# Sterownie obiektami 3D – uczenie ze wzmocnieniem

**Damian Kantorowski**

## 1. Wykorzystane biblioteki

Gymnasium (MuJoCo), Stable Baselines3, RL Baselines3 Zoo, PyTorch, Optuna, Tensorboard

## 2. Opis algorytmów

### Proximal Policy Optimization (PPO)

PPO to algorytm on-policy, który wykorzystuje mechanizm actor-critic do optymalizacji polityki. Algorytm ten charakteryzuje się stabilnością uczenia i jest szczególnie skuteczny w środowiskach z ciągłymi przestrzeniami akcji.

Inicjalizacja:

1. Zainicjalizuj sieć polityki $\pi\theta(a|s)$ z parametrami $\theta$
2. Zainicjalizuj sieć wartości $V^{\phi}(s)$ z parametrami $\phi$
3. Ustaw hiperparametry: współczynnik clippingu $\varepsilon$, learning rate, rozmiar batcha

Główna pętla uczenia:

4. Zbierz trajektorie używając obecnej polityki $\pi\theta$
5. Oblicz advantage estimates $A^t = \delta t + (\gamma\lambda)\delta t+1 + \ldots$ gdzie $\delta t = rt + \gamma V(st+1) - V(st)$
6. Oblicz returns $Rt = At + V(st)$

Aktualizacja polityki:

7. Dla każdego mini-batcha:
   - Oblicz ratio $rt(\theta) = \pi\theta(at|st) / \pi\theta\_old(at|st)$
   - Oblicz clipped objective: $L^{CLIP} = \min(rt(\theta)At, clip(rt(\theta), 1-\varepsilon, 1+\varepsilon)At)$
   - Aktualizuj parametry $\theta$ maksymalizując $L^{CLIP}$

Aktualizacja funkcji wartości:

8. Aktualizuj parametry $\phi$ minimalizując błąd kwadratowy: $L^{VF} = (V\phi(st) - Rt)^2$
9. Powtarzaj kroki 4-8 do osiągnięcia zbieżności

### Soft Actor-Critic (SAC)

SAC to algorytm off-policy. Jest szczególnie skuteczny w środowiskach z dużymi przestrzeniami stanów.

Inicjalizacja:

1. Zainicjalizuj sieć aktora $\pi\phi(a|s)$ z parametrami $\phi$
2. Zainicjalizuj dwie sieci krytyka $Q\psi1(s,a)$ i $Q\psi2(s,a)$ z parametrami $\psi1$, $\psi2$
3. Zainicjalizuj target networks dla krytyków: $Q\psi\bar{1}$, $Q\psi\bar{2}$
4. Zainicjalizuj bufor doświadczeń R
5. Ustaw parametr temperatury $\alpha$ (lub jego automatyczne dostrajanie)

Główna pętla uczenia:

6. Dla każdego kroku środowiska:
   - Obserwuj stan st
   - Wybierz akcję $at \sim \pi\phi(\cdot|st)$
   - Wykonaj akcję i obserwuj (st, at, rt, st+1)
   - Zapisz przejście w buforze R

Aktualizacja krytyków:

7. Próbkuj mini-batch z bufora R
8. Oblicz target dla Q-funkcji:
   - Próbkuj $at+1 \sim \pi\phi(\cdot|st+1)$
   - $yt = rt + \gamma(\min(Q\psi\bar{1}(st+1, at+1), Q\psi2(st+1, at+1)) - \alpha \log \pi\phi(at+1|st+1))$
9. Aktualizuj parametry $\psi1$, $\psi2$ minimalizując: $L = (Q\psi i(st,at) - yt)^2$

Aktualizacja aktora:

10. Próbkuj akcje $at \sim \pi\phi(\cdot|st)$ dla stanów z mini-batcha
11. Aktualizuj $\phi$ maksymalizując: $J = E[\min(Q\psi1(st,at), Q\psi2(st,at)) - \alpha \log \pi\phi(at|st)]$

Aktualizacja target networks:

12. Soft update target networks: $\psi\bar{i} \leftarrow \tau\psi i + (1-\tau)\psi\bar{i}$

Automatyczne dostrajanie temperatury (opcjonalne):

13. Aktualizuj $\alpha$ minimalizując: $J(\alpha) = -\alpha(\log \pi\phi(at|st) + \bar{H})$
14. Powtarzaj kroki 6-13

### Twin Delayed DDPG (TD3)

TD3 to algorytm off-policy zaprojektowany do obsługi ciągłych przestrzeni akcji. Charakteryzuje się szybkim uczeniem, ale wymaga więcej kroków do osiągnięcia maksymalnej nagrody.

Inicjalizacja:

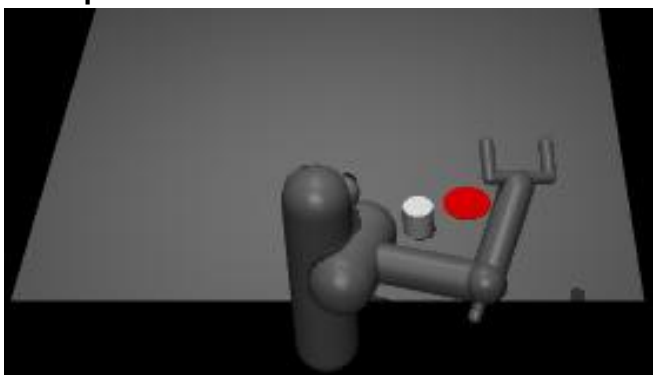
1. Zainicjalizuj sieć aktora $\pi\theta(s)$ z parametrami $\theta$
2. Zainicjalizuj dwie sieci krytyka $Q\phi1(s,a)$ i $Q\phi2(s,a)$ z parametrami $\phi1$, $\phi2$
3. Zainicjalizuj target networks: $\pi\bar{\theta}$, $Q\phi\bar{1}$, $Q\phi\bar{2}$
4. Zainicjalizuj bufor doświadczeń R
5. Ustaw hiperparametry: policy delay d, target noise $\sigma$, noise clip c

Główna pętla uczenia:

6. Dla każdego kroku środowiska:
   - Obserwuj stan st
   - Wybierz akcję $at = \pi\theta(st) + \varepsilon$ gdzie $\varepsilon \sim N(0, \sigma^2)$
   - Wykonaj akcję i obserwuj (st, at, rt, st+1)
   - Zapisz przejście w buforze R

Aktualizacja krytyków:
7. Próbkuj mini-batch z bufora R
8. Oblicz target akcję z szumem:
- $\tilde{a} = \pi\theta(st+1) + clip(\varepsilon, -c, c)$ gdzie $\varepsilon \sim N(0, \sigma^2)$
9. Oblicz target wartość:
- $yt = rt + \gamma \min(Q\phi\bar{1}(st+1, \tilde{a}), Q\phi 2(st+1, \tilde{a}))$
10. Aktualizuj parametry ϕ1, ϕ2 minimalizując: $L = (Q\phi i(st,at) - yt)^2$
Opóźniona aktualizacja aktora:
11. Co d kroków:
- Aktualizuj θ maksymalizując: J = E[Qϕ1(st, πθ(st))]
- Soft update wszystkich target networks:
- $\theta\bar{} \leftarrow \tau\theta + (1-\tau)\theta\bar{}$
- $\phi i\bar{} \leftarrow \tau\phi i + (1-\tau)\phi i\bar{}$
12. Powtarzaj kroki 6-11

## 3. Opis środowisk



**Pusher**

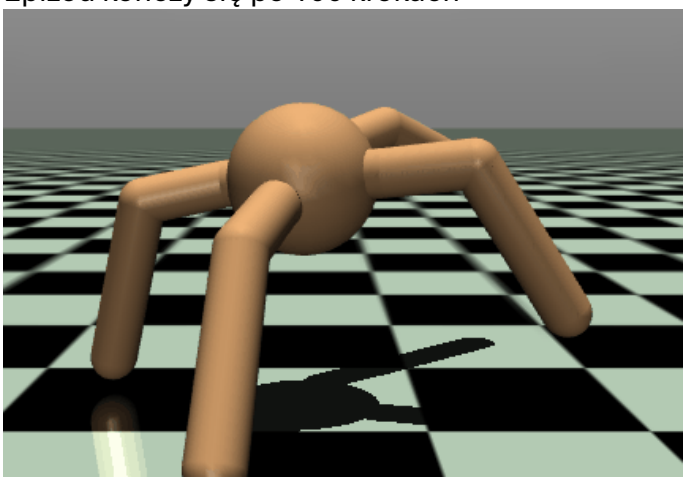
Pusher symuluje zadanie manipulacji obiektów, gdzie 7-stawowe ramię robotyczne musi pchać cylindryczny obiekt do określonej pozycji docelowej. To środowisko łączy kontrolę ramienia z fizyką kontaktu obiekt-manipulator.

**Przestrzeń obserwacji (23 wymiary):**
Pozycja końcówki ramienia w 3D (3 wartości)
Pozycja obiektu w 3D (3 wartości)
Pozycja celu w 3D (3 wartości)
Pozycja końcówki ramienia względem obiektu (3 wartości)
Pozycja obiektu względem celu (3 wartości)
Prędkości końcówki ramienia w 3D (3 wartości)
Prędkości obiektu w 3D (3 wartości)
Kąty wszystkich stawów ramienia (7 wartości - tylko orientacja, bez pozycji)

**Przestrzeń akcji (7 wymiarów):**
Momenty obrotowe dla każdego stawu ramienia
Wartości ciągłe w zakresie [-2, 2]
Kontrola 7-stawowego ramienia robotycznego
Bez bezpośredniej kontroli obiektu (tylko przez kontakt fizyczny)

**Funkcja nagrody:**
Nagroda za cel: -odległość między obiektem a celem (im bliżej, tym wyższa nagroda)
Nagroda za dotarcie: Bonus gdy obiekt znajdzie się w pobliżu celu
Kara za kontrolę: -0.1 * suma kwadratów akcji (zachęca do płynnych ruchów)
Kara za odległość od obiektu: Zachęca ramię do pozostania blisko obiektu

**Warunki zakończenia:**
Obiekt spadnie poza obszar roboczy
Epizod kończy się po 100 krokach



**Ant**

Ant symuluje robota z ciałem centralnym i czterema nogami, każda z dwoma stawami. Robot ma 8 stopni swobody i porusza się w przestrzeni 3D. To złożone środowisko wymagające koordynacji wszystkich kończyn.

**Przestrzeń obserwacji (27 wymiarów):**
Pozycja z (wysokość) tułowia
Orientacja tułowia w 3D (4 kwaterniony)
Kąty wszystkich stawów (8 wartości)
Prędkości kątowe wszystkich stawów (8 wartości)
Prędkości liniowe tułowia w 3D (3 wartości)
Prędkości kątowe tułowia w 3D (3 wartości)

**Przestrzeń akcji (8 wymiarów):**
Momenty obrotowe dla każdego stawu (4 stawy biodrowe + 4 stawy "kolanowe")
Wartości ciągłe w zakresie [-1, 1]
Kontrola niezależna każdego stawu

**Funkcja nagrody:**
Nagroda za prędkość: Prędkość w kierunku x
Nagroda za przetrwanie: +1 za każdy krok (zachęca do utrzymania się w pionie)
Kara za kontrolę: -0.5 * suma kwadratów akcji
Kara za kontakt: -0.5 * suma sił kontaktu (zachęca do delikatnych ruchów)

**Warunki zakończenia:**
Wysokość tułowia < 0.2m lub > 1.0m
Epizod kończy się po 1000 krokach lub wcześniej przy upadku

**Humanoid**

Humanoid to najbardziej złożone środowisko, symulujące dwunożnego robota humanoidalnego. Model składa się z tułowia, dwóch rąk i dwóch nóg z realistyczną strukturą stawów. Robot ma 17 stopni swobody i musi nauczyć się chodzenia lub biegu w pozycji pionowej.

**Przestrzeń obserwacji (376 wymiarów):**

Pozycja z (wysokość) tułowia
Orientacja tułowia w 3D (4 kwaterniony)
Kąty wszystkich stawów (23 wartości)
Prędkości kątowe wszystkich stawów (23 wartości)
Prędkości liniowe tułowia w 3D (3 wartości)
Prędkości kątowe tułowia w 3D (3 wartości)
Pozycje wszystkich części ciała względem środka masy
Prędkości wszystkich części ciała
Siły kontaktu z podłożem

**Przestrzeń akcji (17 wymiarów):**

Momenty obrotowe dla każdego kontrolowanego stawu
Wartości ciągłe w zakresie [-0.4, 0.4]
Kontrola stawów w biodrach, kolanach, kostkach, barkach, łokciach

**Funkcja nagrody:**

Nagroda za prędkość: 1.25 * prędkość w kierunku x
Nagroda za przetrwanie: +5 za każdy krok w pionie
Kara za kontrolę: -0.1 * suma kwadratów akcji
Kara za kontakt: -1e-3 * suma sił kontaktu głowy z podłożem
Dodatkowe nagrody: Za utrzymanie prawidłowej postawy

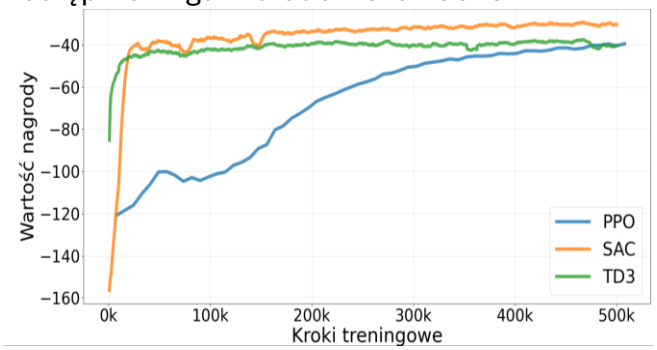
**Warunki zakończenia:**

Wysokość tułowia < 1.0m lub > 2.0m
Pozycja z głowy < 1.0m (upadek na głowę)
Epizod kończy się po 1000 krokach lub wcześniej przy upadku

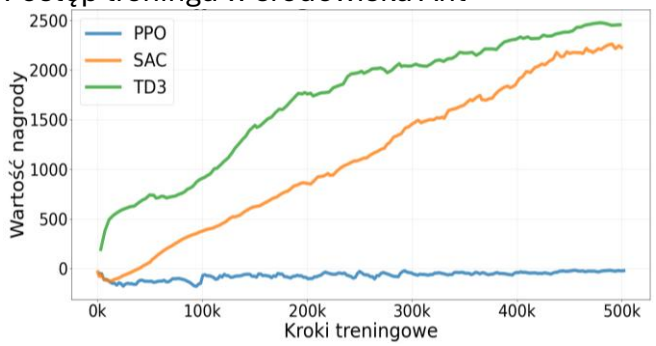
## 4. Wyniki

Treningi trwające 500000 kroków przeprowadzone zostały z użyciem trzech algorytmów w trzech środowiskach, w każdym przypadku użyto dwóch zestawów hiperparametrów odpowiednich dla danego algorytmu: domyślnych znajdujących się w implementacji z biblioteki Stable Baselines3 oraz dostrojonych pod konkretne środowisko. Optymalizacja hiperparametów w RL Baselines3 Zoo obejmowała po około 100 prób treningów, każda po maksymalnie 50000 kroków, w celu zmaksymalizowania średniej wartości nagrody. Wykorzystano sampler TPE (Tree of Parzen Estimators) do inteligentnego wyboru kolejnych konfiguracji hiperparametrów oraz pruner median do wczesnego przerywania nieperspektywicznych prób. W celu przyspieszenia eksperymentów użyta została wersja PyTorch działająca z backendem CUDA na karcie graficznej Nvidia GeForce RTX 5060 Ti. Postęp treningów prezentują wykresy średnich nagród z epizodów podczas najlepszego przebiegu danego algorytmu. Ostateczna ewaluacja zawiera średnią nagród z 50 symulacji przeprowadzonych dla różnych ziaren generatora.

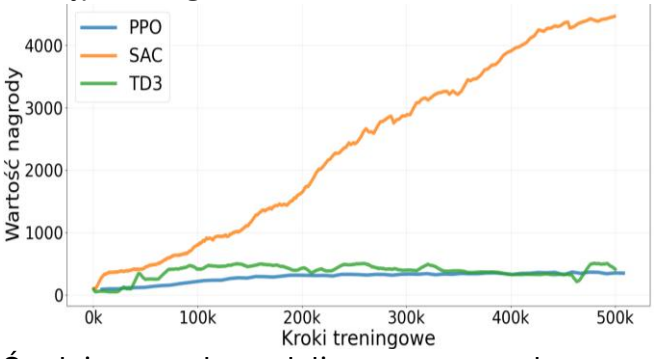
Postęp treningu w środowisku Pusher




Postęp treningu w środowisku Ant



Postęp treningu w środowisku Humanoid




Średnie nagrody modeli wytrenowanych z domyślnymi hiperparametrami

| | **Pusher** | **Ant** | **Humanoid** |
|---|---|---|---|
| **PPO** | -32.57 ± 3.65 | 119.21 ± 150.55 | 430.21 ± 85.57 |
| **SAC** | -26.17 ± 4.04 | 2687.78 ± 983.27 | 5004.01 ± 322.52 |
| **TD3** | -34.77 ± 4.50 | 2745.22 ± 770.73 | 125.55 ± 8.78 |

Średnie nagrody modeli wytrenowanych z dostrojonymi hiperparametrami

| | **Pusher** | **Ant** | **Humanoid** |
|---|---|---|---|
| **PPO** | -295.59 ± 3.31 | 823.14 ± 410.09 | 347.79 ± 21.81 |
| **SAC** | -34.72 ± 4.09 | 2563.90 ± 1558.92 | 2967.81 ± 1493.29 |
| **TD3** | -41.39 ± 4.54 | 983.46 ± 6.91 | 449.18 ± 53.27 |

W większości przypadków domyślne hiperparametry okazały się bardziej odpowiednie. Ostatecznie, do demonstracji efektów uczenia ze wzmocnieniem wybrano najlepsze modele: SAC w środowisku Pusher, TD3 w środowisku Ant oraz SAC w środowisku Humanoid.

## 5. Źródła

Mark Towers i in.: Gymnasium: A Standard Interface for Reinforcement Learning Environments
https://github.com/Farama-Foundation/Gymnasium
Antonin Raffin i in.: Stable-Baselines3: Reliable Reinforcement Learning Implementations
https://github.com/DLR-RM/stable-baselines3
Antonin Raffin: RL Baselines3 Zoo
https://github.com/DLR-RM/rl-baselines3-zoo
John Schulman i in.: Proximal Policy Optimization Algorithms
Tuomas Haarnoja i in.: Soft Actor-Critic Algorithms and Applications
Scott Fujimoto i in.: Addressing Function Approximation Error in Actor-Critic Method