Sterowanie autonomicznym bezzałogowym statkiem powietrznym z wykorzystaniem uczenia przez wzmacnianie

Paweł Miera, Hubert Szolc, Tomasz Kryjak
AGH Akademia Górniczo-Hutnicza im. S. Staszica, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Laboratorium Systemów Wizyjnych, Zespół Wbudowanych Systemów Wizyjnych, al. Mickiewicza 30, Kraków, 30-059, Polska

Streszczenie: Uczenie przez wzmacnianie ma coraz większe znaczenie w sterowaniu robotami, a symulacja odgrywa w tym procesie kluczową rolę. W obszarze bezzałogowych statków powietrznych (BSP, w tym dronów) obserwujemy wzrost liczby publikowanych prac naukowych zajmujących się tym zagadnieniem i wykorzystujących wspomniane podejście. W artykule omówiono opracowany system autonomicznego sterowania dronem, który ma za zadanie lecieć w zadanym kierunku (zgodnie z przyjętym układem odniesienia) i omijać napotykane w lesie drzewa na podstawie odczytów z obrotowego sensora LiDAR. Do jego przygotowania wykorzystano algorytm Proximal Policy Optimization (PPO), stanowiący przykład uczenia przez wzmacnianie (ang. reinforcement learning, RL). Do realizacji tego celu opracowano własny symulator w języku Python. Przy testach uzyskanego algorytmu sterowania wykorzystano również środowisko Gazebo, zintegrowane z Robot Operating System (ROS). Rozwiązanie zaimplementowano w układzie eGPU Nvidia Jetson Nano i przeprowadzono testy w rzeczywistości. Podczas nich dron skutecznie zrealizował postawione zadania i był w stanie w powtarzalny sposób omijać drzewa podczas przelotu przez las.

Słowa kluczowe: Uczenie przez wzmacnianie, Drony, Autonomiczne sterowanie, ROS, Gazebo

1. Wprowadzenie

Bezzałogowe statki powietrzne BSP (ang. *Unmanned Aerial Vehicle* – UAV), zwane również dronami, mają najczęściej formę samolotów lub wielowirnikowców. Charakteryzują się szeroką gamą potencjalnych zastosowań:
- inspekcja niedostępnych miejsc [1],
- transport drobnych przesyłek, na przykład szybkie dostarczanie krwi lub dostawy zaopatrzenia medycznego w trudno dostępne tereny [2],
- monitorowanie i analiza terenu na podstawie ortofotomapy utworzonej za pomocą zdjęć z kamery umieszczonej na dronie [3],
- monitorowanie natężenia ruchu ulicznego w mieście [4].

Większość z tych przypadków co najmniej w pewnym stopniu wykorzystuje automatyczny system sterowania. W najprostszym ujęciu polega on na locie kolejno do zadanych wcześniej punktów, których współrzędne są określone przez długość i szerokość geograficzną, kąt obrotu oraz wysokość. Wykonanie takiej misji wymaga ciągłego dostępu sterownika lotu do sygnału GPS (ang. *Global Positioning System*), który umożliwia uzyskanie globalnej pozycji pojazdu za pomocą danych dostarczanych przez satelity orbitujące wokół Ziemi. Dron może jednak znajdować się w zamkniętych pomieszczeniach, jak jaskinia [5], tunel, magazyn lub hala. Wówczas nie ma on dostępu do sygnału GPS. W takich miejscach można zastosować odpowiednie sensory do określania położenia. Umożliwiają one uzyskanie aktualnej pozycji względem punktu startowego, która jest obliczana na podstawie fuzji danych z czujnika inercyjnego i wizyjnego.

Obecnie w wielu pracach naukowych rozpatruje się sterowanie dronami za pomocą uczenia przez wzmacnianie (ang. *Reinforcement Learning*). Takie podejście polega na szkoleniu agenta do wykonywania określonego zadania. Trening ma na celu maksymalizację otrzymywanego zwrotu (ang. *return*) za wykonywanie działań w środowisku (najczęściej symulacyjnym). Wyszkolona strategia agenta (ang. *policy*) może być następnie przeniesiona do rzeczywistości i odpowiadać m.in. za sterowanie prędkościami drona w osiach X, Y i Z, także w warunkach braku sygnału GPS.

W artykule przedstawiono zastosowanie uczenia przez wzmacnianie do zadania nawigowania dronem na podstawie danych zwracanych przez obrotowy czujnik LiDAR. W tym celu zastosowano algorytm Proximal Policy Optimization

Autor korespondujący:
Tomasz Kryjak, tomasz.kryjak@agh.edu.pl

Artykuł recenzowany
nadesłany 31.10.2022 r., przyjęty do druku 09.11.2023 r.

(PPO). Przypadek testowy stanowił przelot w linii prostej (w zadanym układzie odniesienia) przez las, tak aby nie dochodziło do kolizji z drzewami. Agenta nauczono za pomocą specjalnie przygotowanego, prostego symulatora, a następnie zaimplementowano go w układzie Jetson Nano firmy Nvidia i przeprowadzono testy w realnych warunkach.

Efekty przeprowadzonych badań należy uznać za satysfakcjonujące. Zaproponowana metoda umożliwia otrzymanie skutecznego algorytmu sterowania dronem w środowisku leśnym, charakteryzującym się znacznym nagromadzeniem przeszkód w postaci drzew. Co więcej, jest ona relatywnie tania i przystępna, gdyż nie wymaga stosowania skomplikowanych środowisk symulacyjnych ani zaawansowanych algorytmów fuzji danych z różnych czujników.

W rozdziale 2 przedstawiono przegląd literatury naukowej związanej z poruszaną tematyką. Szczególną uwagę zwrócono na prace, w których wykorzystano uczenie przez wzmacnianie. W rozdziale 3 omówiono zaproponowaną metodę sterowania dronem. Zawarto w nim opis użytego środowiska symulacyjnego, szczegóły opracowanych agentów oraz specyfikację systemu sprzętowego, który wykorzystano do testów w rzeczywistości. Rozdział 4 prezentuje rezultaty uzyskane za pomocą zaproponowanej metody sterowania. Opisano w nim zarówno wyniki uczenia podczas symulacji, jak i efekty przeniesienia systemu do rzeczywistości. Ostatni rozdział 5 podsumowuje całość artykułu i wskazuje kierunki dalszego rozwoju.

2. Powiązane prace

Wykorzystanie uczenia przez wzmacnianie do zadania sterowania różnymi robotami stanowi obecnie bardzo popularne podejście. Jest to odzwierciedlone relatywnie dużą liczbą nowych prac naukowych dotyczących tego zagadnienia.

W artykule [6] przedstawiono porównanie wyników poszczególnych algorytmów uczenia przez wzmacnianie w różnych środowiskach. Zostały one podzielone na takie, w których występowało sterowanie dyskretne (wektor akcji może przyjmować tylko określone wartości) i ciągłe (wartości wektora akcji mogą być dowolne). W obu przypadkach algorytm PPO osiągał bardzo dobre wyniki, ustępując tylko metodzie SAC (ang. *Soft Actor Critic*) w środowiskach ciągłych, gdzie wymagał dłuższego czasu szkolenia w celu osiągnięcia dobrych rezultatów.

W artykule [7] opisano zastosowanie uczenia przez wzmacnianie do zadania bezpiecznej zmiany pasa przez samochód w sytuacji ruchu wielu pojazdów na drodze. W tym celu przygotowano symulację w narzędziu Unity oraz wyszkolono algorytmy PPO i SAC. Skuteczność obu algorytmów wynosiła ponad 90 %, ale czas szkolenia pierwszego z nich był znacznie krótszy.

Uczenie przez wzmacnianie znajduje również coraz szersze zastosowanie w dziedzinie dronów, co opisano w artykule przeglądowym [8]. Autorzy wykazują wzrost zainteresowania wykorzystaniem tych algorytmów do sterowania prędkością i wysokością lotu wielowirnikowców. Zwracają również uwagę na problemy związane z przenoszeniem takich systemów z środowiska symulacyjnego do warunków rzeczywistych. Są nimi – między innymi – ograniczenia sprzętowe oraz różnice między tymi dwoma środowiskami (ang. *sim-to-real*).

Artykuł [9] przedstawia przygotowanie systemu, którego misja polegała na wylądowaniu dronem na ruchomej platformie. Zadanie to zostało wykonane dzięki użyciu agenta RL, dla którego obserwacjami były aktualne stany pojazdu. Po przeprowadzeniu procesu szkolenia w środowisku symulacyjnym, algorytm został uruchomiony na stacji naziemnej, a sygnały sterujące były przesyłane przez sieć Wi-Fi. Skuteczność działania została potwierdzona zarówno w środowisku symulacyjnym, jak i w rzeczywistości.

Kolejnym przykładem wykorzystania uczenia przez wzmacnianie do sterowania dronami jest praca przedstawiona w artykule [10], w którym autorzy zaprezentowali sposób wyznaczania trajektorii przelotu drona przez bramki w zmieniającym się środowisku. Zadanie to zostało wykonane przez wyszkolenie agenta za pomocą algorytmu PPO w środowisku symulatora Fligthmare. Opracowana strategia była następnie testowana w trakcie tysiąca przelotów, a ostateczny współczynnik nieudanych prób dla najlepszej sieci wynosił 0,6 %. Jedna z wygenerowanych trajektorii została również uruchomiona na rzeczywistym dronie, jednak w trakcie przelotu wystąpiły duże błędy w śledzeniu pozycji.

Przedstawiona analiza pokazała, że algorytm PPO charakteryzuje się dobrymi wynikami oraz akceptowalnym czasem uczenia. Dlatego też zdecydowano się go wykorzystać w prowadzonych badaniach.

3. Zaproponowana metoda

Niniejsza praca koncentruje się na wykorzystaniu uczenia przez wzmacnianie do sterowania dronem. Rozpatrzony problem dotyczy przelotu przez nieznane środowisko przestrzenne, jakim jest las. Zaproponowaną metodę można podzielić na trzy etapy:

1. Przygotowanie środowiska symulacyjnego
2. Opracowanie agentów realizujących zadanie
3. Implementacja algorytmu sterowania w systemie sprzętowym.

Szczegóły realizacji każdego z etapów przedstawiono w dalszej części rozdziału. Założono przy tym, że system drona mającego omijać drzewa w lesie będzie otrzymywał informacje o otoczeniu za pomocą sensora RPLIDAR A2 [11]. Jest to urządzenie, które wykorzystuje czujnik laserowy zamontowany na obrotowej podstawie do mierzenia odległości. Wynikiem jego działania jest konturowy skan środowiska, składający się z odległości do najbliższych obiektów dla określonych kątów obrotu. Jego zasięg wynosi 16 m, zaś rozdzielczość kątowa 0,225°.

3.1. Środowisko symulacyjne

W celu jak najlepszego wyszkolenia agentów, zdecydowano się na wykorzystanie dwóch symulatorów. Pierwszy z nich, przygotowany od podstaw w języku Python, miał na celu wsparcie procesu uczenia. Drugi symulator został użyty do wprowadzenia korekt algorytmu, niezbędnych przy przeniesieniu systemu do układu sprzętowego. Tutaj zastosowano gotowe środowisko Gazebo Simulator.

Implementację pierwszego symulatora rozpoczęto od stworzenia klasy drona jako obiektu poruszającego się ruchem zmiennym. Metoda obliczenia przyspieszenia została przedstawiona we wzorze (1), a jego wartość ograniczono do zakresu $[-max\_acc, max\_acc]$. Aktualna prędkość i pozycja drona są obliczane odpowiednio ze wzorów (2) i (3) (analogicznie dla każdej osi prostokątnego układu współrzędnych).

$$a = \frac{v_{req} - v_0}{dt} \tag{1}$$

$$v = v_0 + a \cdot dt \tag{2}$$

$$x = x_0 + v \cdot dt + \frac{a \cdot dt^2}{2} \tag{3}$$

gdzie: a – przyspieszenie [m/s^2], v_{req} – prędkość zadana [m/s], v_0 – prędkość z poprzedniego kroku symulacji [m/s], dt – czas kroku symulacji [s], v – prędkość aktualna [m/s], x – pozycja aktualna [m], x_0 – pozycja z poprzedniego kroku symulacji [m].

Współczynniki w równaniach (2) i (3) dobrano tak, aby przy takim samym wymuszeniu zmiana pozycji i prędkości w opracowanym symulatorze drona była jak najbardziej zbliżona do wartości otrzymanych z Gazebo. Przykład uzyskanych w ten sposób rezultatów na wykresie dla osi Y przedstawiono na rysunkach 1a i 1b. Pomimo zauważalnych różnic, uzyskana dokładność okazała się wystarczająca do dalszych eksperymentów.

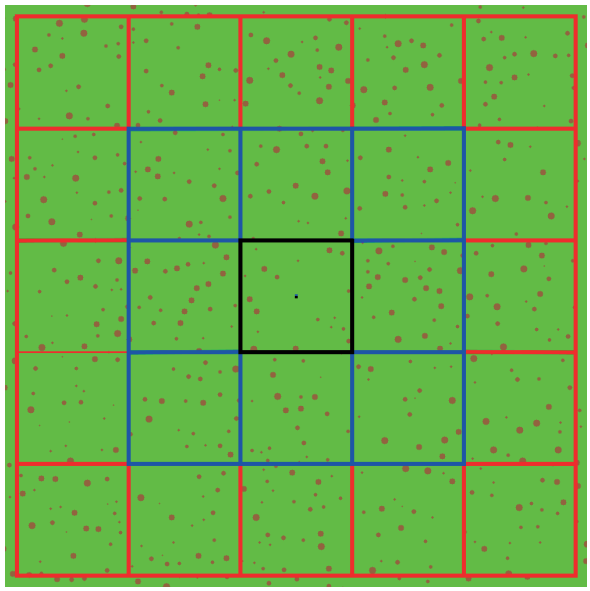
Istotną funkcjonalność opracowanego symulatora stanowi możliwość generowania wirtualnego lasu. Na rysunku 2 przedstawiono losową mapę z naniesioną siatką o boku długości 16 m. Do każdej z komórek zostały przypisane znajdujące się w niej drzewa. Ich atrybuty stanowią współrzędne położenia środka oraz promień konara. Podział na odrębne sektory zastosowano, aby w trakcie szukania drzew znajdujących się w zasięgu lasera, uwzględnić tylko te należące do aktualnej komórki drona (kolor czarny) oraz komórek sąsiadujących (kolor niebieski). Pozostała część siatki w kolorze czerwonym nie jest uwzględniana w danym kroku obliczeń, co pozwala na ich przyspieszenie.

Uruchomienie symulacji rozpoczyna się od inicjalizacji zmiennych i stworzenia obiektu drona z zerowymi wartościami pozycji i prędkości. Następnie w każdej komórce siatki losowo generowane są drzewa, z uwzględnieniem następujących parametrów:

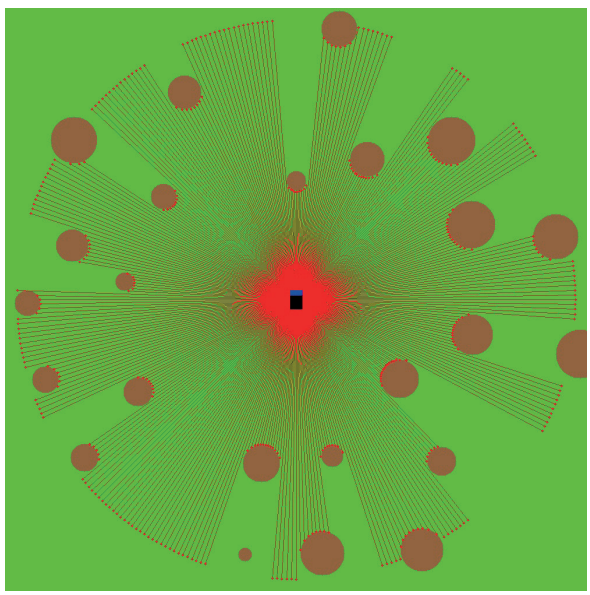
- `trees_per_grid` – określa liczbę drzew, które mogą się znajdować w jednej komórce siatki (wartość z przedziału 30–60, losowana po każdej iteracji);
- `tree_radius_range` – określa zakres promieni drzew w metrach;
- `trees_min_distance` – określa minimalną odległość między generowanymi drzewami.

Wartości współczynników dobrano tak, aby symulacja przypominała jak najbardziej rzeczywisty las. Jeśli oprogramowanie nie potrafiło znaleźć nowej lokalizacji dla kolejnej przeszkody przez 1000 prób, pętla była przerywana, przez co w aktualnej siatce znajdowało się mniej drzew.

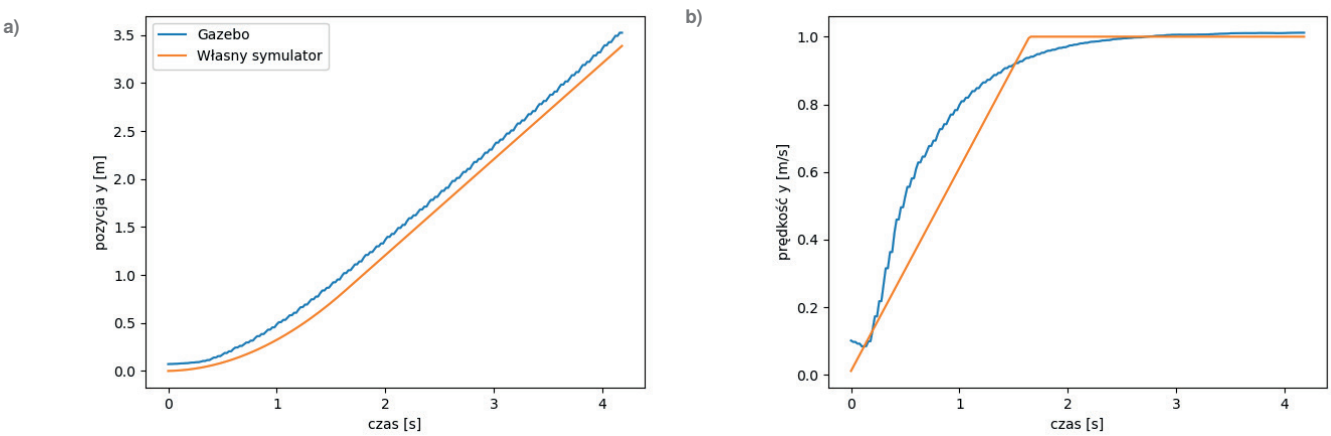
Przygotowano również symulację urządzenia RPLIDAR. Otrzymywane w jej ramach pomiary uzyskano przez znale-



Rys. 2. Mapa symulacji lasu z naniesioną siatką podziału na sektory
Fig. 2. Map of the forest simulation with the grid plotted



Rys. 3. Okno symulacji
Fig. 3. Display window of the prepared simulation



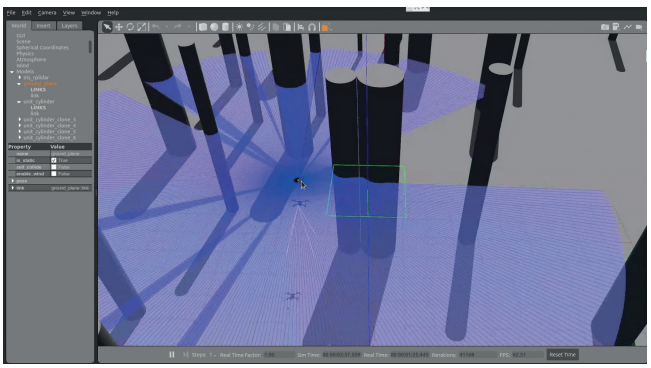
Rys. 1. Porównanie wartości uzyskanych w opracowanym symulatorze (kolor pomarańczowy) oraz w środowisku Gazebo: a) pozycja, b) prędkość wzdłuż osi Y
Fig. 1. Comparison of the values between our simulator (orange) and Gazebo environment: a) position, b) velocity along the axis Y

zienie punktów wspólnych równania prostej i okręgu. W tym celu wykorzystano fakt przechodzenia prostej przez punkt P0, którego współrzędne były równe (co do wartości) różnicy pomiędzy pozycją drona i środka drzewa. Opracowana implementacja wspomnianych równań w języku Python odrzuca również niechciane rozwiązania wynikające z przecięcia okręgu w dwóch punktach oraz tej samej wartości tangensa dla kątów przesuniętych o wielokrotność π.

Końcowy wygląd okna symulacji przedstawiono na rysunku 3. Przygotowany symulator został wykorzystany do wyszkolenia agentów RL (podrozdział 3.2). Jego kod źródłowy został także udostępniony w repozytorium Github[1].

Podczas przenoszenia algorytmów na platformę sprzętową wykorzystano również środowisko Gazebo. Sterowanie dronem odbywa się w nim poprzez komunikację z trybem SITL (ang. *software-in-the-loop*) oprogramowania PX4. Dzięki temu kod przygotowany w symulacji może działać bez zmian na rzeczywistym dronie. Całość jest również zintegrowana z systemem ROS (ang. *Robot Operating System*), co ułatwia komunikację między wszystkimi zastosowanymi sensorami.

W celu wykorzystania symulatora Gazebo, przygotowano specjalną mapę (rys. 4). Drzewa są na niej generowane jako wysokie walce o różnych promieniach. W przypadku drona i czujnika RPLIDAR, wykorzystano gotowe modele, dostępne wraz z oprogramowaniem PX4. W domyślnym pliku konfiguracyjnym zmieniono jednak niektóre parametry, takie jak rozdzielczość, częstotliwość próbkowania, wysokość względem podstawy oraz zasięg lasera.



Rys. 4. Przygotowana mapa do symulacji drona z sensorem RPLIDAR w środowisku Gazebo
Fig. 4. A map prepared to simulate a drone with the RPLIDAR sensor in the Gazebo environment

3.2. Opracowany agent RL

Do sterowania dronem wykorzystano algorytmy uczenia przez wzmacnianie. Ogólna zasada działania takiego podejścia polega na tym, że agent wykonuje pewną akcję w środowisku na podstawie otrzymywanych obserwacji i wartości nagrody. Równocześnie zmienia w ten sposób stan środowiska, co wpływa na jego kolejne decyzje. Agent podejmuje decyzje stosując opracowaną strategię, czyli funkcję, która na podstawie obserwacji zwraca określoną akcję. Jest ona zazwyczaj implementowana jako sieć neuronowa, której parametry są zmieniane w trakcie szkolenia. Architektura sieci może składać się z kilku warstw w pełni połączonych, jednak w niektórych środowiskach zastosowanie mają również warstwy rekurencyjne (są wyposażone w pamięć, przez co wcześniejsze obliczenia mają wpływ na aktualny wynik). Jeżeli obserwacja ma postać obrazu, model sieci neuronowej może zawierać w sobie również warstwy konwolucyjne oraz podpróbkujące (ang. *pooling*).

Najprostszym algorytmem uczenia przez wzmacnianie jest Q-learning. Wybór kolejnych realizowanych akcji odbywa się za pomocą tak zwanej „tablicy Q" (ang. *Q-Table*). Zawiera ona zwrot (zdyskontowana suma nagród), który zostanie potencjalnie otrzymany po wykonaniu danej akcji w konkretnym stanie i podejmowaniu dalszych decyzji, zgodnie z wykorzystywaną strategią. Jest to tzw. funkcja wartości akcji (ang. *action-value function*). Wartości komórek tablicy Q są aktualizowane iteracyjnie za pomocą równania Bellmana [12]. Q-learning oddziałuje na środowisko za pomocą innej strategii niż ta, która jest przez niego optymalizowana. Pewną modyfikację tego podejścia stanowi algorytm SARSA (ang. *State-Action-Reward-State-Action*). Tutaj jedna i ta sama strategia jest zarówno aktualizowana, jak i wykorzystywana do wykonywania akcji. Na podstawie omówionych podejść powstały popularne algorytmy uczenia przez wzmacnianie:

- Deep Q Network (DQN),
- Proximal Policy Optimization (PPO),
- Soft Actor Critic (SAC),
- Trust Region Policy Optimization (TPRO),
- Advantage Actor Critic (A2C).

Na podstawie przedstawionego przeglądu literatury, w pracy zdecydowano się wykorzystać algorytm PPO. Na początku założono, że celem drona jest przelecenie 30 m przez las wzdłuż osi X (w zadanym układzie odniesienia), bez kolizji z przeszkodami. Jedynymi informacjami, które otrzymywał agent PPO, były odczyty z obrotowego sensora LiDAR. Obserwacje nie zawierały żadnych danych o aktualnej prędkości ani pozycji drona. Były one przetwarzane przez obowiązującą agenta strategię, którą zaimplementowano jako sieć neuronową. W celu dopasowania obserwacji do przyjmowanego przez nią zakresu liczb $< -1; 1 >$, przeprowadzono normalizację zgodnie ze wzorem (4):

$$r = \left(r - h_2\right) / h_2 \tag{4}$$

gdzie: r – odległości otrzymane z obrotowego sensora LiDAR [m], h_2 – połowa maksymalnego zasięgu lasera [m].

Opracowano również funkcję nagrody, która stanowiła sumę następujących składników:

- kary za zbliżenie się do drzewa na odległość mniejszą niż 0,15 m — wartość: −0,25;
- kary za zderzenie się z drzewem — wartość: −1,5;
- kary za oddalanie się od linii środkowej środowiska (względem osi Y), której wartość wynosiła $-0{,}1 \cdot \|p_y\|$, gdzie p_y to współrzędna Y drona;
- nagrody za dużą prędkość w osi X (v_x), obliczana jako $0{,}8 \cdot v_x$ — jeśli dron leciał w złym kierunku, jej wartość była mnożona przez −3.

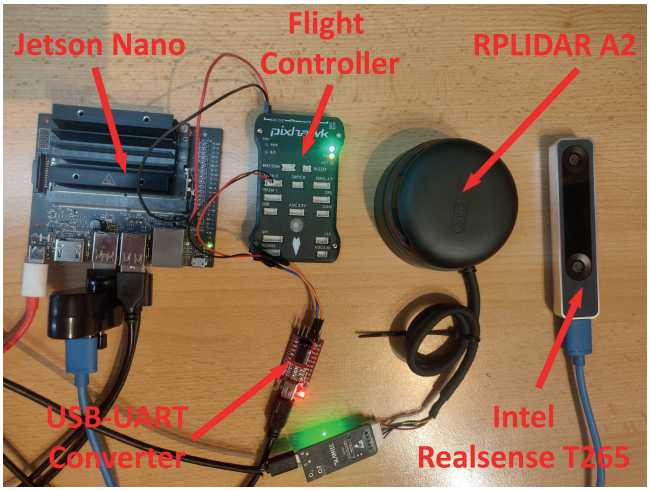
Wielowarstwowy perceptron MLP (ang. *multilayer perceptron*) został wykorzystany zarówno do realizacji algorytmu strategii, jak i obliczania funkcji wartości. W obu przypadkach składał się on z dwóch warstw ukrytych z aktywacją ReLU (ang. *rectified linear unit*). Ich wymiary wynosiły odpowiednio 128 neuronów dla strategii oraz 256 neuronów dla funkcji wartości.

Szkolenie agenta PPO odbywało się przez uruchomienie skryptu napisanego w języku Python. Za jego pomocą najpierw inicjalizowany był przygotowany symulator (opisany w podrozdziale 3.1) oraz model agenta RL, a następnie uruchamiany trening dla zadanej liczby epizodów.

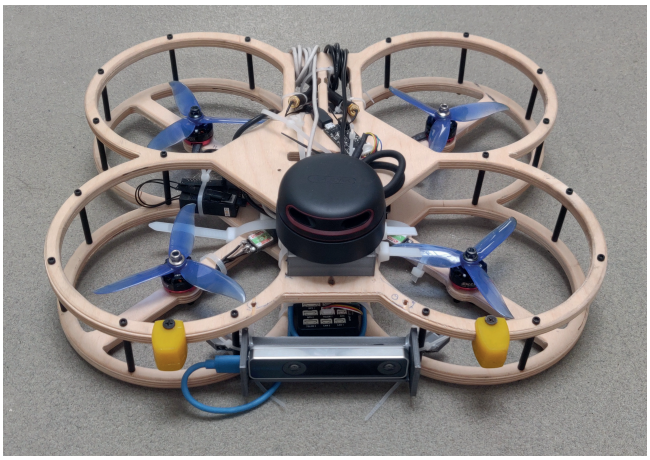
3.3. System sprzętowy

Po przeprowadzeniu procesu uczenia, gotowego agenta RL zaimplementowano w układzie sprzętowym Jetson Nano, który pełnił rolę komputera pokładowego. Komunikował on

1 https://github.com/vision-agh/python_drone_game_ laser_scan



Rys. 5. Komponenty sprzętowe zastosowane do sterowania dronem
Fig. 5. The hardware components used to control the drone



Rys. 6. Konfiguracja startowa drona wykorzystanego do realizacji eksperymentów w rzeczywistości
Fig. 6. Drone's launch configuration for the real world test scenarios

się z kontrolerem lotu Black Cube, na którym uruchomiono oprogramowanie PX4. Wymiana danych między tymi urządzeniami odbywała się za pomocą portu szeregowego z użyciem konwertera USB-UART. Wykorzystano ponadto wspominany już RPLIDAR A2, a także kamerę Intel Realsense T265. Ostatnie z wymienionych urządzeń miało przy tym dostarczać informację o aktualnej lokalizacji pojazdu wymaganą przez kontroler lotu Black Cube. Wszystkie komponenty komunikujące się z komputerem pokładowym zostały przedstawione na rysunku 5.

Podczas implementacji algorytmów sterowania wykorzystano systemy operacyjne Ubuntu 18.04 oraz ROS Noetic (który uruchomiono w specjalnie przygotowanym kontenerze Docker). Dzięki temu zapewniono wysoki stopień podobieństwa środowiska programistycznego na docelowej platformie sprzętowej względem zastosowanego symulatora Gazebo. Przyczyniło się to do ułatwienia całego procesu migracji.

4. Uzyskane wyniki

Zaproponowana metoda uczenia przez wzmacnianie do zadania sterowania dronem implikuje dwustopniowy proces weryfikacji. W pierwszej kolejności test jest realizowany na etapie symulacji, w trakcie uczenia agenta. Następnie opracowany system jest weryfikowany w rzeczywistym środowisku operacyjnym. Rezultaty obu etapów przedstawiono poniżej.

4.1. Proces szkolenia agenta

Podczas realizacji procesu szkolenia wykorzystano implementację algorytmu PPO w bibliotece *stable_baselines3* [13]. Obliczenia prowadzono na komputerach klasy PC, z których każdy wyposażony był w układ GPU Nvidia GeForce RTX 3060. Szkoleni agenci RL podlegali ewaluacji polegającej na wykonaniu 100 misji przelotu przez las wygenerowany za pomocą przygotowanego symulatora. Za każdym razem rozmieszczenie oraz wielkość drzew były losowane od nowa, aby sprawdzić algorytm sterowania na możliwie dużej liczbie przypadków. Lot uznawano za udany, gdy dron przebył zakładaną odległość 30 m, unikając po drodze kolizji z innymi obiektami. Najwyższa uzyskana skuteczność (zgodnie z przedstawionym kryterium) wyniosła 91 %.

Następnie przeprowadzono analogiczne eksperymenty przy wykorzystaniu symulatora Gazebo. Tutaj badano przede wszystkim, czy agent zachowuje się poprawnie w środowisku programowym zbliżonym do docelowego. Ze względu na statyczny charakter mapy ograniczono się do wykonania pięciu przelotów testowych. Wyniki eksperymentów potwierdziły poprawność przeprowadzonej implementacji – agent w sposób powtarzalny unikał kolizji z drzewami.

4.2. Loty w rzeczywistości

Do realizacji lotów w rzeczywistych warunkach wykorzystano czterowirnikowiec z ramą wykonaną z drewna. Cała konstrukcja charakteryzowała się względnie dużą masą, co skutkowało niskim stosunkiem ciągu do masy oraz krótkimi czasami lotów. Dron w konfiguracji startowej został przedstawiony na rysunku 6.

Loty testowe przeprowadzono w lesie, w miejscu jak najbardziej podobnym do warunków założonych w symulacji. Dron na swojej trasie mógł głównie napotkać drzewa z małą liczbą niskich gałęzi, a średniej wysokości krzewiny nie występowały zbyt często. Misje przeprowadzono z różnych punktów startowych w ten sposób, aby pojazd zawsze był kierowany w podobny obszar lasu. Celem misji było pokonanie zakładanej odległości wzdłuż osi X (w zadanym układzie odniesienia) bez uderzenia w żadne drzewo. Wszystkie testy dodatkowo nadzorowano za pomocą kontrolera radiowego, co umożliwiało manualne przejęcie kontroli w sytuacji awaryjnej.

Tab. 1. Wyniki lotów testowych zrealizowanych w rzeczywistości
Tab. 1. Results of the real-world test scenarios

Wszystkie przeloty	25	100 %
Bez uderzenia w drzewo	13	52 %
Kontynuowane po uderzeniu w drzewo	7	28 %
Zakończone po uderzeniu w drzewo	5	20 %

Łącznie zrealizowano 25 przelotów testowych. Ich wyniki przedstawiono w tabeli 1. Zdecydowana większość przelotów (80 %) zakończyła się sukcesem. Dron z powodzeniem przeleciał zadany dystans przez las, unikając po drodze większości napotykanych drzew. Należy podkreślić, że w żadnym przypadku konstrukcja drona nie uległa uszkodzeniu.

Sama skuteczność przelotów jest jednak zauważalnie niższa niż uzyskana w symulatorze. Stanowi to przykład problemu określanego w literaturze jako *sim-to-real gap*. Należy pamiętać, że podczas symulacji wykorzystano proste figury geometryczne o regularnym kształcie, które można uznać za uproszczony model drzewa. Mimo to agent RL był w stanie na tej podstawie zbudować wystarczająco generyczny obraz środowiska, który umożliwił wielokrotne skuteczne przeloty

Rys. 7. Klatka z filmu przedstawiającego jeden z przelotów testowych
Fig. 7. A frame from the video showing one of the real-world test flights

przez las. W tym świetle uzyskany rezultat należy uznać za satysfakcjonujący na obecnym etapie badań.

Na rysunku 7 przedstawiono zdjęcie z jednego z przeprowadzonych rzeczywistych przelotów. Film przedstawiający podsumowanie działania systemu można natomiast zobaczyć w serwisie YouTube[2].

5. Podsumowanie

W artykule przedstawiono zastosowanie algorytmów uczenia przez wzmacnianie w celu realizacji systemu sterowania dronem, który miał za zadanie autonomicznie lecieć w określonym kierunku i omijać napotykane przeszkody (drzewa) na podstawie odczytów z obrotowego sensora LiDAR. Podczas szkoleniu agenta RL wykorzystano algorytm PPO. W tym celu opracowano autorski symulator, napisany w języku Python, który w uproszczony sposób odwzorowywał środowisko leśne. Umożliwiał on generowanie losowych map, składających się z dowolnie rozmieszczonych drzew o odmiennych średnicach konarów. Strategię agenta oraz funkcję wartości zaimplementowano jako wielowarstwowe perceptrony. Opracowana funkcja nagrody umożliwiła ich pomyślne wyszkolenie, dzięki czemu agent uzyskał najlepszą skuteczność przelotu na poziomie 91 %.

Przy przenoszeniu systemu na rzeczywistą platformę sprzętową wykorzystano środowisko Gazebo. Przygotowano w nim imitację drzew w postaci wysokich walców o różnych promieniach. Nauczony agent został uruchomiony w tak opracowanym środowisku i był w stanie wykonywać skuteczne przeloty w sposób powtarzalny. Opracowany algorytm sterowania zaimplementowano w układzie Nvidia Jetson Nano i przeprowadzono testy na rzeczywistym dronie. W zdecydowanej większości prób pojazd skutecznie przeleciał przez las, niejednokrotnie unikając jakiegokolwiek uderzenia w drzewo.

Prezentowane rezultaty stanowią jedynie pewien etap prac związanych z wykorzystaniem algorytmów uczenia przez wzmacnianie do sterowania dronami. Tym niemniej są one obiecujące i uzasadniają kontynuację obranego kierunku badań. Dalszy rozwój tego podejścia zakłada przede wszystkim wykorzystanie bardziej zaawansowanych graficznie symulacji 3D. Ponadto planowane jest również zastosowanie sensorów innego typu, w szczególności kamer RGB lub głębi, a także czujników zdarzeniowych (neuromorficznych). Wśród potencjalnych usprawnień należy także wskazać możliwość wprowadzenia algorytmów „douczania" agenta już w trakcie realizacji misji w rzeczywistości oraz implementację całego systemu na innych platformach sprzętowych (np. heterogenicznych układach SoC FPGA) w celu przyspieszenia obliczeń i zmniejszenia zużycia energii.

2 https://www.youtube.com/watch?v=JqosupMgu7g

Bibliografia

1. Mandirola M., Casarotti C., Peloso S., Lanese I., Brunesi E., Senaldi I., *Use of UAS for damage inspection and assessment of bridge infrastructures*, "International Journal of Disaster Risk Reduction", Vol. 72, 2022, DOI: 10.1016/j.ijdrr.2022.102824.
2. Ackerman E., Koziol M., *The blood is here: Zipline's medical delivery drones are changing the game in Rwanda*, "IEEE Spectrum", Vol. 56, No. 5, 2019, 24–31, DOI: 10.1109/MSPEC.2019.8701196.
3. Carabassa V., Montero P., Crespo M., Padró J.-C., Pons X., Balagué J., Brotons L., Alcañiz J.M., *Unmanned aerial system protocol for quarry restoration and mineral extraction monitoring*, "Journal of Environmental Management", Vol. 270, 2020, DOI: 10.1016/j.jenvman.2020.110717.
4. Roldán J.J., Garcia-Aunon P., Peña-Tapia E., Barrientos A., *SwarmCity Project: Can an Aerial Swarm Monitor Traffic in a Smart City*?, 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2019, 862–867, DOI: 10.1109/PERCOMW.2019.8730677.
5. Koval A., Kanellakis C., Vidmark E., Haluska J., Nikolakopoulos G., *A Subterranean Virtual Cave World for Gazebo based on the DARPA SubT Challenge*, CoRR, abs/2004.08452, 2020, DOI: 10.48550/arXiv.2004.08452.
6. Loon K.W., Graesser L., Cvitkovic M., *SLM Lab: A Comprehensive Benchmark and Modular Software Framework for Reproducible Deep Reinforcement Learning*, arXiv, 2019. DOI: 10.48550/ARXIV.1912.12482.
7. Muzahid A.J., Kamarulzaman S.F., Rahman A., *Comparison of PPO and SAC Algorithms Towards Decision Making Strategies for Collision Avoidance Among Multiple Autonomous Vehicles*, 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), 2021, 200–205. DOI: 10.1109/ICSECS52883.2021.00043.
8. Jagannath J., Jagannath A., Furman S., Gwin T., *Deep Learning and Reinforcement Learning for Autonomous Unmanned Aerial Systems: Roadmap for Theory to Deployment*, arXiv, 2020, DOI: 10.48550/ARXIV.2009.03349.
9. Rodriguez-Ramos A., Sampedro C., Bavle H., de la Puente P., Campoy P., *A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform*, "Journal of Intelligent & Robotic Systems", Vol. 93, No. 1, 2019, 351–366. DOI: 10.1007/s10846-018-0891-8.
10. Song Y., Steinweg M., Kaufmann E., Scaramuzza D., *Autonomous Drone Racing with Deep Reinforcement Learning*, 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, DOI: 10.1109/IROS51168.2021.9636053.
11. Slamtec RPLIDAR-A2 Laser Range Scanner, [www.slamtec.com/en/ Lidar/A2].
12. Bellman R., *Dynamic programming*, Princeton University Press, 1957.
13. Raffin A., Hill A., Gleave A., Kanervisto A., Ernestus M., Dormann N., *Stable-Baselines3: Reliable Reinforcement Learning Implementations*, "Journal of Machine Learning Research", Vol. 22, 2021, 1–8. [http://jmlr. org/papers/v22/20-1364.html].

Control of an Autonomous Unmanned Aerial Vehicle Using Reinforcement Learning

Abstract: Reinforcement learning is of increasing importance in the field of robot control and simulation plays a key role in this process. In the unmanned aerial vehicles (UAVs, drones), there is also an increase in the number of published scientific papers involving this approach. In this work, an autonomous drone control system was prepared to fly forward (according to its coordinates system) and pass the trees encountered in the forest based on the data from a rotating LiDAR sensor. The Proximal Policy Optimization (PPO) algorithm, an example of reinforcement learning (RL), was used to prepare it. A custom simulator in the Python language was developed for this purpose. The Gazebo environment, integrated with the Robot Operating System (ROS), was also used to test the resulting control algorithm. Finally, the prepared solution was implemented in the Nvidia Jetson Nano eGPU and verified in the real tests scenarios. During them, the drone successfully completed the set task and was able to repeatable avoid trees and fly through the forest.

Keywords: reinforcement learning, RL, drones, automatic control, ROS, Gazebo

mgr inż. Paweł Miera
miera@student.agh.edu.pl

Absolwent kierunku Automatyka i Robotyka (specjalność: Inteligentne Systemy Sterowania) na Wydziale Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie. Jego zainteresowania naukowe obejmują bezzałogowe statki powietrzne (BSP), zwłaszcza ich systemy percepcji oraz sterowania przy wykorzystaniu algorytmów sztucznej inteligencji.

mgr inż. Hubert Szolc
szolc@agh.edu.pl
ORCID: 0000-0003-3018-5731

Doktorant Szkoły Doktorskiej AGH oraz asystent badawczo-dydaktyczny w Zespole Wbudowanych Systemów Wizyjnych, w Katedrze Automatyki i Robotyki, w Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie. Interesuje się algorytmami sterowania dla pojazdów autonomicznych na podstawie informacji wizyjnej, w tym danych z kamer zdarzeniowych. W swoich badaniach wykorzystuje heterogeniczne platformy obliczeniowe, zwłaszcza układy SoC FPGA do implementacji sprzętowej przygotowanych algorytmów.

dr inż. Tomasz Kryjak
tomasz.kryjak@agh.edu.pl
ORCID: 0000-0001-6798-4444

Adiunkt w Zespole Wbudowanych Systemów Wizyjnych, w Katedrze Automatyki i Robotyki, w Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie. Jego badania koncentrują się na wbudowanych systemach wizyjnych implementowanych w układach FPGA i SoC FPGA. Interesuje się systemami percepcji i sterowania pojazdów autonomicznych, kamerami zdarzeniowymi i obliczeniami neuromorficznymi. Jest członkiem IEEE (Senior Member), członkiem komitetów sterujących konferencji DASIP i DSD oraz członkiem komitetu technicznego konferencji ARC, a także redaktorem prowadzącym w czasopiśmie Microprocessors and Microsystems. Jest autorem lub współautorem ponad 120 prac naukowych.