Jędrzej Głowaczewski 175592
Miłosz Głowaczewski 175842
Szymon Gęsicki 175692

Sprawozdanie projektu z przedmiotu Sztuczna Inteligencja

Opis projektu

Tematem naszego projektu była symulacja drona posiadającego pewną “inteligencję”. Początkowym założeniem było stworzenie modułu SI który pozwoli na sprawne, samodzielne utrzymywanie stałego nachylenia względem podłoża. Dron posiada dwa silniki, z lewej oraz prawej strony podwozia. Każdemu silnikowi możemy nadać siłę z przedziału <0,1>. Naszym zadaniem było stworzenie modułu SI, który na podstawie dostarczonych danych opisujących aktualny stan pojazdu, określi odpowiednią moc dla obu silników.

Do zrealizowania tego zadania wykorzystaliśmy trzy różne metody:

- **Logika rozmyta** - jest to metoda wnioskowania zbliżona do sposobu rozumowania człowieka. Ta metoda pozwala na tworzenie intuicyjnych reguł opartych na spostrzeżeniach, które określają zachowanie sztucznej inteligencji, np.
    - *jeżeli dron jest mocno pochylony w lewo nadaj dużą moc dla lewego silnika.*
    - *jeżeli dron szybko obraca się w prawo nadaj dużą moc dla prawego silnika.*

  Pierwszym etapem jest fuzyfikacja danych wejściowych, czyli przypisanie określonych danych wejściowych do przedziałów rozmytych. Następnie przy użyciu wcześniej zdefiniowanych reguł określane są dane wyjściowe w postaci rozmytej. Ostatecznie dokonuje się defuzyfikacji, czyli określa się formalną wartość zmiennych wyjściowych.
- **Sieć neuronowa** - struktura matematyczna, składająca się z neuronów połączonych synapsami. Sieci neuronowe pozwalają realizować skomplikowane obliczenia, bez znajomości sprecyzowanego algorytmu. Sieci neuronowe są poddawane procesowi ‘trenowania’ czyli znalezieniu takiej konfiguracji wag przydzielonych do poszczególnych synaps, aby zwracały poprawne wyniki.
  Inspiracją sieci neuronowych była budowa naturalnych neuronów oraz układów nerwowych.
- **Regulator PID** - *regulator proporcjonalno-całkująco-różniczkujący* - regulator stosowany w układach regulacji składający się z trzech członów: proporcjonalnego, całkującego i różniczkującego. Najczęściej jego celem jest utrzymanie wartości wyjściowej na określonym poziomie, zwanym wartością zadaną.

Projekt został zaimplementowany w języku programowania Python.

Zastosowane biblioteki

- **pygame**: biblioteka do utworzenia okienkowej aplikacji w języku Python
- **pymunk**: biblioteka zajmującą się podstawową fizyką drona, dzięki niej mogliśmy zasymulować grawitację oraz inne siły działające na drona.
- **numpy**: biblioteka dodająca obsługę dużych, wielomiarowych tablic i macierzy, a także duży zbiór funkcji matematycznych wysokiego poziomu.
- **fpstimer**: pomocnicza biblioteka do określenia ilości klatek na sekundę aby na dowolnym komputerze symulacja działała w zbliżony sposób.
- **skfuzzy**: biblioteka do obsługi logiki rozmytej
- **tensorflow**: platforma do uczenia maszynowego
- **keras***:* biblioteka do obsługi sieci neuronowej

Jak korzystać z symulatora

Instalacja

- Na początku trzeba upewnić się że zainstalowany jest pythona w wersji 3. Niestety najnowsza wersja 3.8 nie jest kompatybilna z wszystkimi bibliotekami dlatego zaleca się używanie wersji python 3.7.x.
- Pobierz repozytorium z strony: https://bitbucket.org/SzymGesicki/dronesimulator/src/master/
- Otwórz konsolę w folderze pobranego repozytorium.
- Wszystkie potrzebne pakiety są spisane w pliku `requirements.txt`. Ich instalacja odbywa się przy pomocy polecenia:

```
pip install -r requirements.txt
```

Uruchomienie projektu

- W konsoli w otwórz folder `simulation`, zawarty w plikach projektu.
- Wybierz oczekiwany moduł AI. Numer wybranego modułu należy podać jako wartość parametru *-ai* podczas uruchamiania aplikacji.

Nazwa modułu	Nr modułu
Logika rozmyta	1 (domyślny)
Sieć neuronowa	2
Regulator PID	3
Brak AI	4

- Uruchomienie projektu:

  ➔ Windows
  `python main.py -ai 1`
  ➔ Linux/MacOs
  `python3 main.py -ai 1`

Sterowanie dronem

- **strzałka do góry** - nadanie tej samej siły w obu silnikach - ruch drona do góry.
- **strzałki lewo/prawo** - przechylenie drona w wybranym kierunku; Przy użyciu symulacji bez modułu AI strzałki odpowiadają za nadanie siły wybranemu silnikowi.
- **klawisz r** - restart drona do pozycji początkowej.
- **klawisz q** - wyjście z symulacji.

Końcowy efekt

Efektem końcowym jest dron z modułem AI odpowiadającym za utrzymywanie równowagi (pochylenia). Użytkownik ma możliwość sterowania pochyleniem docelowym, zmieniając w ten sposób pozycję drona. Istnieje możliwość wyboru modułu logiki drona:

- logika rozmyta,
- sieć neuronowa,
- logika oparta na regulatorze PID,
- brak AI.

Co stały, określony czas do wybranego modułu SI podawane są wartości wejściowe opisujące aktualny stan drona. Moduł oblicza zalecane wartości mocy dla obu silników, które zostają zwrócone jako dane wyjściowe. Wartości zwrócone przez kontroler są następnie ustawiane w obu silnikach.

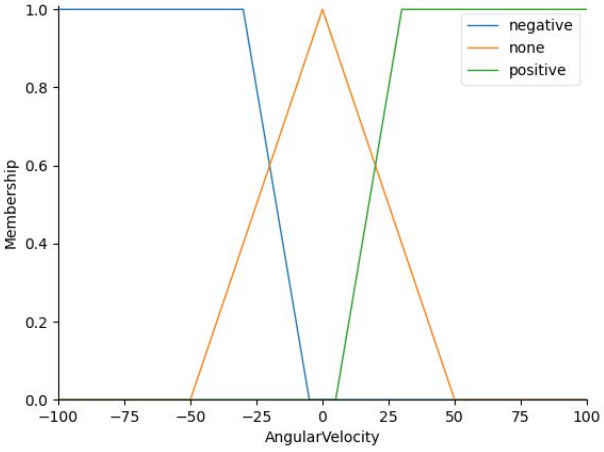
Dzięki modułowi SI dbającym o utrzymanie stałego pochylenia, sterowanie dronem jest znacznie ułatwione.Używając strzałki do góry dron nadaje równą moc dla obu silników. Używając bocznych strzałek zmieniamy docelowe pochylenie drona, w ten sposób korygując kierunek jego lotu.

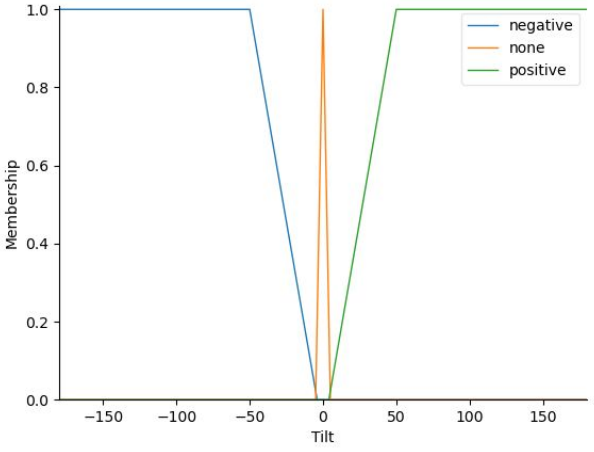
Istnieje również możliwość całkowitego wyłączenia SI. W tej wersji symulacji boczne strzałki odpowiadają za moc poszczególnych silników.

Logika rozmyta

Moduł SI oparty na logice rozmytej został stworzony przy użyciu biblioteki skfuzzy. Kontroler jako dane wejściowe przyjmuje kąt pochylenia oraz przyspieszenie kątowe. Obie wejściowe wartości liczbowe są modelowane na następujące funkcje przynależności:

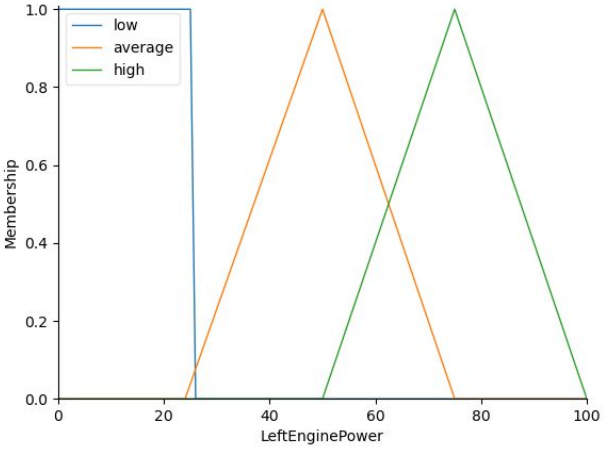
- “negative” - dron pochyla się w lewą stronę (prędkość / nachylenie są ujemne),
- “none” - wartość równa lub bliska zeru,
- “positive” - dron pochyla się w prawą stronę (prędkość / nachylenie są dodatnie).

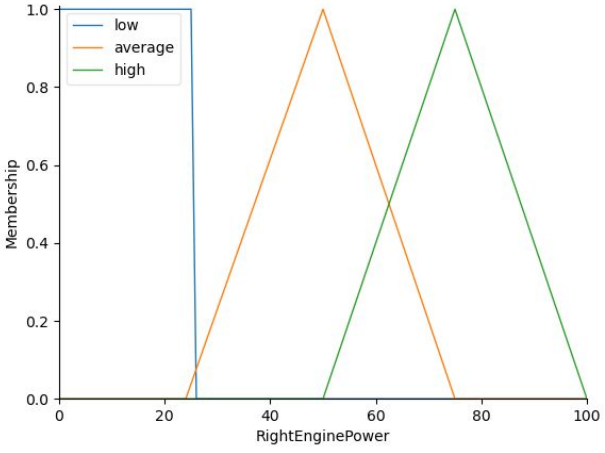




Dla obu silników modelowane są identyczne funkcje przynależności:

- “low” - moc zerowa lub bliska zeru,
- “average” - średnia moc,
- “positive” - duża moc.





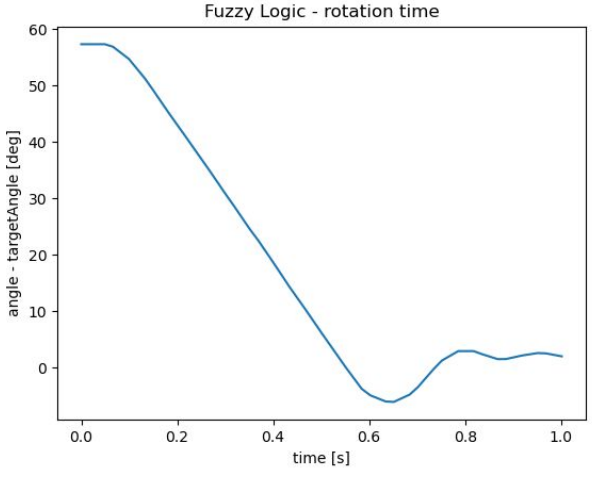
Następnie stworzone zostały reguły określające dla jakiego kąta nachylenia oraz prędkości kątowej, jaka jest oczekiwana moc każdego z silników.

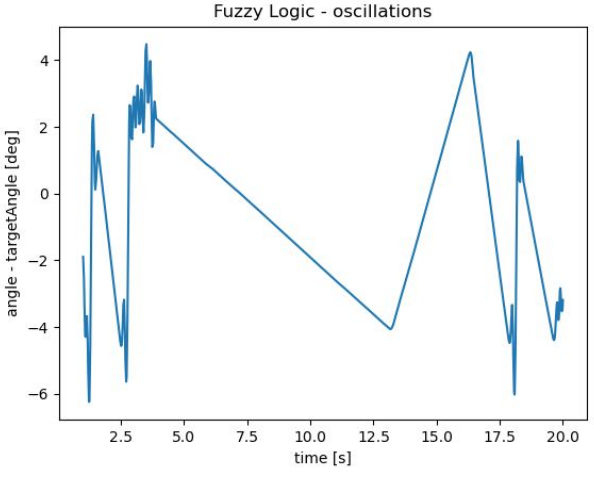
INPUT		OUTPUT	
Tilt	**Angular Velocity**	**Left Engine Power**	**Right Engine Power**
Positive	Positive	High	Low
	None	Average	Low
	Negative	Low	Low
None	Positive	Average	Low
	None	Low	Low
	Negative	Low	Average
Negative	Positive	Low	Low
	None	Low	Average
	Negative	Low	High

Funkcje przynależności oraz reguły zostały dobrane w sposób eksperymentalny “na intuicję”. Dlatego efekty tej implementacji nie są idealne. Takie podejście - naturalne dla logiki rozmytej - pozwoliło na stosunkowo szybką implementację bez potrzeby specjalistycznej wiedzy. Prawdopodobnie bardziej profesjonalne analizy wyników przejściowych, poparte prawami i wzorami fizycznymi, pozwoliłyby na znacznie lepsze wyniki.

Wyniki

Dron korzystając z modułu opartego na logice rozmytej nie zapewnia idealnej stabilności. Występują bardzo nieregularne oscylacje o zmiennym okresie i amplitudzie. Mimo to sterowaniem dronem jest komfortowe. Czas obrotu drona o 60º to ok. 0,5-0,6 sekundy.





Jędrzej Głowaczewski 175592
Miłosz Głowaczewski 175842
Szymon Gęsicki 175692

Regulator PID

Regulator PID pracuje w pętli sprzężenia zwrotnego, oblicza wartość uchybu jako różnicę pomiędzy pożądaną wartością zadaną i zmierzoną wartością zmiennej procesu i działa w taki sposób, by zredukować uchyb poprzez odpowiednie dostosowanie sygnału podawanego na wejście regulowanego obiektu.

Algorytm obliczeń regulatora PID zawiera trzy oddzielne stałe parametry i dlatego czasami bywa nazywany regulatorem z trzema członami: proporcjonalnym, całkującym i różniczkującym, oznaczonymi odpowiednio P, I i D.

Poglądowo działanie tych członów w odniesieniu do czasu można zinterpretować następująco:

- działanie członu P kompensuje uchyb bieżący,
- człon I kompensuje akumulację uchybów z przeszłości,
- człon D kompensuje przewidywane uchyby w przyszłości.

Do implementacji został wykorzystany prosty wzór:

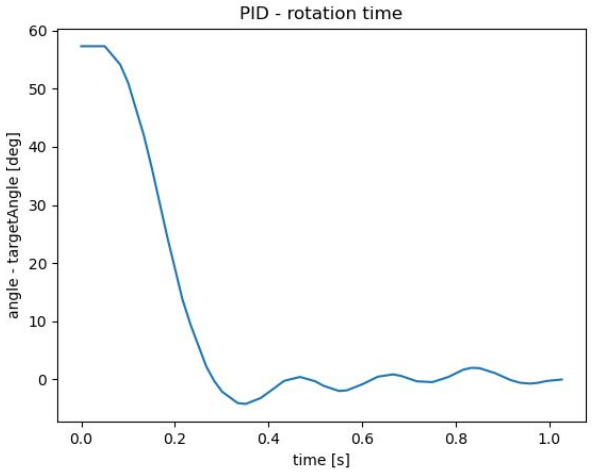
$$\text{siła} = K_p * (\text{kąt\_docelowy} - \text{kąt\_bieżący}) + K_d * \text{prędkość\_kątowa}$$

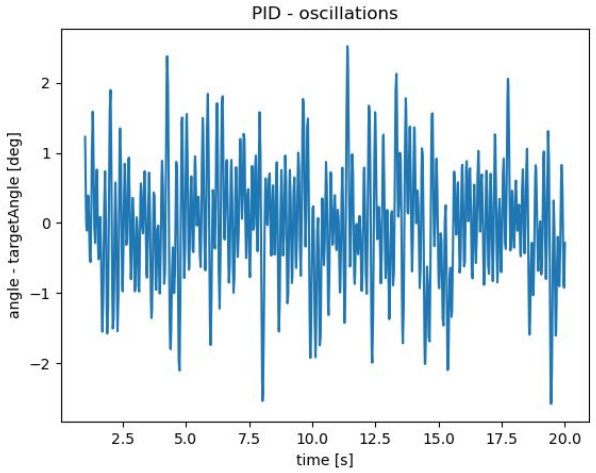
Stałe K_p, K_d zostały dobrane w sposób eksperymentalny “na intuicję”. Wyglądają następująco.
$K_p = 65$
$K_d = -9$

Zrezygnowaliśmy z implementacji członu różniczkującego, ponieważ dron stabilnie utrzymywał równowagę korzystając tylko z dwóch pozostałych członów.

Wyniki

Dron przy tej metodzie radzi sobie bardzo dobrze. Oscylacje są szybkie i mają małą amplitudę. Czas obrotu o 60° jest dwukrotnie krótszy niż przy wykorzystaniu logiki rozmytej. Algorytm jest prosty w implementacji i niewymagający pod kątem mocy obliczeniowej dlatego mógłby być implementowany w prawdziwych dronach w których nie mamy dostępu do szybkich procesorów.





Jędrzej Głowaczewski 175592
Miłosz Głowaczewski 175842
Szymon Gęsicki 175692

Sieć neuronowa

Za pomocą biblioteki Tensorflow, przy wykorzystaniu Keras, została stworzona i wytrenowana sieć neuronowa w opisany poniżej sposób:

- Wygenerowanie danych:
  - Wylosowanie N pozycji drona, (kąt, prędkość kątowa)
  - Podanie wylosowanych pozycji do modułu logiki rozmytej,
  - Obliczenie odpowiedniej reakcji.
  - Zapisanie do pliku csv symetrycznych akcji, plik zawiera:
    - oczekiwaną siłę lewego silnika,
    - oczekiwaną siłę prawego silnika,
    - aktualną pozycję,
    - prędkość kątową drona
- Trenowanie drona
  - Wczytanie do pamięci wygenerowanych danych
  - Znormalizowanie danych, do wartości <0,1>
  - Podzielenie danych na dwie części - do trenowania(75%), do testowania(25%)
  - Obliczenie dokładności na testowych danych

Podczas lotu drona, w stałych odstępach czasu, zostaje odczytana wartość pozycji i prędkości kątowej drona. Następnie zostaje podana do modelu sieci neuronowej który przewiduje siłę lewego i prawego silnika.

Sieć neuronowa składa się z 3 warstw wewnętrznych gęstych, aktywowanych funkcją ‘relu’. Warstwy zawierają po 8 neuronów.

Jako funkcja straty został zastosowany średni błąd kwadratowy.
Do optymalizacji sieci został wykorzystany algorytm NAdam.
Dokładność sieci neuronowej po zakończonym treningu była sprawdzana za pomocą funkcji średniego błędu.

Nauka ostatecznej sieci odbyła się w 5 epokach w paczkach o wielkości 16, na danych testowych o wielkości 20 000 rekordów.
Zastosowana konfiguracja została wybrana, ze względu na bardzo dobre wyniki, przy jednocześnie krótkim czasie nauki.

Wyniki

Sieć neuronowa, po zakończonym treningu, potrafiła z bardzo wysoką dokładnością odwzorować działanie regulatora PID oraz logiki rozmytej.

Po uczeniu na bazie algorytmu PID wykresy oscylacji i obrotu są niemal identyczne z tymi z oryginalnego algorytmu. Dron idealnie zachowuje równowagę. Jednak przy szybkich

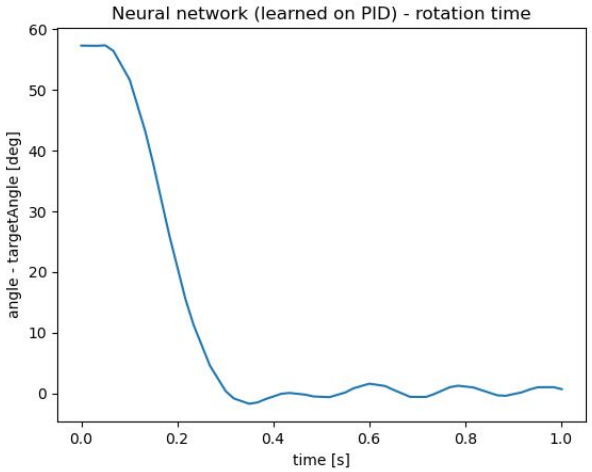
zmianach kierunku dron potrafi obrócić się wokół własnej osi. Wynika to prawdopodobnie z przycięcia wartości wejściowych.

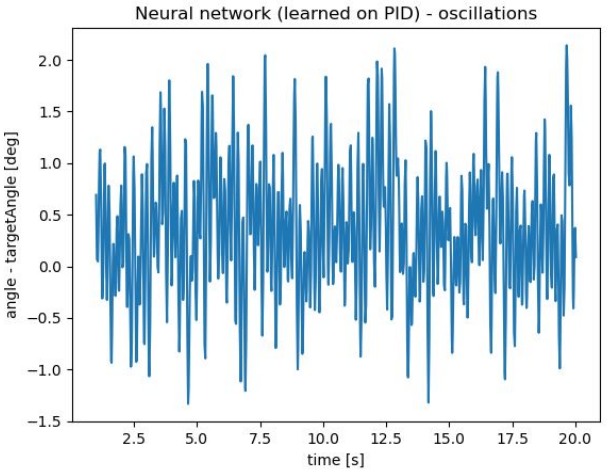
Po uczeniu na bazie logiki rozmytej, podczas sterowania dronem, rozróżnienie czy dron korzysta z sieci neuronowej, czy z modułu na którym sieć była trenowana było prawie niemożliwe.

W plikach projektu zostały zamieszczone wytrenowane modele sieci neuronowej.

- ‘data/drone_model’ - model wytrenowany na podstawie działania regulatora PID
- ‘data/drone_model_fuzzy’ - model wytrenowany na podstawie działania modułu logiki rozmytej

Moduł można ustawić w pliku NeuralNetworkAI.py. Domyślnie ustawiony jest model wytrenowany na bazie algorytmu PID.





Wnioski

Po obserwacjach możemy stwierdzić że najlepsze wyniki dostarczył algorytm PID. Był prosty w implementacji, dał bardzo dobre wyniki i wymagał niewielkiej mocy obliczeniowej.
Algorytm oparty na logice rozmytej dostarczył gorszych wyników i był trudniejszy w implementacji. Sieć neuronowa może dostarczyć wyników niemal dowolnej jakości, ale uzależnione jest to od jakości danych na których przebiega trenowanie. Była trudna w implementacji i wymaga stosunkowo dużej mocy obliczeniowej.
Bardzo ważnym atutem algorytmu PID jest prostota implementacji, ponieważ pozwala to na użycie tego algorytmu na mikrokontrolerach, które nie posiadają możliwości obsłużenia trudniejszych programów korzystających z dużych bibliotek, wymagających dużej mocy obliczeniowej. Szybkość obliczeń pozwala na częstsze aktualizacje stanu drona, co w rzeczywistych warunkach może okazać się kluczowe. Dłuższe wykonywanie obliczeń przez dwa pozostałe algorytmy może powodować niewystarczającą szybkość reakcji na zmienne warunki pogodowe lub inne sytuacje losowe.