, e|\theta^{\mathcal{C}})$ with random weights $\theta^{\mathcal{A}}$ and $\theta^{\mathcal{C}}$
Initialize target network $\mathcal{A}'$ and $\mathcal{C}'$ with random weights $\theta^{\mathcal{A}'} \leftarrow \theta^{\mathcal{A}}$ and $\theta^{\mathcal{C}'} \leftarrow \theta^{\mathcal{C}}$
Initialize replay buffer $\mathcal{R}$
for *n=1* to $N_{episodes}$ **do**:

    Procedure RobotInit
    Generate a random stable starting state s in the unstructured environment
    end procedure RobotInit

    Procedure Simulation
    for t=0 to T **do**:
        gererate action $a_t \leftarrow \mathcal{A}(s_t, e|\theta^{\mathcal{A}})$
        Execute action a_t: observe reward r_t and new state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{R}$
        Sample a random **minibatch** of N transitions (s, a, r, s') $from$ $\mathcal{R}$
        for (s_i, a_i, r_i, s_{i+1}) in minibatch **do**:
            set $y_i = r_i + \gamma \cdot \mathcal{C}'(s_{i+1}, \mathcal{A}'(s_{i+1}, e|\theta^{\mathcal{A}'}), e|\theta^{\mathcal{C}'})$
        end for
        Update critic by minimizing the loss $L = \frac{1}{N} \sum_i \left(y_i - \mathcal{C}(s_i, a_i, e|\theta^{\mathcal{C}})\right)^2$
        Update actor policy by policy gradient
        Update target networks $\mathcal{A}'$ and $\mathcal{C}'$
        checkStopCondition()
    end for
    end procedure Simulation
end for