STABILIZAREA UNEI DRONE FOLOSIND A.I.

Cuprins:

Introducere:

In acest proiect se urmărește stabilizarea unei drone în două dimensiuni (XoY) împotriva diverselor intemperii precum condiții de vânt schimbătoare, gravitație și inerție, folosind regresie liniară.

Metodologie:

Biblioteci folosite:

- Pygame - pentru crearea ferestrei şi a environment-ului
- Pymunk - pentru implementarea de elemente de fizică precum forţe, obiecte şi pereţi
- Random – pentru generarea vântului aleator
- Math – pentru elemente de trigonometrie (mai ales in afișarea direcției vântului)
- Numpy – pentru array-uri şi preluarea datelor din fişier
- Tensorflow – pentru algoritmii de ML şi elemente ajutătoare în calcularea erorii şi acurateţii.

Fișiere cuprinse în proiect:

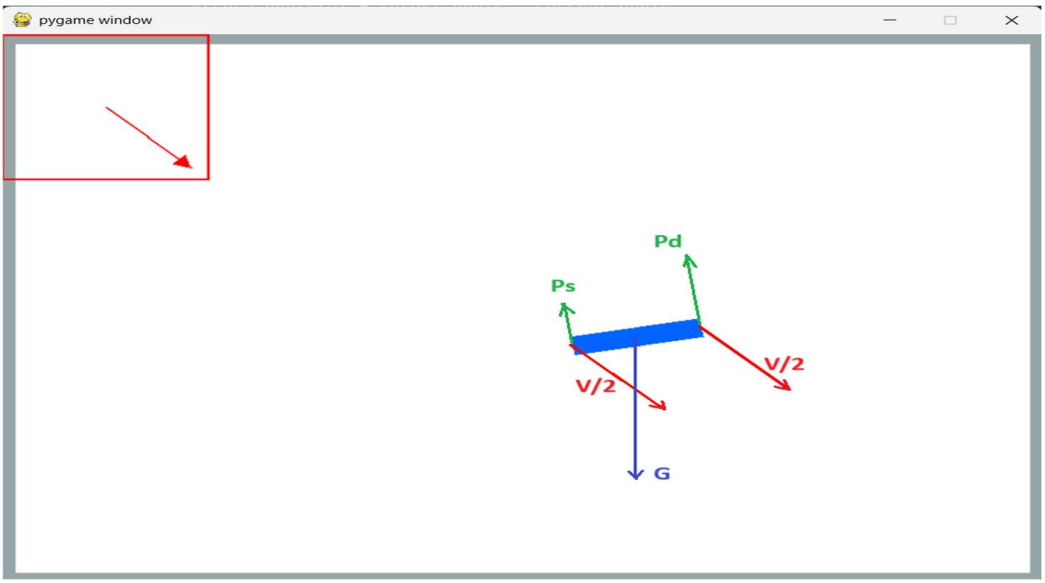
- PID_Environment.py – ruleaza environment-ul cu drona stabilizata de PID_controller, lăsând user-ul să o deplaseze.
- AI_Environment.py – un fisier similar celui anterior, cu singura diferență constituind-o faptul că drona este acum stabilizată de modelul de regresie + algoritmi de calculare a acuratetii.

- Data.csv – fisierul de date care contine aproximativ 3000 de linii si 11 coloane
- (variantă alternativă)model_train.py – fișierul care se ocupă de antrenarea alternativei de model cu SGD

Environment:

Pentru a desfășura proiectul, am generat un obiect dreptunghiular (considerat dronă) cu o masă de 8kg și câte o forță perpendiculară pe dronă aplicată la fiecare capăt, reprezentând propulsoarele.

Poziția "target" urmărită de dronă, poate fi modificată prin apăsarea tastelor (W, A, S, D), iar vântul poate fi oprit/pornit prin apăsarea tastei V.



De asemenea, am adăugat o funcție pentru simularea unor condiții de vânt care sunt generate aleator ca direcție și intensitate. Cadranul din stânga-sus, prezent pe întreg parcursul rulării, îmfățișează prin săgeți direcția și intensitatea menționate mai sus. Ele se schimbă odată la 5 secunde.

PID Controller:

Pentru alcătuirea unui set de date aproximativ corect, pe care se poate realiza antrenarea unui model A.I., a fost nevoie, mai întâi de folosirea unui Controller Proportional, Integrator, Derivativ. PID Controller-ul este un algoritm non-A.I., care folosește constant eroarea față de poziția urmărită, pentru a genera corecții asupra forțelor aplicate la motoarele dronei.

În acest scop, am relizat un total de 3 PID Controller-e (unul pentru corecția erorii pe abscisă, unul pe ordonată și unul pentru unghiul dronei față de orizontală, fiecare pentru ambele motoare).

Cunoscând faptul că integrala în discret, poate fi considerată o sumă, iar derivata este diferența între starea finală și cea inițială, am creat câte un vector de erori pentru fiecare dimensiune studiată (errors_x, errors_y, errors_angle), fiecare conținând câte 3 erori: cea curentă (diferența între poziția urmărită și cea actuală a dronei), cea din urmă cu un ciclu de ceas și cea din urmă cu doi cicli de ceas. Acestea ne dau un total de 9 valori de eroare ca input-uri, din care putem genera forțele necesare la fiecare moment pentru a urmări stabilitatea în punctul de referință (target_position).

Setul de date și regresia:

Setul de date este alcatuit din aproximativ 6000 de linii si 11 coloane, primele 9 reprezentând input-ul (cele 9 valori de eroare), iar ultimele 2, output-ul (cele 2 forțe aplicate la propulsoare). Pentru ușurința lucrului cu datele, ele sunt normalizate in intervalul [-1, 1].

Modelul folosește regresie liniara pentru a aproxima o funcție cat mai aproape de valorile din date, apoi, cu aceste aproximări, reușește să stabilizeze drona în fața unor condiții noi (random).

Rezultate obținute:

Folosind algoritmul simplu LinearRegression(), calitatea rezultatelor obținute este acceptabilă, întrucât, obținem niște valori relativ mici ale Mean Squared Error. Un exemplu după o rulare este mai jos:

```
MSE Left Thruster: 0.02641535742179177, MSE Right Thruster: 0.03286779435585731

Process finished with exit code 0
```

În ceea ce privește R2, am obținut valori de puțin peste 0.5, ceea ce se poate interpreta prin faptul ca modelul are o performanta moderata dar nu este perfect si mai exista factori care influențează variabilele si nu sunt capturați de model. Acest lucru se poate datora și unei rulări insuficient de lungă pentru a verifica mai multe condiții.

```
MSE Left Thruster: 0.044123543767485236, MSE Right Thruster: 0.047060222149839716
R2 Left Thruster: 0.5065375811462858, R2 Right Thruster: 0.5299094181925029

Process finished with exit code 0
```

În urma unei încercări de a folosi un model de algoritm de optimizare (Stochastic Gradient Descend), am obținut performanțe ușor mai ridicate. Am atașat și un fișier numit train_model.py în arhivă care se ocupa de antrenarea acestui model pe aceleași date. Rezultatele sunt următoarele:

```
MSE Left Thruster: 0.024008913628138447, MSE Right Thruster: 0.03154529636609863
R2 Left Thruster: 0.6279835861090071, R2 Right Thruster: 0.5726268518707465
```

Detrimentul în folosirea acestui algoritm este o scădere monumentală în FPS-uri, fapt care afectează negativ modelul, în mediu practic, din pricina fizicii programului și a parametrilor PID modificați pentru 60FPS.

Concluzii:

Având în vedere faptul că datele de antrenare nu sunt nici ele perfecte, întrucât până și algoritmul original de controller PID dă greș în anumite cazuri, iar parametrii K_p, K_d, Ki necesită perioade lungi de ajustare cu tehnica ”trial and error”, aș putea spune că rezultatele sunt mulțumitoare în ceea ce privește modelul care aproximează un rezultat în urma acestor date.

Vă mulțumesc pentru atenția acordată proiectului meu!