

Tomasz GRZEJSZCZAK, Michał LASAK, Maja RUDNICKA
Politechnika Śląska

GŁĘBOKIE UCZENIE ZE WZMOCNIENIEM A REGULACJA PID: PORÓWNANIE AUTONOMICZNEJ NAWIGACJI STEROWCA W ŚRODOWISKU 3D

Streszczenie. W niniejszym artykule porównano algorytmy sterowania PID i DRL (PPO) w autonomicznej nawigacji sterowca po punktach orientacyjnych z wykorzystaniem symulacji 3D opartej na LiDAR-ze. Testy przeprowadzono w ramach misji obejmujących wiele punktów orientacyjnych w środowisku z przeszkodami; jako wskaźniki skuteczności przyjęto wskaźnik powodzenia, długość trasy, liczbę kroków potrzebnych do zakończenia misji oraz odległość od przeszkód. Algorytm PID sprawdza się w środowiskach o niewielkiej gęstości przeszkód, natomiast algorytm DRL uczy się płynniejszych i bardziej odpornych trajektorii oraz radzi sobie z gęstszą siecią przeszkód. Opracowano wstępną wersję sprzętu służącą do przeniesienia wyników symulacji do warunków rzeczywistych.

DEEP REINFORCEMENT LEARNING VS. PID CONTROL: A COMPARISON OF AUTONOMOUS NAVIGATION OF A BLIMP IN 3D ENVIRONMENT

Summary. This paper compares PID and DRL (PPO) control for autonomous blimp waypoint navigation using 3D LiDAR simulation. Tested on multi-waypoint missions through clutter, metrics are success rate, path length, completion steps, and clearance. PID works in sparse environments, but DRL learns smoother, more robust trajectories and handles denser obstacles. Preliminary hardware is built for sim-to-real transfer

1. Wstęp

W ostatnich latach znacznie wzrosło wykorzystanie bezzałogowych statków powietrznych (UAV) w różnych sektorach, w tym w monitorowaniu środowiska, zarządzaniu kryzysowym oraz operacjach poszukiwawczo-ratowniczych. Chociaż dominują platformy o skrzydłach obrotowych i stałych, statki powietrzne lżejsze od powietrza (LTA), takie jak autonomiczne sterowce, oferują wyjątkowe zalety w przypadku misji długotrwałych ze względu na niskie zużycie energii i naturalne bezpieczeństwo w środowiskach z udziałem ludzi lub w zatłoczonych przestrzeniach [6, 11]. W przeciwieństwie do quadkopterów, których działanie ogranicza krótki czas pracy baterii, sterowce mogą pozostawać w powietrzu przez dłuższy czas, co czyni je idealnymi do ciągłego monitorowania oraz zadań usługowych w pomieszczeniach [6, 3].

Pomimo tych zalet autonomiczna nawigacja sterowców jest utrudniona przez złożoną, nieliniową dynamikę oraz znaczne opóźnienia w reakcji systemu [6, 3]. Sterow-

ce są często niedostatecznie sterowane, zazwyczaj pozbawione bezpośredniej kontroli bocznej, a ich wyporność jest wrażliwa na zmiany ciśnienia otoczenia i temperatury [6, 11]. Ponadto nawigacja w zatłoczonych środowiskach 3D wymaga solidnych strategii sterowania, zdolnych do przetwarzania wielowymiarowych danych z czujników, takich jak poziome i pionowe skany LiDAR, przy jednoczesnym utrzymywaniu bezpiecznej odległości od przeszkód [5].

2. Przegląd literatury

Tradycyjne regulatory PID są powszechnie stosowane ze względu na prostotę, jednak ich ręczne strojenie dla nieliniowej dynamiki sterowców w środowiskach o dużej gęstości przeszkód pozostaje wyzwaniem [3, 6]. Jako obiecującą alternatywę wskazuje się algorytmy głębokiego uczenia ze wzmocnieniem (DRL), takie jak PPO (Proximal Policy Optimization) oraz SAC (Soft Actor-Critic), które pozwalają na uzyskanie płynniejszych i bardziej odpornych trajektorii [6, 2].

Współczesne badania nad planowaniem tras dla UAV ewoluują od klasycznych heurystyk (A^* , Dijkstra) w stronę rozwiązań opartych na uczeniu, integrujących fuzję danych z czujników LiDAR i wizyjnych w czasie rzeczywistym [4, 5, 1]. Zaawansowane podejścia obejmują systemy hybrydowe łączące sterowanie predykcyjne (MPC) z DRL [9] oraz struktury dwupoziomowe optymalizujące sterowanie platformami niedostawianymi poprzez rekonfigurację środka masy [11]. Kluczowym wyzwaniem pozostaje transfer wyuczonych polityk z symulacji do rzeczywistości (ang. *sim-to-real*), realizowany poprzez wykorzystanie symulatorów o wysokiej wierności oraz technik randomizacji domeny [7, 8]. Badania porównawcze dowodzą, że kontrolery oparte na RL mogą osiągać stabilność i precyzję przewyższającą loty pilotowane przez ludzi [8].

3. Metodyka

3.1. Środowisko i model symulacyjny

Eksperymenty przeprowadzono w przestrzeni o wymiarach $10 \times 10 \times 5$ m, zawierającej przeszkody w formie filarów i obiektów zawieszonych, tworzących wąskie korytarze (rys. 3). Sterowiec opisuje wektor stanu $s = (x, y, z, \theta)^\top$. Wykorzystano model kinematyczny:

$$\dot{x} = v_f \cos \theta, \quad \dot{y} = v_f \sin \theta, \quad \dot{z} = v_v, \quad \dot{\theta} = \omega \quad (1)$$

gdzie $v_f \geq 0$, v_v i ω to zadawane prędkości sterujące. Pojazd wyposażono w 24-wiązkowy LiDAR poziomy (360°) oraz dwie wiązki pionowe, których odczyty są normalizowane.

3.2. Model percepcji

Percepcja otoczenia opiera się na dwóch wirtualnych systemach lidarowych:

- **LiDAR poziomy:** 24 wiązki rozmieszczone co 15° (360° zasięgu), mierzące odległość do najbliższych przeszkód lub ścian.
- **LiDAR pionowy:** 2 wiązki skierowane w dół i w górę, mierzące dystans do podłogi i sufitu.

Odległości są obliczane poprzez wyznaczanie punktów przecięcia promienia z płaszczyznami ścian lub prostopadłościanami przeszkód. Wszystkie odczyty podlegają normalizacji poprzez podzielenie ich przez maksymalny zasięg w danej płaszczyźnie ($\sqrt{L_x^2 + L_y^2}$ dla poziomu oraz L_z dla pionu).

3.3. Definicja misji i funkcje celu

Zadanie polega na sekwencyjnym odwiedzeniu czterech punktów orientacyjnych w_i przy promieniu akceptacji $r_{wp} = 0,5$ m. W przypadku agenta DRL zastosowano funkcję nagrody obejmującą: premię za przetrwanie (+0,1/krok), nagrodę za postęp w stronę celu ($d_{prev} - d_{curr}$), premie za punkty pośrednie (+20) i końcowy (+100) oraz karę za kolizję (-10).

3.4. Klasyczny regulator PID

Jako punkt odniesienia dla modelu nieuczącego się zaprojektowano ręcznie dostrójony regulator PID, który generuje trzy sygnały sterujące v_f , ω , v_v na podstawie aktualnego stanu i aktywnego punktu nawigacyjnego. Regulator składa się z trzech oddzielnych pętli:

Sterowanie kursem. Požadany kurs θ_{des} jest kątem wektora skierowanego od sterowca do punktu nawigacyjnego w płaszczyźnie x - y :

$$\theta_{des} = \text{atan2}(w_{y,i} - y, w_{x,i} - x). \quad (2)$$

Szybkość odchylenia jest regulowana zgodnie z algorytmem PID na podstawie błędu kursu $e_\theta = \text{wrap}(\theta_{des} - \theta)$:

$$\omega = K_p^\theta e_\theta + K_i^\theta \int e_\theta dt + K_d^\theta \frac{de_\theta}{dt}, \quad (3)$$

z współczynnikami wzmocnienia $K_p^\theta = 0,8$, $K_i^\theta = 0,0$, $K_d^\theta = 0,2$ po dostrojeniu.

Sterowanie wysokością. Prosty regulator proporcjonalny steruje prędkością pionową na podstawie błędu wysokości $e_z = w_{z,i} - z$:

$$v_v = K_p^z e_z, \quad K_p^z = 0,5. \quad (4)$$

Prędkość do przodu. Prędkość posuwu jest ustalana proporcjonalnie do odległości poziomej do punktu nawigacyjnego, z ograniczeniem do $v_f^{\max} = 1,0$ m/s:

$$v_f = \min \left(K_f \sqrt{(w_{x,i} - x)^2 + (w_{y,i} - y)^2}, v_f^{\max} \right), \quad K_f = 0,5. \quad (5)$$

Unikanie przeszkód. Do polecenia odchylenia oraz prędkości lotu do przodu dodawany jest reaktywny człon odpychający na podstawie minimalnego odczytu LiDAR-u w płaszczyźnie poziomej $d_{\min}^H$. Niech $d_{\text{safe}} = 1,5$ m będzie marginesem bezpieczeństwa. Gdy $d_{\min}^H < d_{\text{safe}}$, określamy kierunek najbliższej przeszkody α_{obs} i obliczamy odpychający błąd kursu $e_{\text{avoid}} = \text{wrap}(\alpha_{\text{obs}} + \pi - \theta)$. Szybkość obrotu jest zwiększana o

$$\omega \leftarrow \omega + K_{\text{avoid}} w(d_{\min}^H) e_{\text{avoid}}, \quad (6)$$

gdzie $K_{\text{avoid}} = 2,0$, a $w(d) = (1 - d/d_{\text{safe}})^2$ jest wagą kwadratową. Jednocześnie prędkość lotu do przodu jest mnożona przez $\max(0, 1 - 0,9 w(d_{\min}^H))$, co powoduje niemal całkowite zatrzymanie sterowca, gdy znajduje się on bardzo blisko przeszkody.

Wszystkie polecenia sterujące są ograniczane do dopuszczalnych przedziałów $v_f \in [0, 1]$, $\omega \in [-1, 1]$, $v_v \in [-0,5, 0,5]$, aby dopasować je do przestrzeni akcji agenta RL.

3.5. Kontroler oparty na głębokim uczeniu ze wzmocnieniem

Zadanie nawigacji formułujemy jako proces decyzyjny Markowa $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$ i trenujemy politykę przy użyciu metody proksymalnej optymalizacji polityki (PPO) [10].

Przestrzeń obserwacji. Agent otrzymuje 29-wymiarowy wektor obserwacji:

$$\mathbf{o} = [\mathbf{h}, \mathbf{v}, \mathbf{g}]^\top \in \mathbb{R}^{29}, \quad (7)$$

gdzie $\mathbf{h} \in [0, 1]^{24}$ to znormalizowane odczyty LiDAR-u w płaszczyźnie poziomej, $\mathbf{v} \in [0, 1]^2$ to znormalizowane odczyty LiDAR-u w płaszczyźnie pionowej, a $\mathbf{g} \in \mathbb{R}^3$ to wektor wskazujący aktualny punkt na trasie wyrażony w układzie odniesienia korpusu:

$$\mathbf{g} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} w_{x,i} - x \\ w_{y,i} - y \\ w_{z,i} - z \end{pmatrix}. \quad (8)$$

Ta reprezentacja egocentryczna eliminuje konieczność szacowania przez agenta swojej pozycji globalnej.

Przestrzeń akcji. Akcje są ciągłe i odpowiadają fizycznym elementom sterującym: $\mathbf{a} = [a_f, a_\omega, a_v]^\top$, gdzie $a_f \in [0, 1]$ (ruch do przodu), $a_\omega \in [-1, 1]$ (prędkość obrotowa wokół osi yaw) oraz $a_v \in [-1, 1]$ (prędkość pionowa). Fizyczne polecenia uzyskuje się poprzez skalowanie: $v_f = a_f \cdot v_f^{\max}$, $\omega = a_\omega \cdot \omega^{\max}$ ($\omega^{\max} = 1, 0$ rad/s), $v_v = a_v \cdot v_v^{\max}$ ($v_v^{\max} = 0, 5$ m/s). Należy zwrócić uwagę na wyraźne ograniczenie ruchu do przodu do wartości nieujemnych, odzwierciedlające ograniczenie kinematyczne rzeczywistego sterowca.

Sieć polityki. Polityka π_ϕ i funkcja wartości V_ψ są reprezentowane przez wielowarstwowy perceptron (MLP) z dwiema warstwami ukrytymi po 64 neurony każda i aktywacjami tanh. Sieć generuje średnią z rozkładu Gaussa o logarytmicznym odchyleniu standardowym niezależnym od stanu, a oddzielna głowica liniowa generuje wartość.

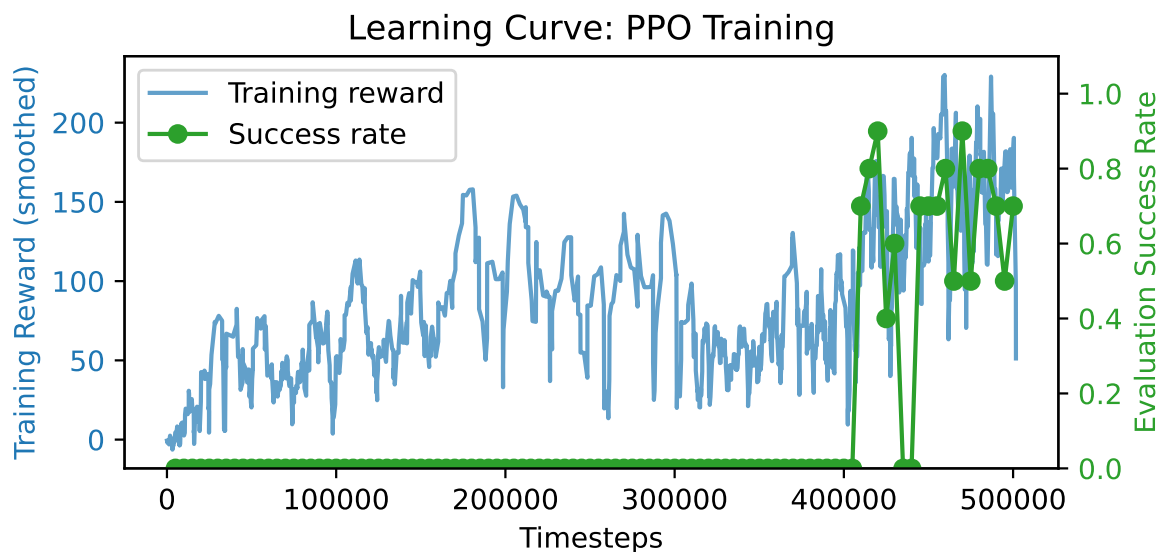
Szczegóły uczenia. Stosujemy algorytm PPO z następującymi hiperparametrami: tempo uczenia 3×10^{-4} , współczynnik dyskontowania $\gamma = 0, 99$, parametr GAE $\lambda = 0, 95$, parametr przycinania $\epsilon = 0, 2$, $T = 2048$ kroków na przebieg zbierania danych, rozmiar mini-partii (minibatch) 64 oraz 10 epok na aktualizację. Agent jest trenowany przez 5×10^5 kroków środowiskowych, co zajmuje około 2 godzin na standardowej stacji roboczej. Podczas uczenia początkowa pozycja sterowca jest losowo wybierana ze środowiska (bez kolizji), podczas gdy punkty pośrednie pozostają stałe (pojedynczy kurs składający się z 4 punktów pośrednich). Do oceny wykorzystano zarówno regulator PID, jak i regulator DRL z jednakowych pozycji startowych $(x_0, y_0, z_0, \theta_0) = (1.0, 1.0, 2.5, 0)$.

4. Układ eksperymentalny

W celu rzetelnego porównania regulatora PID z agentem DRL zdefiniowano spójny scenariusz testowy oraz protokół oceny. Eksperymenty symulacyjne zrealizowano w środowisku Gymnasium przy użyciu języka Python 3.10.

4.1. Scenariusz symulacji i protokół szkolenia

Środowisko testowe o wymiarach $10 \times 10 \times 5$ m zawiera trzy filary o różnej wysokości oraz jedną przeszkodę zawieszoną, co tworzy wąskie korytarze i wymaga



Rys. 1. Nagroda za epizod i wskaźnik sukcesu podczas uczenia uśrednione dla 10 ostatnich epizodów.

od kontrolera planowania ścieżek 3D. Misja polega na zaliczeniu czterech punktów orientacyjnych rozmieszczonych tak, aby linie proste między nimi przebiegały w pobliżu przeszkód.

Agent DRL był trenowany przez 5×10^5 kroków przy użyciu algorytmu PPO. Podczas uczenia pozycja startowa i kąt odchylenia sterowca były wybierane losowo, co pozwoliło agentowi na naukę generalizacji zachowań w różnych konfiguracjach początkowych. Proces ten trwał około 2 godzin a krzywą uczenia przedstawiono na rysunku 1.

4.2. Protokół oceny

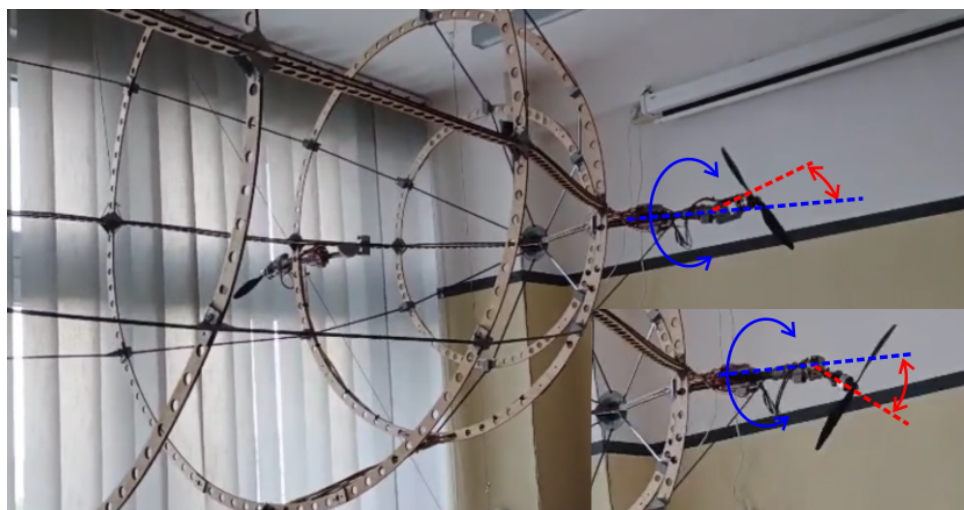
Oba regulatory oceniono w 50 niezależnych epizodach, startując z identycznego stanu początkowego $(x_0, y_0, z_0, \theta_0) = (1, 1, 2, 5, 0)$. Jako kluczowe wskaźniki efektywności przyjęto:

- **Współczynnik powodzenia:** odsetek misji zakończonych bez kolizji i w limicie czasu,
- **Długość trasy oraz liczbę kroków** potrzebnych do sukcesu,
- **Minimalną odległość LiDAR-u:** najmniejszy dystans do przeszkód zachowany podczas lotu.

Dodatkowo wyznaczono optymalną długość ścieżki w linii prostej (23, 25 m), stanowiącą punkt odniesienia dla efektywności obu metod.

4.3. Rzeczywista platforma sterowca

Aby potwierdzić praktyczne zastosowanie badań, opracowano fizyczny prototyp sterowca o długości 120 cm wypełniony helem. Gondola, wykonana techniką druku 3D i cięcia laserowego, zawiera Raspberry Pi 4, Arduino Nano, sensor LiDAR YDLidar X4 oraz jednostkę IMU BNO055. Zbieżność konfiguracji czujników rzeczywistego obiektu



Rys. 2. Rzeczywista platforma sterowca w trakcie budowy z przegubem śmigła typu 2R.

(rys. 2) z modelem symulacyjnym umożliwi przyszły transfer wypracowanych polityk sterowania w ramach techniki *sim-to-real*.

5. Wyniki

Oba regulatory oceniono w misji obejmującej cztery punkty orientacyjne, startując ze statycznego stanu początkowego (1, 0; 1, 0; 2, 5; 0) [6]. Wskaźniki ilościowe, stanowiące średnią z 50 epizodów, zestawiono w tabeli 1. Zarówno regulator PID, jak i agent DRL osiągnęły 100% wskaźnik sukcesu, co potwierdza skuteczność obu metod w zadanym środowisku [3].

Tabela 1

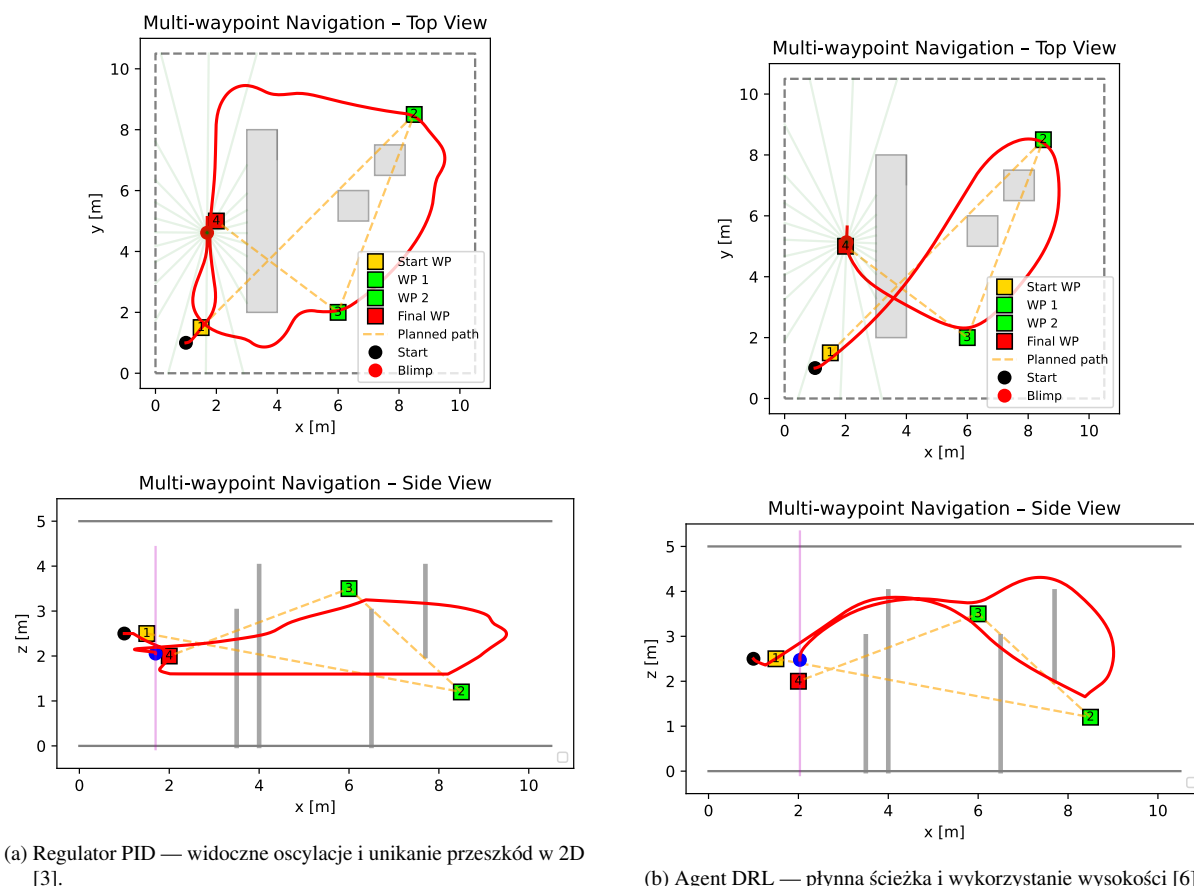
Porównanie jakości dwóch strategii sterowania (średnia z 50 epizodów).

Wskaźnik	PID	DRL (PPO)
Współczynnik sukcesu (%)	100,0	100,0
Średnia liczba kroków	427,0	243,0
Średnia długość ścieżki (m)	31,91	25,45
Optymalna ścieżka w linii prostej (m)		23,25
Średnia min. odległość LiDARu (m)	0,59	0,59

5.1. Analiza ilościowa i jakościowa

Mimo identycznej skuteczności, agent DRL wygenerował znacznie wydajniejsze trajektorie, wykonując średnio o 43% mniej kroków niż PID (243 vs 427). Średnia długość trasy DRL wyniosła 25,45 m, co jest wynikiem tylko o 9,5% gorszym od ścieżki optymalnej, podczas gdy PID pokonał trasę o 37% dłuższą od teoretycznego minimum. Obie metody zachowały bezpieczny margines od przeszkód (0,59 m).

Różnice w jakości wynikają z odmiennych strategii (rys. 3). PID wykazuje trudności z oscylacjami w pobliżu punktów docelowych oraz brakiem proaktywnego wykorzystania wysokości — omija przeszkody wyłącznie w płaszczyźnie *xy* [3]. Agent



Rys. 3. Porównanie reprezentatywnych trajektorii lotu sterowca w misji 3D.

DRL, wytrenowany algorytmem PPO, proaktywnie przelatuje nad niskimi przeszkodami, co znacznie skraca trasę.

Pod względem obliczeniowym obie metody są odpowiednie dla Raspberry Pi 4. PID zużywa znikomą ilość mocy procesora, natomiast inferencja sieci DRL zajmuje poniżej 0,5 ms [6, 10], co pozwala na sterowanie w czasie rzeczywistym.

6. Wnioski i dalsze badania

Badania potwierdziły, że DRL przewyższa klasyczny regulator PID pod względem jakości trajektorii w środowisku 3D dzięki zdolności do nieliniowej adaptacji i wykorzystania trzeciego wymiaru. Choć PID oferuje pełną interpretowalność i brak konieczności uczenia, DRL lepiej generalizuje się do złożonych układów przeszkód bez ręcznego strojenia.

Dalsze prace skupią się na:

- transferze polityki na fizyczny sterowiec (rys. 2) z wykorzystaniem *domain randomization* [7, 8],
- nawigacji w obecności przeszkód dynamicznych i ruchomych celów,
- fuzji danych z kamery monokularowej i wysokościomierza ultradźwiękowego [5],
- opracowaniu systemów hybrydowych łączących bezpieczeństwo PID z adaptacyjnością DRL [9].

Praca została zrealizowana przy wsparciu finansowym Politechniki Śląskiej w ramach subwencji na utrzymanie i rozwój potencjału badawczego (BK) w roku 2026 oraz przy wsparciu finansowym Fundacji ORLEN im. Ignacego Łukasiewicza w ramach programu „TURBOgrant” (DKN/00059/2025).

LITERATURA

1. Abbas M. T., Hanif F.: Design and control of an autonomous drone navigation system using embedded ai. *Journal of Engineering and Computational Intelligence Review*, 2025, Vol.3(2), p. 68–80.
2. Albaker B. M.: A deep reinforcement learning-based adaptive control strategy for uavs in dynamic and complex environments. *Al-Iraqia Journal for Scientific Engineering Research*, 2025, Vol.4(1), p. 77–88.
3. Fan Y., Cheng H., Wang H., Zhang F.: Multi-waypoint navigation control for robotic blimps using deep reinforcement learning. *2025 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2025, p. 1–6.
4. Kumar P., Pal K., Govil M. C.: Comprehensive review of path planning techniques for unmanned aerial vehicles (uavs). *ACM Computing Surveys*, 2025, Vol.58(3), p. 1–44.
5. Li J., Xiong X., Yan Y., Yang Y.: A survey of indoor uav obstacle avoidance research. *IEEE Access*, 2023, Vol.11, p. 51861–51891.
6. Liu Y. T., Price E., Goldschmid P., Black M. J., Ahmad A.: . Autonomous blimp control using deep reinforcement learning, *arxiv preprint arxiv:2109.10719*, 2021.
7. Liu Y. T., Singh N., Ahmad A.: Task and domain adaptive reinforcement learning for robot control. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, p. 656–663.
8. Neelesh M.: A reinforcement learning agent for uav control: Mathematical foundations, implementation, and human-vs-ai benchmarking. *International Journal of Trend in Scientific Research and Development*, 2025, Vol.9(2), p. 242–254.
9. Sarathi A., Gangadhara K. K. L.: Multi-vehicle guidance for decentralised swarm of drone using adaptive mpc, dnl, and rrt* for optimal path planning and obstacle avoidance. *2025 International Conference on Emerging Technology in Autonomous Aerial Vehicles (ETA AV)*, 2025, p. 1–6.
10. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O.: . Proximal policy optimization algorithms, *arxiv preprint arxiv:1707.06347*, 2017.
11. Wang X., Wang H., Fan Y., Cheng H., Zhang F.: . Bi-level reinforcement learning control for an underactuated blimp via center-of-mass reconfiguration, *arxiv preprint arxiv:2605.01289*, 2026.