Návrh riadenia lietajúcich dronov pomocou

Neuroevolúcie

Aleš Melichar1

1 Ústav vedy a výskumu, FEI STU v Bratislave

2 Ústav Robotiky a kybernetiky

xmelichar@stuba.sk

Abstrakt – Táto práca sa zaoberá návrhom algoritmu riadenia pre UAV/kvadrokoptéru s cieľom dosiahnuť požadovanú pozíciu statického cieľa v trojrozmernom priestore. Využíva neuro-evolúciu s rozširujúcou topológiou (NEAT), umožňujúcu autonómnu navigáciu pomocou neurónových sietí riadených evolučným algoritmom. Realizácia práce prebiehala v hernom vývojovom programe Unity, v ktorom bolo vyvíjané simulačné prostredie. Na implementáciu riadiacich algoritmov sa využívala voľne dostupná knižnica Neat-python.

# Úvod

Drony sa stávajú čoraz dôležitejšou súčasťou moderného sveta. Ich využitie siaha od doručovania zásielok a monitorovania infraštruktúry až po záchranné akcie a pomoc pri spracovaní obilnín v poľnohospodárstve. Kľúčovou vlastnosťou dronu je jeho schopnosť presne sa navigovať v trojrozmernom priestore a dosiahnuť zadaný cieľ [1].

Cieľom tejto práce je navrhnúť algoritmus riadenia pre UAV/kvadrokoptéru, ktorej úlohou je dosiahnuť požadovanú pozíciu statického cieľa. Keďže pri riadení kvadrokoptéry je náročne definovať správne rýchlosti rotorov v konkrétnom stave, nie je možné použiť učenie s učiteľom. Preto sme sa rozhodli riešiť problém evolučnými technikami učenia. Algoritmus je vytvorený využitím neuro-evolúcie s rozširujúcou topológiou (NEAT – Neuro-evolution of augumented topologies).

Dron sa pohybuje v 3D priestore, v ktorom vníma polohu cieľa a snaží sa ho dosiahnuť. V práci sa zameriame na metódu autonómnej navigácie drona, ktorá využíva neurónovú sieť modifikovanú evolučným algoritmom.

V rámci Unity je vyvíjané simulačné prostredie, ktoré zabezpečuje vizualizáciu pohybu kvadrokoptéry a interakciu s prostredím. Na strane implementácie algoritmov riadenia a učenia je využívaná knižnica Neat-python, ktorá poskytuje prostriedky na vytvorenie a evolúciu neurónových sietí [2].

Toto prostredie umožňuje integráciu neurónových sietí do simulácie v Unity, avšak prepojenie a trénovanie neurónových sietí pomocou evolučných algoritmov vyžaduje dodatočné techniky a implementáciu komunikačných kanálov na zdieľanie informácií do a z prostredia.

# Dron

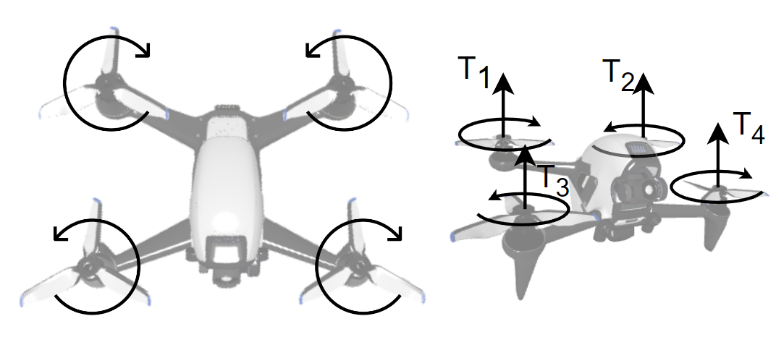
Bezpilotné lietajúce prostriedky (ďalej ako UAV), často označované ako drony, sú letecké vozidlá, ktoré lietajú autonómne alebo s diaľkovým ovládaním, pričom sa na palube nenachádza ľudský operátor [1].

Rozpoznávame viacero typov UAV, a to:

* UAV s pevným krídlom. Tieto UAV sa používajú v úlohách, kedy je potrebné prekonať dlhé vzdialenosti a pokryť väčšie plochy [3].
* UAV viac-rotorové/s rotačným krídlom. Sú vhodné do aplikácií, kedy je potrebné státie na mieste napríklad pri inšpekcií elektrických vedení [3].
* Hybridné UAV, ktoré sú kombináciu pevného krídla a rotorov. Podobne ako viac-rotorové UAV majú možnosť vertikálneho vzletu a pristátia, ako aj dlhú dĺžku letu [3].

V tejto práci použijeme viac-rotorové UAV a to konkrétne so 4 rotormi. Takýto dron teda pozostáva zo skeletu (drak) a štyroch rotorov, ktoré sú umiestnené rovnomerne od stredu tela, ktorý sa zhoduje s jeho barycentrom. Rotory vytvárajú ťah a krútiaci moment, čo umožňuje kvadrokoptére meniť pozíciu a vzletovú výšku [4].

Štandardne sa dva páry rotorov točia opačným smerom ako je zobrazené na Obr. č. 1.



Obr. č. 1 Vľavo: otáčanie rotorov kvadrokoptéry, vpravo: smer ťahu pre každý rotor Ti

Každý rotor vytvára ťah Ti, ktorý je kolmý na rameno kvadrokoptéry. Riadením rýchlosti jednotlivých rotorov je možné kontrolovať rotáciu drona okolo jeho horizontálnej osi, v podstate sa jedná o klopenie a nakláňanie viď Obr. č. 2. Aby sa dron točil okolo vlastnej osi, je potrebné aby sa jeden rotor točil rýchlejšie, a to tak aby vznikol rozdiel v ťahu voči protiľahlému rotoru . Naopak, aby sa predišlo nežiaducemu otáčaniu okolo osi y, zvýšenie rýchlosti jedného z rotorov musí byť kompenzované rovnakým znížením rýchlosti rotoru proti nemu (po tej istej osi), čo má za následok nulový točivý moment okolo vertikálnej osi tela [4].

A drone with a diagram of the same direction

Description automatically generated with medium confidence

Obr. č. 2 Osi otáčania kvadrokoptéry

Existuje viacero úrovní autonómie dronov, od úrovne 0 (žiadna autonómia) až po úroveň 5, kedy hovoríme o plnej autonómií [5].

V tejto práci sa budeme zaoberať úrovňou autonómie 3, čo znamená, že dron je schopný lietať autonómne misie, no schopnosť vyhýbania v komplexnom priestore je obmedzená.

V súčasnosti sú autonómne drony skvelým nástrojom na vykonávanie rôznych úloh. Používajú sa aj v oblastiach kde je ohrozený ľudský život [6].

# NEAT

Neuro-evolúcia je evolučný algoritmus na úpravu neurónových sietí pričom požíva genetické operácie ako mutácia, kríženie a selekcia. Týmito operáciami upravuje váhy a biasy neurónovej siete vďaka čomu umožňuje nájsť možné riešenie konkrétneho problému. Rozpoznávame dva druhy neuro-evolúcie a to konvenčnú neuro-evolúciu a neuro-evolúciu augmentatívnych topológií (ďalej ako NEAT) [7].

Konvenčná neuro-evolúcia sa zaoberá hľadaním optimálnych váh a biasov fixnej štruktúry neurónovej siete. Tieto systémy sa trochu líšia od prírody tým, že gény vyvíjajúcich sa umelých neurónových sietí v neuro-evolúcii s fixnou topológiou doslova kódujú ich váhy, ktoré sú od narodenia zamrznuté. Týmto spôsobom sú siete "narodené" s vedomím všetkého, čo kedy budú vedieť, a počas svojho "života" sa už nič ďalšie nenaučia [8].

Váhy avšak nie sú jediným aspektom neurónových sietí, ktoré prispievajú k ich správaniu. Topológia alebo štruktúra neurónovej siete taktiež značne ovplyvňuje ich správanie. NEAT teda okrem hľadania optimálnych hodnôt váh a biasov upravuje aj štruktúru siete [9].

V kontexte riadenia kvadrokoptéry sme zvolili NEAT pre jeho schopnosť adaptácie a riešenia komplexných problémov. NEAT umožňuje dynamicky meniť topológiu neurónovej siete, čo je dôležité pri navigácii v rôznych prostrediach a situáciách. Táto adaptabilita pomáha zlepšovať výkonnosť riadiaceho algoritmu a zvyšovať úspešnosť kvadrokoptéry pri dosahovaní stanovených cieľov.

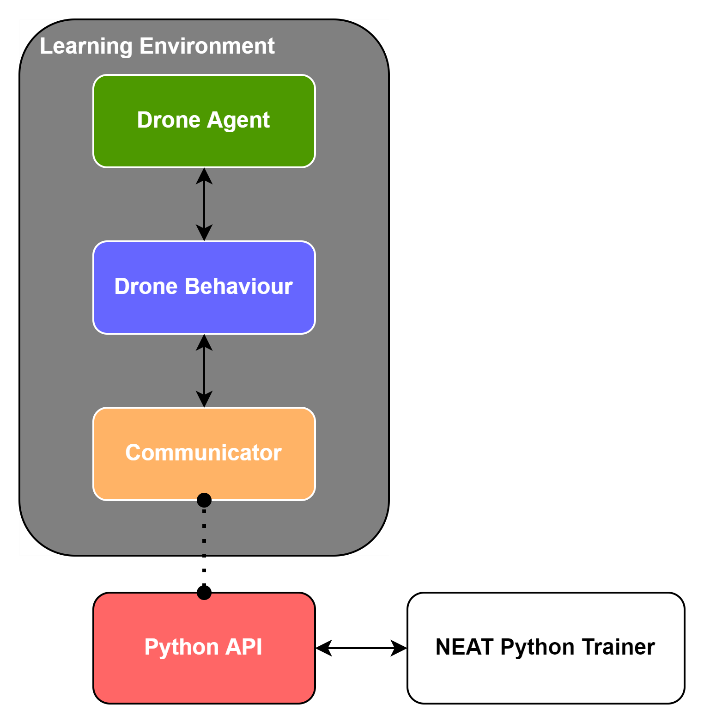
# Implementácia riešenia

Na implementáciu simulačnej časti nášho výskumu sme zvolili hernú platformu Unity. Unity je vďaka svojmu robustnému fyzikálnemu enginu vhodný na simuláciu realistických fyzikálnych javov. Okrem toho Unity integruje ML-Agents toolbox, open-source platformu pre vývoj a trénovanie agentov pre úlohy zamerané na učenie s odmenou. Vďaka tejto kombinácií sme mohli efektívne implementovať kvadrokoptéru do herného prostredia.

Unity avšak nie je navrhnuté na trénovanie, či už konvenčnej neuro-evolúcie alebo NEAT, kvôli čomu bolo potrebné použiť externú knižnicu Neat-python.

## Prepojenie Unity s Neat-Python

Pre spustenie trénovania kvadrokoptéry bolo nevyhnutné prepojiť Neat-python a Unity cez nízko úrovňové API. Tento prístup umožnil komunikáciu medzi Python skriptom, ktorý obsahuje implementáciu NEAT algoritmu, a herným enginom Unity, čo nám umožnilo efektívne trénovať kvadrokoptéru priamo v hernom prostredí. Obr. č. 3 znázorňuje logiku komunikácie medzi týmito dvoma technológiami.



Obr. č. 3 Komunikácia medzi Neat-python a Unity

Simulačné prostredie (Learning environment) obsahuje tri komponenty, a to agenta, chovanie (behaviour) a komunikátor.

* Agent - pripojený k hernému objektu (akejkoľvek postave v scéne), zabezpečuje zbieranie informácií prostredia a vykonáva akcie za ktoré dostáva odmenu alebo pokutu [10].

V našom prípade je to samotný dron

* Chovanie (Behaviour) – definuje špecifické atribúty agenta, ako je počet akcií ktoré môže agent vykonávať. V podstate sa jedná o funkciu, ktorá prijíma informácie od agenta a vracia mu akcie [10].   
  V našom prípade môžeme informácie chápať ako fyzikálne vlastnosti drona a pozícia cieľa.
* Komunikátor – spája simulačné prostredie s nízko úrovňovým Python API, čo umožňuje trénovaciemu algoritmu získavať informácie z prostredia a do prostredia posielať akcie [10].

V našej práci sme od simulačného prostredia vyžadovali informácie z komponentu Chovanie každých 5 simulačných krokov, kde jeden simulačný krok je rovný fixnej časovej konštantne 0.02 [s], čo je rovné 50 snímkam za sekundu.

## Úloha drona

Dron má za úlohu dostať sa do statickej kocky ako je zobrazené na Obr. č. 4. V takejto kocke musí kvadrokoptéra zotrvať určitý časový interval.

Pozícia kocky sa mení podľa aktuálnej úrovne simulácie, kde po dosiahnutí prvotnej pozície kocky sa zvýši úroveň obťažnosti, čo znamená, že sa zmení pozícia kocky do iného smeru. Vďaka tomuto prístupu získame robustnejšie riadenie schopné dostať drona do viacerých pozícií.

A two drones with a cube and a cube with arrows

Description automatically generated with medium confidence

Obr. č. 4 Ukážka cieľa riadenia

## Štruktúra siete

Počiatočná štruktúra neurónovej siete, ktorú používame pre náš problém, je založená na plne prepojenej doprednej neurónovej sieti. Na vstupy sme priviedli vektor o veľkosti 13 hodnôt. Skladá sa z prvkov zobrazených v Tab. 1.

Tab. 1 Vstupy do neurónovej siete

|  |  |  |
| --- | --- | --- |
| Veľkosť | Stĺpec 1 | Jednotky |
| 1 | Uhol do cieľa | [º] |
| 3 | Vektor do cieľa | [m] |
| 3 | Rýchlosť drona | [m/s] |
| 3 | Uhlová rýchlosť drona | [rad/s] |
| 3 | Naklonenie drona | [m] |

Evolúciu začíname s 20 skrytými neurónmi v jednej skrytej vrstve, s náhodnými hodnotami váh inicializovanými pomocou Gaussovho rozdelenia. Výstup siete boli 4 hodnoty pre výkon rotorov kvadrokoptéry. Ako aktivačné funkcie sme použili funkciu sigmoid, vďaka ktorej sme dostali hodnoty z požadovaného intervalu <0; 1> (žiadny výkon rotora, plný výkon rotora).

Populácia siete mala veľkosť 50 jedincov, ktorých sieť sa upravovala počas 2000 generácií. Genetický algoritmus obsahoval aditívnu mutáciu váh a biasov a mutáciu štruktúry, obe s pravdepodobnosťou 15%. Pridanie a odobranie ako neurónu tak aj váh boli vykonané s pravdepodobnosťou 20%. V GA sa kríženie topológií a kríženie jedincov vykonávalo náhodne.

Výsledná neurónová sieť najlepšieho jedinca mala pomerne nízku komplexnosť, čo môžeme prisúdiť genetickému algoritmu knižnice Neat-python. Celkovo sieť obsahovala 18 neurónov v skrytej vrstve a 273 prepojení.

## Simulácia drona

Dron je ovládaný štyrmi vstupmi, kde pre každý rotor je vypočítaný ťah a výsledný krútiaci moment. Vstupy sú v rozsahu <0; 1>, kde 1 znamená plné otáčky rotora a 0 žiadne otáčky rotora. Keďže vieme presnú pozíciu rotora, vypočítame spomínane sily pomocou nasledujúcich rovníc, ktoré použijeme v simulačnom prostredí:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

kde je sila (výsledný ťah) aplikovaný na pozíciu pre í-ty rotor, je výstup neurónovej siete pre i-tý rotor, je koeficient ťahu vypočítaný tak aby sa dron pri vlastnej váhe 1.3 [kg] a , teda polovičnému výkonu rotorov vznášal a konečne je vektor nahor voči lokálnej transformácií objektu.

Pre krútiaci moment použijeme rovnicu v tvare:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

kde je výsledný krútiaci moment aplikovaný na pozíciu   
í-teho rotora, je smer otáčania sa rotora ±1, je koeficient pre násobenie výsledného krútiaceho momentu a je vektor nadol voči globálnemu súradnicovému systému

## Výpočet fitness funkcie

Samotná simulácia trvala 1000 simulačných krokov (20 [s]). Agent môže simuláciu ukončiť skôr a to:

* Preklopením, kedy veľkosť vektora vertikálnej osi drona (y), prekoná vopred stanovenú hranicu
* Vyjdením z pomyselného priestoru prostredia, v tvare gule s vopred stanovenou veľkosťou priemeru

Za každý krok t dostane agent malú odmenu za zotrvanie vo vzduchu v podobe:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Taktiež v každom kroku vypočítavame odmenu za vzdialenosť cieľa a drona v tvare:

|  |  |
| --- | --- |
|  | (4) |

kde sú súradnice cieľa, sú súradnice drona, obe v globálnom súradnicovom systéme, je maximálna vzdialenosť drona od cieľa.

Z fitness funkcie sa odpočítava pokuta za uhlovú rýchlosť v tvare:

|  |  |
| --- | --- |
|  | (5) |

kde je uhlová rýchlosť v kroku t a je maximálna uhlová rýchlosť objektu.

V neposlednom rade do fitness funkcie zahŕňame aj normalizovanú odmenu za uhol k cieľu:

|  |  |
| --- | --- |
|  | (6) |

kde je uhol v kroku t medzi smerom drona a smerom k cieľu. Celková fitness funkcia potom vyzerá nasledovne:

|  |  |
| --- | --- |
|  | (7) |

Maximálna fitness je teda vtedy, keď je k cieľu najbližšie pričom má nulovú uhlovou rýchlosťou (netočí sa okolo vlastnej osi) a je na cieľ priamo natočený. K fitness sa taktiež pripočíta odmena, ak dron zasiahne cieľ a rovnako ak v ňom zotrvá počas simulačného kroku. Pri dotknutí sa cieľa je odmena +100, pri zotrvaní v cieli sa priráta +1.

# Záver

V tejto práci sme predstavili koncept kvadrokoptéry riadenej algoritmom Neat-python v simulačnom prostredí Unity. Aj keď sme zaznamenali isté ťažkosti so zotrvaním v statickom cieli, veríme, že ďalší vývoj a vylepšovanie tohto algoritmu môže viesť k efektívnym metódam riadenia dronov, ktoré sú schopné sa lepšie prispôsobiť dynamickým podmienkam a úlohám v reálnom svete.

Možnosti pre ďalší vývoj zahŕňajú optimalizáciu parametrov algoritmu, prispôsobenie fitness funkcie, implementáciu dodatočných mechanizmov, integráciu senzorov, simuláciu prostredia a experimentovanie s rôznymi modelmi kvadrokoptér. Tieto návrhy môžu poskytnúť smerovanie pre ďalší vývoj algoritmu NEAT v kontexte riadenia kvadrokoptéry v simulačnom prostredí Unity.

# Literatúra

1. G.R. Bhat, M.A. Dudhedia, Autonomous drones and their influence on standardization of rules and regulations for operating–A brief overview,Results in Control and Optimization,Volume 14,2024,100401,ISSN 2666-7207,https://doi.org/10.1016/j.rico.2024.100401. (<https://www.sciencedirect.com/science/article/pii/S2666720724000316>)
2. McIntyre, A., Kallada, M., Miguel, C. G., Feher de Silva, C., & Netto, M. L. neat-python [Computer software]
3. Asif Ali Laghari, Awais Khan Jumani, Rashid Ali Laghari, Haque Nawaz, Unmanned aerial vehicles: A review, Cognitive Robotics,Volume 3,2023,Pages 8-22,ISSN 2667-2413,https://doi.org/10.1016/j.cogr.2022.12.004.(https://www.sciencedirect.com/science/article/pii/S2667241322000258)
4. Mariani, M.; Fiori, S. Design and Simulation of a Neuroevolutionary Controller for a Quadcopter Drone. Aerospace 2023, 10, 418. <https://doi.org/10.3390/aerospace10050418>
5. Alsadik, Bashar & Nex, Francesco. (2021). The Rise in UAV Inspections for Civil Infrastructure https://www.gim-international.com/content/article/the-rise-in-uav-inspections-for-civil-infrastructure. GIM International.
6. Mohsan, Syed Agha Hassnain & Othman, Nawaf Qasem Hamood & Li, Yanlong & Alsharif, Mohammed & Khan, Muhammad. (2023). Unmanned Aerial Vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends. Intelligent Service Robotics. 10.1007/s11370-022-00452-4.
7. S. Risi and J. Togelius. Neuroevolution in games: State of the art and open challenges. IEEE Transactions on Computational Intelligence and AI in Games, 7(1):1–16, 2015.
8. Kenneth O. Stanley . Neuroevolution: A different kind of deep learning, O’Reilly <https://www.oreilly.com/radar/neuroevolution-a-different-kind-of-deep-learning>
9. K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. Evolutionary Computation, 10(2):99–127, 2002
10. Juliani, Arthur and Berges, Vincent-Pierre and Teng, Ervin and Cohen, Andrew and Harper, Jonathan and Elion, Chris and Goy, Chris and Gao, Yuan and Henry, Hunter and Mattar, Marwan and Lange, Danny, Unity: A general platform for intelligent agents, arXiv preprint arXiv:1809.02627, https://arxiv.org/pdf/1809.02627.pdf