REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK
35. évfolyam (2023) 3. szám 51–68. • DOI: 10.32560/rk.2023.3.4

Mihályi Géza

Az UAV-pályatervezés kihívásai és lehetséges megoldásai

Kutatásom során az UAV-pályatervezés nehézségeit és kihívásait vizsgáltam. Bemutatom az esetlegesen felmerülő legismertebb problémákat. Ilyen lehet a „pontszerű test"-probléma (Point Vehicle) vagy a „kocogó"-probléma (Jogger's Problem). Bemutatom a legismertebb és jelen tudásunk szerint leghatásosabb, State-of-Art[1] megoldásokat is, mint a Visible Graph vagy az A alapú algoritmusok.*

Kulcsszavak: *pályatervezés, A* algoritmus, Q-Learning, UAV, önvezető*

1. Bevezetés

UAV[2]-kat már használtak az 1800-as évek közepén is, de akkor még nem úgy néztek ki, mint a mai drónok, főleg hőlégballonok voltak, amelyeket felderítésre és gyakorlásra használtak. De térjünk vissza időben a 2013-as évbe, amikor a drónok a köztudatban is egyre ismertebbek lettek. Az Amazon cég ugyan bejelentette, hogy drónokat fog használni a termékek szállításához, ez a mai napig sem történt meg. Ám például Afrikában arra is láthatunk példát, hogy drónokkal oldják meg a vér-, illetve eszközszállítást. A módszer költséghatékonyabb, mint a helikopter, és gyorsabb, mint a mentőautó. A másik előny, hogy az UAV elér olyan helyekre is, ahová hagyományos úton nem, vagy csak nehezen tudnánk eljutni [1], [2]. A Goldman Sachs 2016-os előrejelzése szerint a globális drónpiac 2020-ra eléri a 100 milliárd USD értéket, amiből 70 milliárdot jósoltak a katonaságnak, 17 milliárdot az átlagos vásárlónak, azoknak, akik hobbiból építenek drónokat, és 13 milliárdot vállalatoknak [3]. A Statista egy újabb, 2021-es kutatása szerint 2021-ben a globális kereskedelmi drónpiac eléri a 26,3 milliárd USD értéket. A 2026-os előrejelzések szerint ez az érték elérheti a 41,3 milliárd USD-t [4]. Ahhoz, hogy az ilyen és ehhez hasonló drónok akár szállítási, akár katonai céloknak megfeleljenek, kell hogy legyen bennük önvezető képesség, aminek fontos része a pályatervezés.

[1] Legkorszerűbb.
[2] Unmanned Aerial Vehicles – pilóta nélküli repülőgép.

2. Problémafelvetés, motiváció

Hogy a drón eljusson A pontból B pontba, jól meghatározott utat kell követnie. Miután ezt az utat meghatározta, az útvonalat folyamatosan frissítenie kell, mivel a városi vagy akár a katonai terep kiszámíthatatlan terep. A drón utat tervez, de a környezetét részben vagy egyáltalán nem ismeri, emiatt folyamatosan alkalmazkodnia kell hozzá. A legtöbb esetben erről van szó, a valós repülés során ritka a statikus közeg, ahol az UAV az előre meghatározott úton tudna haladni. A dinamikusan változó környezetekre hozott megoldást Xiaojian Hou és csapata [5], akik a mesterséges intelligenciát, azon belül is a megerősítéses tanulást hívták segítségül, abból is a fejlesztett Q-Learning algoritmust (a későbbiekben részletesebben is szó lesz róla).

Ahhoz, hogy a drónokat elfogadja a társadalom, biztonságossá/biztonságosabbá kell tenni őket. Városon felüli repülésnél több komolyabb baleset is történhet, ami rosszabb esetben emberéleteket is követelhet, jobb esetben magántulajdont károsíthat. Minden ilyen baleset minimalizálásának céljából Quan Shao és csapata cost-benefit assessment modellt[3] készített, amely vizsgálja a veszély lehetőségeit és a szolgáltatás előnyeit [6]. Ezenkívül általános értelemben az egyik leghasznosabb ütközéselemző modell a The Longitudinal Reich Collision Risk Model,[4] amelyet az előbb említett kutatásban is használnak, és használják nem csak drónok esetén, hogy megállapítsák, mekkora az esélye, hogy két légi jármű ütközik egymással. A modellt egy kutatócsoport fel is használta, hogy kiderítse, mekkora valószínűséggel ütköznek össze a repülők leszálláskor [7]. Motivációmul szolgál, hogy mindezen ismeretekkel, mind a saját programozói tudásommal együtt szimuláljak egyes algoritmusokat, és megpróbáljam továbbfejleszteni őket.

3. Szakmai-tudományos előzmények

Annak eldöntésére, hogy melyik útkereső algoritmus a legjobb, régóta keresik a választ, több összehasonlító kutatás készült már a témában, mint például B. Moses Sathyaraj és csapata munkája, amelyben összehasonlítottak több algoritmust is, köztük a Distance Vector algoritmust,[5] a Floyd-Warshall's algoritmust, illetve a manapság egyik legjobban elterjedtet, az A* algoritmust. A kutatásban megállapítják, hogy a BFS[6] akkor is abba az irányba halad tovább a cél felé, amikor már látszik, hogy nem az a legrövidebb irány, illetve, hogy az A* algoritmus mindig megtalálja a legrövidebb utat [8]. Egy 12 évvel későbbi kutatásban Shubhani Aggarwal és Neeraj Kumar hasonlóval állnak elő, ahol megvizsgálják különböző koncepciójukat elméletben. Ez a kutatás modernebb, abban az értelemben, hogy az újabb technológiákat is vizsgálja, mint például a mesterséges intelligencia lehetőségeit. Ez a kutatás kitér a pályatervezés nehézségeire és a megoldásaira is, illetve jövőbeni kihívásaira is. A kutatásban felvetődik, hogy milyen jövőbeli kutatásokat lenne célszerű követni [9]:

[3] Költség-haszon értékelési modell.
[4] Hosszúsági ütközési modell – ennek a modellnek a segítségével határozzák meg, hogy mekkora eséllyel ütközik össze két légi jármű.
[5] Távolság-vektoros algoritmus – az egyes csomópontok tudják a távolságot közöttük és az összes többi lehetséges csomópont között.
[6] Breadth-First Search – keresési algoritmus, amely először az adott csúcshoz legközelebbi, vele egy szinten lévő csúcsokat vizsgálja át, és csak utána folytatja a keresést mélyebben. Szélességi keresésnek hívják.

a) habár a pályatervezés hatékonyságára már több kutatás is készült, még mindig tele van kihívásokkal és lehetőségekkel;
b) talán az egyik legfontosabb a drónok szempontjából, hogy sokáig a levegőben legyenek képesek maradni, emiatt az energiahatékonyságra is nagy hangsúlyt kell fektetni a kutatásokban is. Ezeket jobb akkumulátorokkal vagy újabb megoldásokkal lehetne elősegíteni, mint a napenergia. Napenergiás megoldásra már láthatunk példát;
c) a drónok közti kommunikációt is fontos lenne kutatni, hogy gyorsabban és vezetékek nélkül tudják átadni egymásnak az információt.

Kisméretű, multirotoros UAV-k városi, illetve katonai műveleti területi repülési pályáinak tervezésével Szabolcsi foglalkozott [10], [11]. Az UAV-k felszálló és leszálló repülési pályáit Szabolcsi vizsgálta, és megadta a lehetséges sikló- és iránypályákat [12], [13].

4. A pályatervezés kihívásai

A szakirodalom alapján a szoftveres rendszer legfontosabb eleme a pályatervezés és ennek kutatása, hogy megtalálják az ideális utat a kiinduló helyzet és a célhelyzet között, elsődleges prioritást élvez. A pályatervezési probléma során a drónnak el kell jutnia A helyről B helyre úgy, hogy közben ne ütközzön a környezetében lévő dolgokkal és más drónokkal. A sima 2D-s környezeti ábrázolás nem lenne megfelelő, mert nem lehetne pontosan elhelyezni a környezetben a tárgyakat, az kell, hogy legyen egy 3D-s komplex kép a környezetről. A pályatervező algoritmus eltérő lehet kereskedelmi drónoknál, illetve katonaiaknál. A kereskedelmi forgalomban lévő drónoknál, illetve a szállítási drónoknál három alapelem van: a Motion planning,[7] a Trajectory planning[8] és a Navigation[9]. A katonai drónok abban térnek el, hogy esetükben több a bemeneti paraméter, amit számításba kell venniük, és amivel egy többcélú optimalizálási probléma jön létre.

Fő célja a pályatervező technikáknak, hogy a számítási szükségletet és a számítás idejét lecsökkentsék, amíg a drón megtalálja az optimális utat. Optimálisnak kell lenniük olyan szempontból is, hogy a drón a lehető legkevesebb energiát fogyassza, rövidebb időbe kerüljön, amíg elér a célig, és ne ütközzön össze közben más drónokkal vagy környezeti tárgyakkal. A pályatervező algoritmusok fő kihívásai a következők [9]:

a) a pálya hossza: a drón által megtett távolságot jelenti a kiinduló állapotból a végállapotba;
b) optimalitás: a teljes rendszernek optimálisnak kell lennie, energia, ár és egyéb elvárás tekintetében. Az optimalitást három módon lehet kifejezni: 1. optimális, 2. szuboptimális, 3. egyáltalán nem optimális;
c) teljesség: akkor nevezzük teljesnek, ha a pályatervezésnél meghatározott feltételeket teljesíti, ha létezik olyan út. Megoldást tár a drón elé, amelyet követnie kell;

[7] Pálya útjának a megtervezése.
[8] Az út megtervezése időben és térben, itt már kalkulálunk a jármű sebességével is.
[9] A kettő egybe illesztése, amikor már a levegőben van, teljesítse a rábízott utat a megadott sebességgel.

d) költséghatékonyság: több összetevőből áll, és függ a teljes számítási költségtől. Összetevői közé tartozik az üzemanyag ára, akkumulátorok töltésének a költsége, szoftver- és hardverárak;
e) időhatékonyság: a drón megteszi az adott utat a kiindulóponttól a végpontig ütközés nélkül a minimális idő alatt;
f) energiahatékonyság: a drón a lehető legkevesebb energiát használja, miközben megteszi útvonalát és számolja a röppályáját a művelet közben;
g) ütközéskerülés: a képesség, hogy ha a röppályájába kerül valami, akkor azt kikerülje, és folytassa az útját anélkül, hogy károsodna a szerkezet vagy az elektronika.

Mindezt még az is nehezíti, hogy a legtöbb környezeti tárgyról csak hiányos információink vannak, ami főleg igaz a dinamikusan mozgó tárgyakra, mint az emberek, autók, más drónok. Ezekkel a drónok tervezőszoftverének menet közben kell megküzdenie. Ahhoz, hogy az ilyen tárgyakat felismerje, a különféle szenzorokon kívül a kamera jelent hatalmas segítséget, ami a mesterséges intelligencia fejlődésével gyakorlatilag az önvezető drónok szemévé tud válni, és egy távolságmérő szenzor segítségével meg tudja határozni, hogy az előtte lévő mozgó tárgy milyen messze van, merre mozog. Meg kell még küzdeni a drónok kinematikai és dinamikai kötöttségével is. Az is megállapítható, hogy minden szituáció más, így nehéz egyetlen olyan algoritmust írni, amely megfelelne mindegyiknek. Minden feladathoz specifikus algoritmust kell írni. Mivel a környezet, amelyben működik a drón, ismeretlen, az egyik megoldás, hogy használjunk valamilyen sejtekre lebontó algoritmust.[10]

Nehezítő körülménynek számítanak a légköri zavarok, atmoszferikus zavarok – szinte lehetetlenné teszik, hogy a drón az előre tervezett pályán tudjon haladni, és ott van még a drón állapotának bizonytalansága és a limitált tudásunk a környezetről.

Az UAV-irányítási problémát háromdimenziós problématérrel, limitált környezeti információval, limitált távolságú szenzorokkal, sebesség- és gyorsaságkényszerekkel és a szenzor adatbizonytalanságával szokás jellemezni.

Ebben a publikációban [14] a kutatók meghatározták, hogy milyen problémákkal találkozhatunk. Az egyik a statikus, ahol a környezetet ismerjük, a másik pedig a dinamikus, ahol a környezetet nem ismerjük teljesen, vagy változik, ahogy haladunk előre az időben. Amikor a tárgyak fix helyen vannak a térben, akkor időfüggetlennek nevezzük őket, amikor tudnak mozogni, akkor időfüggőnek. Az általános problémák közül megkülönböztetjük a „pontszerű testet" (Point Vehicle), a „kocogó"-problémát (Jogger's Problem), a „bogár"-problémát (Bug Problem), a „súlyozott régió"-problémát (Weighted Region Problem), az „időben változó tér"-problémát (Time-Varying Environments) és a „sétáló"-problémát (Mover's Problem).

4.1. „Pontszerű test"-probléma

A jármű egy pontként van modellezve a térben. A konfigurációs tér megegyezik az effektív térrel, amiatt nem kell semmit csinálnia. Az egyik legegyszerűbb probléma, és az optimalitás megegyezik a kezdő- és a végpont távolságával.

[10] Cell Decomposition Method.

4.2. „Kocogó"-probléma

A kocogó dinamikai problémájával foglalkozik, amikor a kocogónak limitált a látási tere. A probléma azt reprezentálja, amikor egy jármű/drón változó, időfüggő környezetben mozog, limitált szenzortávolsággal. Optimalitását úgy lehetne meghatározni, ha minimalizálja a repülési időt a kiinduló helyzet és a cél között, vagy minimalizálja valamilyen másik attribútumát, mint például az energiafogyasztását.

4.3. „Bogár"-probléma

Különleges változata a „kocogó"-problémának: a jármű betekintési szöge 0. Ilyenkor a járműnek meg kell érintenie, vagy nagyon közel kell kerülnie a tárgyakhoz, hogy érezze őket.

4.4. Az „időben változó tér"-probléma

A járműnek el kell kerülnie azokat a tárgyakat, amelyek időben mozognak. Optimális megoldása, hogy minimalizáljuk a megtételéhez szükséges időt vagy az úthosszt.

4.5. „Sétáló"-probléma

Akadályokkal teli *mezőn* kell keresztülmenni, hogy elérjük a célt. A járművet merev testnek vesszük. Feltételezzük, hogy az objektumnak nincs semmilyen dinamikai kötése. A probléma a jármű komplexitását vizsgálja hozzáadva az akadálymező komplexitását, majd az ebből következő számot m vagy M számnak nevezi.

5. Pályatervezés-lépés

A kereskedelmi és szállítási drónoknál a Motion Planning felel az út optimalizálásáért, hogy minimalizálja az út hosszát, a fordulási szöget és azt a maximális repülési távolságot, amelyet meg tud tenni az üzemanyaggal. A Trajectory Planning kiegészíti a Motion Planning részt azzal, hogy a tervezett úthoz olyan értékeket rendel, mint a sebesség, idő, illetve az UAV kinematikája. A Navigation olyan részekből áll, mint a Motion Planning, Trajectory Planning, ütközéselkerülés és helyzetmeghatározás (lokalizáció). Az alaptervezés két fázisból áll. Az első fázis az úgynevezett pre-processing,[11] azaz előkészületek. Ebben a fázisban a már említett 3D-s környezetre rárajzoljuk a szükséges pontokat és vonalakat, majd a gráfokból álló térképet generálunk [9]. Habár az előző publikációban [9] azt fejtegették, hogy nem lehet, vagy nem olyan pontosan lehet 2D-s környezetben elhelyezni a tárgyakat, addig egy másik publikációban [15] amellett érvelnek, hogy elegendő 2D-ben megtervezni, a 3D-s környezetet pedig a műveleti résznél kell figyelembe venni. A drón különböző útvonalpontok között közlekedik.

[11] Előkészületek: a nyers adatokat tisztítjuk, és a gép számára értelmezhető állapotba hozzuk.

Itt megállapítanak 2 formulát, az egyik a biztonságos távolság a drón és a tárgyak között, ezt $r_{biztonságos}$-el jelöljük, illetve a már korábban is említett maximálisan megtehető távolságot. A maximálisan megtehető távolság képlete:

$$L \leq L_{max}, L = \sum_{i=1}^{m-1} l_i \quad (1)$$

Ahol az l_i a különböző waypontok közötti távolság, az L_{max} a maximális távolság, az m pedig az útvonalpontok számát jelöli. Az ütközésmentes úton lévő útvonalpontoknak meg kell felelnie a következő képletnek:

$$\frac{[(x_i - x_k)\cos(\theta) + (y_i - y_k)\sin(\theta)]^2}{(a - r_{biztonságos})^2} + \frac{[(x_i - x_k)\cos(\theta) + (y_i - y_k)\sin(\theta)]^2}{(b - r_{biztonságos})^2} = 1 \quad (2)$$

Ahol a jelölések a következők [15]:

- x_i, y_i – az adott waypontok koordinátái;
- a, b – az ellipszis két féltengelye;
- θ – a fél nagytengely dőlése;
- x_k, y_k – az adott tárgyak középpontja;
- $r_{biztonságos}$ – a drón és a különböző természeti tárgyak minimális távolsága.

Minden ilyen pályatervező technika és algoritmus ezen pontok és vonalak alapján határozza majd meg az ideális pályát. A második fázis az úgynevezett lekérdezésfázis *(Query)*. Ebben a fázisban különböző algoritmusok futnak le, olyanok, mint az ant colony[12] vagy a Floyd–Warshall.[13]

A katonai drónoknál több kitétel van, mert azok ki vannak téve az ellenséges erőknek is; hogy láthatatlanok maradjanak az ellenség számára, el kell hogy kerüljék a radarokat, „láthatatlanná" kell válniuk.

1. Először egy úgynevezett rejtett utat *(Stealthy Path)* kell kialakítania. A rejtett út lényege, hogy úgy kormányozza a drónt, hogy annak radarjelét az ellenséges radarok ne tudják befogni. A feladat nehéz, mert nem minden drónnak ugyanolyan a radarjele, ami azt jelenti, hogy a radar által kibocsátott sugárzást nem ugyanúgy verik vissza, az egyik irányba jobban, mint a másikba. Ha a visszavert jelben valamilyen tüske van, akkor az algoritmusnak el kell irányítania a radaroktól.
2. Másodszor az algoritmusnak meg kell felelnie, hogy a pálya útja minimális legyen azzal együtt, hogy számításba veszi a rejtett utat, miközben kielégíti a repülő dinamikai kényszereit is.

[12] Hangyakolónia: algoritmus, valószínűségszámító módszer, amely a grafikonokon vagy gráfokon keresztül meg tudja találni a jó utakat. A mesterséges hangyák az összes lehetséges megoldást reprezentáló paraméterben mozogva keresik meg az optimális megoldásokat.

[13] Floyd–Warshall-algoritmus: ez az algoritmus megtalálja a legrövidebb utakat egy súlyozott gráfban. Megtalálja az összes csúcspár közötti legrövidebb távolságok hosszát. Végigmegy az élek minden kombinációján, miközben folyamatosan javítja a becslést a két csúcs közötti legrövidebb útvonalra vonatkozóan, amíg a becslés nem lesz optimális.

3. Harmadszor, a szoftvernek meg kell felelnie a misszió céljának. A gyakorlatban ez a legtöbb esetben azt jelenti, hogy egy akció több drónos támadást tartalmaz, vagyis a szoftvernek figyelnie kell, hogy a drónok egyszerre érjenek célba.
4. A szoftvernek az UAV processzorán kell futnia, hogy ha valamiért a röppályát újra kell tervezni, akkor ezt valós időben *(Real-Time)* tudja megtenni [16].

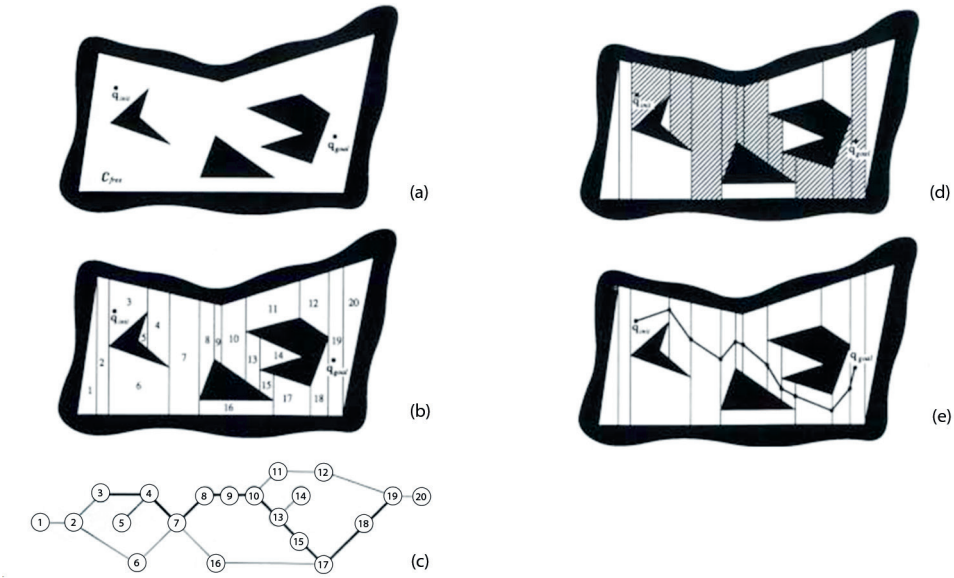
6. A pályatervezés egyes megoldásai

Annak érdekében, hogy a számítási szükségletet és idejét lecsökkentsék, különböző publikációk, különbözőképpen osztották fel az adott területet. Például voltak, akik hibrid megosztási technikát alkalmaztak, és a lefedettségi területet pontos formákra bontották, háromszögekre. Mások hasonló elvek mentén spirálszerű formákra bontották fel az adott komplex lefedettségi területet [17]. Azonban, ahogy feljebb már írtam, az, hogy milyen algoritmust választunk, attól függ, hogy milyen problémát kell megoldanunk. Például, ha azt szeretnénk, hogy a drón átrepüljön a sivatag felett, vagy az alatta lévő utat kövesse, ahol esetleg kisebb kanyarok vannak, és nincsen zavaró tényező, akkor szükségtelen, hogy olyan algoritmust használjunk, amely belekalkulálja a dinamikus változásokat, mert ezzel csak a számítási igények nőnek. A leggyakrabban használt mérték, amivel meghatározzuk, hogy milyen, mennyire bonyolult algoritmust kell használnunk, azt a térben lévő akadályok határozzák meg. Ezeket általában az akadályok száma alapján határozzák meg, N számnak nevezik. Arról, hogy milyen mértékeket célszerű alkalmazni, Rhinehart írt ebben a kutatásában [18]. Annak érdekében, hogy egy algoritmust sikeresnek lehessen tekinteni, mind műveleti, mind számítási területen meg kell felelnie. Folyamatosan megfelelő távolságot kell tartania az akadályoktól, sima útvonalat kell követnie, ami mellett figyelnie kell az energiafelhasználásra és a megtett távolságra. Kell egy megbízható, valós időben történő számítás is, amiben nem történnek váratlan késések, mivel csak így lehet biztosítani a műveleti kritériumokat. Emiatt minél kisebbnek kell lennie a számítási komplexitásnak: minél gyorsabb az algoritmus, annál gyorsabban tud reagálni a kialakult helyzetre és változtatni az irányát, műveletét.

A jelenlegi State-of-Art megoldások a következők:

6.1. Sejtekre lebontó algoritmus (Cell Decomposition Method)

Az alapvető ötlet, hogy az utat a kezdőpont és a végpont között meg lehet úgy is határozni, ha a köztük lévő szabad helyet felbontjuk több kisebb régióra, amelyeket sejteknek *(cells)* hívunk. Az ezután létrejövő kapcsolati gráf az egymással szomszédos sejtek kapcsolatán alapul, ahol is minden sejtet egy csomópont reprezentál a szabad térben. A csomópontok között lévő kapcsolatok megmutatják, mely sejtek szomszédosak egymással. Majd folytonos út jön létre, amelyet ha simán követünk, el tudunk jutni a kiindulópontból a végpontba. A kép hivatott bemutatni, hogyan történik ez meg a gyakorlatban [19].



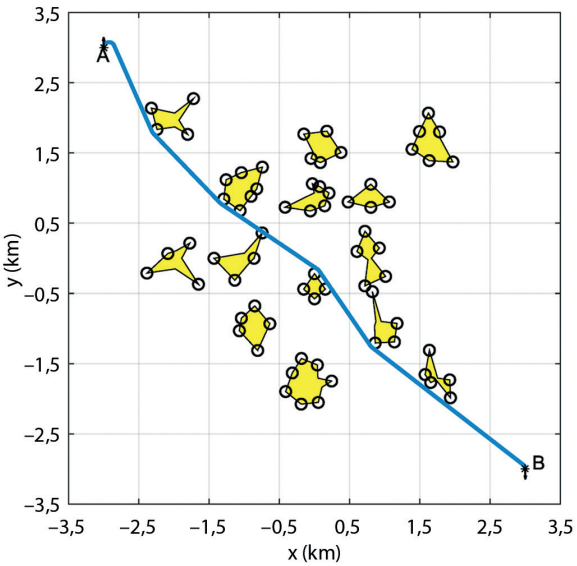
1. ábra
Látható, hogyan történik a sejtekre bontás [19]

A folyamat lépései kifejtve a következők:

a) Először fel kell bontani a szabad teret kisebb részekre, ebben a példában sokszögek által van határolva. A sokszögeket felbontjuk háromszögekre, illetve négyszögekre, úgy, hogy párhuzamosokat húzunk a belső alakzatok csúcsaihoz.
b) Utána minden sejt, felbontott rész kap egy számot, és mostantól csomópontként lesz ábrázolva a kapcsolati gráfban.
c) Az egymással szomszédos csomópontokat a feldarabolt képen összekötjük.
d) A kapcsolati gráfból kirajzolódó utat fel lehet vetíteni a feldarabolt képre. A csomópontokhoz tartozó sejteket besatírozva jelölték.
e) Ezután a sejthatárolók középpontját összekötötték, és ebből alakult ki az az út, amellyel el lehet jutni a kezdőponttól a végpontig.

6.2. Visible Graph – látható gráf

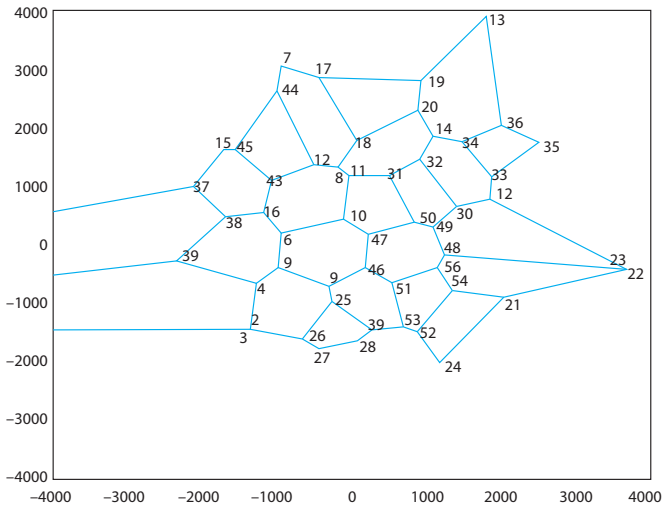
Ez a megoldás kifejezetten a „pontszerű test"-probléma megoldása. Viszont ezt csak 2D-s környezetben lehet számítani. Az út ebben az esetben úgy alakul ki, hogy az algoritmus „súrolja" a sokszögek csúcsait, majd egy útvonaltervet csinál ezekből a vonalakból, úgy, hogy minden csúcsot összeköt minden csúccsal, amit lát. Mivel a minimális út többször is nagyon közel kerül az akadályokhoz az út folyamán, nem, vagy alig garantál valamilyen baleset-megelőzést olyan környezetnél, ahol valamilyen bizonytalanság van az akadályok pozíciójában. Az illusztráció a következő kutatásban szerepel [20].



2. ábra
Visible Graph, azaz „látható él"-módszer, amikor a drón egyik élpontból megy a következő látható élpontba [20]

6.3. Voronoi Roadmap

Ez a fajta algoritmus épít egy vázat, amely maximális távolságra van az akadályoktól, majd megtalálja a minimális távolságot, amin követni tudja ezt a vázat. Ez az algoritmus is kétdimenziós, viszont próbálták áttenni 3D-be, ami sikerült is, de nem optimális. Voltak kutatások, amelyek hierarchikus Voronoi-gráfot készítettek, amelyet általánosítani lehet több dimenzióra is. A diagram felépítése a következő: van N darab pont a síkon, és minden szomszédos pont összekapcsolva egy háromszöget alkot. A háromszög minden oldalára merőleges felezőt állítunk, majd ezek a felezők a pontok körül különböző sokszögeket alkotnak. Ezek a sokszögek a Voronoi-sokszögek, és sok ilyen Voronoi-sokszögből lesz a Voronoi-diagram. Amikor a drón repül ezen a diagramon, egy értékelő funkció fut rajta végig, hogy melyik élt kellene használnia ahhoz, hogy optimális legyen az útja, illetve hogy elkerülje a veszélyeket katonai környezetben. Ezt a megoldást gyakran használják a katonaságnál. Az illusztráció a következő kutatásban jelent meg [21].



3. ábra
Radarveszélyek Voronoi-diagramja [21]

6.4. A-Star-alapú algoritmusok

Régóta használatban van a pályakeresést kutató közösségekben. Elsősorban egyszerűségének és könnyű változtathatóságának köszönhető nagy előnye más algoritmusokkal szemben. Nem túlságosan komplex, cserébe viszonylag nagy tárhelyre van szüksége, hogy tárolja az általa használt adatokat. Determinisztikus algoritmus, amelyet először arra használtak, hogy megtalálja a legrövidebb utat egy 2D-s négyzetrácsban. Iterációs, vagyis a csomópontokból mindig abba az irányba indul el, amerre a legkevesebb az út költsége. Ez a költség, ami most az úthossz, úgy számolódik, hogy a kiinduló csomópont távolsága az adott pontig, plusz az adott pont becsült távolsága a végponttól. Ezt a becsült távolságot heurisztikus távolságnak nevezik. Van néhány ilyen heurisztikus távolság, mint például a Manhattan-távolság. A Manhattan-távolság 2 pont és $x = (x_1, x_2, \ldots x_n)$ és $y = (y_1, y_2, \ldots y_n)$ között n-dimenziós térben a pontok távolságának az összege minden dimenzióban.

$$d(x, y) = \sum_i^n |x_i - y_i| \tag{3}$$

Az elnevezés onnan ered, hogy ezt a távolságot teszi meg egy autó egy városban. Szokták még nevezni L_1 és 1-norm távolságnak is [22]. Minél pontosabb ez a becslés, annál gyorsabban működik az algoritmus. Ez az algoritmus mindig megtalálja az optimális megoldást, ha h heuralisztika kielégíti a

$$h(x) \leq d(x, y) + h(y) \tag{4}$$

kiegészítő feltételt a gráf minden (x, y) élére (ahol d az adott él hosszát jelöli) [23]. Ettől függetlenül, ahogy Daniel Foead és csapata rámutat ebben a kutatásában [24], az A* algoritmus

nem elegendő magában, több kiegészítő algoritmus is kellhet hozzá. Főleg akkor, amikor többügynökös útkeresési problémát kell megoldani, mert ilyenkor gyakran előfordul, hogy a két ügynök ütközik egymással a nem megfelelő út miatt. A mostani kutatások is az A*-ral kapcsolatban arról szólnak, hogy lehetne az A*-t módosítani, hogy tudjon kezelni ügynököt is. A felmerülő megoldás nevét az állatvilágból kapta: rajintelligencia *(Swarm Intelligence)*. Az alkalmazásra már példát is láthattunk az Inteltől, amikor az égre küldtek 500 drónt, és fényjátékot mutattak be. A metódus bebizonyította, hogy csökkenti a számítási igényt, azonban ez sem általános megoldás.

6.5. „Potenciális mező"-metódus (Potential Field Method)

Az előzőektől kicsit eltérően, ámbár kicsit az A* algoritmushoz hasonlítva, megjelent a Potential Field Method is. Hatalmas előnye, hogy az ismeretlen eseményeken is túl tud lépni, meg tudja őket oldani azzal, hogy az UAV-nek, vagy bármilyen másik önvezető robotnak, számításba veszi a jelenlegi helyzetét és a helyzetének lehetőségeit, realitásait. Ebben a metódusban kétfajta erő lép fel. A vonzó erőket a célok és állomások bocsátják ki, és vannak a taszító erők, amelyeket pedig az akadályok adnak. Az A*-hoz hasonlóan ez is közkedvelt algoritmus a kutatásokban, mivel egyszerű, és magas biztonsági szinttel dolgozik, illetve használható valós időbeli alkalmazásra is, mivel nincsenek benne bonyolult matematikai képletek, vagyis a számítási idő is kevés. A hatékonyságával kapcsolatban Krogh és Thorpe 1986-ban állítottak fel egy általános „potenciális mező"-metódust, amelyben kombinálják a globális- és a lokálisút-tervezést. Ezenkívül Brooks 1986-ban és Arkin 1989-ben közölt publikációiban valós szenzoradatok és a „potenciális mező"-metódus alapján mozgatták a robotjaikat. Nagyvonalakban úgy kell elképzelni ezt a metódust, hogy az UAV-t pontszerű testnek tekintjük, amelyre hatnak a külső pozitív és negatív erők. Ezeknek az erőknek az összeadásával megkapjuk az úgynevezett eredő erőt, amelynek az irányába a drónunk el fog indulni. Viszont ennek az algoritmusnak nagy hátránya, hogy könnyen a lokális minimumba eshet, és onnan nem tud kijutni sehogyan sem. Pályatervezés során az algoritmusnak el kell döntenie, hogy hogyan mozgassa az UAV-t a kezdő ponttól a végpontig anélkül, hogy ütközne, és az útja folytonos legyen. Ezt az algoritmust gyakran használják olyan eseményeknél, ahol a környezet ismeretlen és dinamikusan változik. Ez, ellentétben az A* algoritmussal, nem valamilyen matematikai távolságot vesz alapul és az alapján indul el adott irányba, majd korrigálja magát, hanem az erők erőssége számít. Bár a távolság itt is szerepet játszik, mert minél közelebb vagyunk a célhoz, annál erősebb az általa kifejtett erő, és annál jobban húzza be az adott robotot [25]. Viszont ennek is vannak hibái és nem optimális eljárásai:

a) a lokális minimum elérését csapda szituációnak nevezik, ebben az esetben a drón nem ér el a célig, mert megáll a lokális minimum pontjánál;
b) leng, illetve kileng/kilenghet a drón, ha akadály közelébe kerül;
c) habár a drón a fizikai méreteit tekintve átférne két akadály között, az erők úgy határozzák, hogy ki kell kerülnie az objektumot, így nem az optimális elérési utat fogja választani;
d) szűk utak között is kilenghet, főleg, ha változás áll be a falak távolsága között.

Ezeket Y. Koren és J. Borenstein fedezték fel 1991-ben [25].

6.6. A virtuális erőmező-metódus (The Virtual Force Field [VFF] Method)

A metódust [26] Y. Koren és J. Borenstein találta ki 1989-ben, ami arra volt tervezve, hogy valós időben lévő, gyorsan mozgó járművek ki tudják kerülni az akadályokat. Kutatásaikból kiderült, hogy ez a metódus koordináta-rendszert használ, amelynek a hisztogramrács nevet adták, amelyben az akadályok vannak. Minden cella valamilyen bizonyossági értékkel van felruházva, ami az algoritmus magabiztosságát szemlélteti, hogy azon a mezőn egy akadály van. Amikor megy a jármű, és a szenzorai érzékelnek 1-1 cellában akadályt, akkor azoknak a celláknak a bizonyossági értékei megnőnek. Ezzel együtt megjelenik a „potenciális mező"-metódus is. Ahogy a robot mozog, egy w_s * w_s lefedő mező követi, amely lefed egy C régiót; ez a régió lesz az aktív régió, és azok a cellák, amelyek beleesnek, lesznek az aktív cellák. Minden cella kiad magából valamilyen taszító erőt a robot felé. Ennek az erőnek a nagysága proporcionális az aktív cellákra, és inverzen proporcionális *(Inversely Proportional)* a d^n-hez, ahol is a d a távolság a cella közepe és a jármű közepe között, és az n egy pozitív szám:

$$F_{ij} = \frac{F_{cr} W^n C_{ij}}{d^n(i,j)} \left(\frac{x_i - x_0}{d(i,j)} \hat{x} + \frac{y_i - y_0}{d(i,j)} \hat{y}\right) \tag{5}$$

ahol is:

- F_{ij} – virtuális taszító erő;
- F_{cr} – taszítóerő-állandó;
- $d(i, j)$ – az i, j aktív cella távolsága a robothoz képest;
- C_{ij} – az aktív cella bizonyossági értéke;
- W – a robot szélessége;
- $x_{0,} x_0$ – a robot adott koordinátái;
- x_i, y_i – az aktív cellák koordinátái.

Feltételezésükben az n = 2 volt. Minden virtuális taszító erő összeadva adja ki az eredő taszító erőt, F_r-t.

$$F_r = \sum_{i,j} F_{ij} \tag{6}$$

Ezzel egyidejűleg ugyanilyen vonó erő is kialakul, amivel a cél maga felé húzza a járművet.

$$F_t = F_{ct} \left(\frac{x_t - x_0}{d_t} \hat{x} + \frac{y_t - y_0}{d_t} \hat{y}\right) \tag{7}$$

Ahol is a D_{ct} a cél vonzási ereje, d_t pedig a robot és a cél távolsága, és az x_t és y_t a célkoordináták. A kettő összegéből kapták meg az eredőerő-vektort, R-t.

$$Q^{\pi}(s,a) = R(s,a) + \tau \sum_{s' \in S} P_{ss'} V^{\pi}(s') \tag{8}$$

Nem sokkal ezek után megcsinálták a Potential Field Method módosított változatát, hogy adott szinten kijavítsák annak hibáit [25]. Kutatásukban kifejtik az előbb felsorolt négy lehetséges hibát részleteiben. A lokális minimumhoz kapcsolódva leírjak, hogy a robot zsákutcába jut, például ha bekerül egy U alakú akadályba. Amikor két akadály között nem fér át a robot, mert az eredő taszító erő ellentétes irányba mutat, a cél pedig mögötte van, ahelyett, hogy átmenne a két akadály között, kikerüli. Megjegyzik, hogy e módszer alkalmazásának legnagyobb

korlátja, hogy ha változik a távolság a robot és az akadály között, akkor a mozgása instabil lesz. Ezeknek a hibáknak a kiküszöbölésére megalkották a Vector Field Histogram (VFH-) módszert.

6.7. A Vector Field Histogram módszer

A kutatásukban J. Borenstein és Y. Koren kifejti [27], hogy ebben a metódusban az adatcsökkentés két részletben zajlik, míg a VFF-ben csak egy volt. Ebben 3 adatreprezentációs szintet különböztetnek meg, amit a kutatásukban részletesebben kifejtenek:

a) a legmagasabb szinten van az a környezet, amelyben a robot tevékenykedik. Itt a koordináta-rendszer folyamatosan frissíti önmagát a fedélzeti szenzorok segítségével;
b) középső szinten egy H-val jelölt egydimenziós polár hisztogram képződik a robot pillanatnyi helyzete köré. Ez a H magába foglal *n* szögletes szektort, aminek α a szélessége. Majd a C* régiót egy transzformációval áttranszformáljuk ebbe a H-ba;
c) a legalsó szinten pedig a VFH-algoritmus kimenete van, a referenciaértékek a vezetésre (gyorsítás, lassítás) és a kormányzásra.

Azonban, mint minden metódusnak, ennek is vannak hiányosságai és korlátai. Ezek közé tartozik, hogy ez az algoritmus nem találja meg mindig az optimális megoldást, csak abban az esetben, ha egy teljes környezetet táplálunk a rendszerébe. Illetve ennél is előfordulhat olyan probléma, hogy zsákutcába ragad és ott köröz. Ezt viszont ki lehet azzal küszöbölni, hogy egy előre megadott szabályt definiálunk neki, bár ebben az esetben sem lesz optimális az út. Erre találták ki azt a megoldást, hogy van egy útmonitorozás, ami jelzi, hogy ha a robot zsákutcába kerül. Ezután beállít magának valamilyen elterelő módot, és addig nem fog törlődni ez az elterelés, amíg a robot újra meg nem látja a célt. Miután meg lett jelölve zsákutcának, a robot lelassít, esetleg megáll, amíg a VFH-algoritmust felfüggesztik. Egy Globális Úttervező (Global Path Planner) algoritmus átveszi az irányítást, hogy új utat tervezzen a hisztogramrácson elérhető információ alapján. Ezután ezt az utat beletáplálja a VFH-algoritmusba.

6.8. Mesterséges intelligencia

A mesterséges intelligenciának köszönhetően egy újfajta út is megnyílt a drónok pályatervezése felé. Ez pedig a Reinforcement Learning (RL), azaz megerősítéses tanulás. Lényege, hogy adott szituációkhoz rendelünk adott akciókat, végül ezeket az akciókat hatásosságuk alapján számszerűen értékeljük. Az RL-ben egy ügynök szerepel, akinek jó előre meghatározott célja van. A tanulásnak három fő komponense van: a környezeti modell, hogy mi hogyan helyezkedik el, mi hol van; az eljárásmódok, hogy menjen egyenesen vagy forduljon jobbra; illetve a jutalmak, amelyek attól függően, hogy milyen eljárást választott, és hogy az közelebb vitte-e a céljához, lehetnek negatívak vagy pozitívak. Jutalmat minden akció/eljárás után kap az ügynökünk. Ez a bizonyos ügynök egy olyan entitás, amelyet megkérünk, hogy csináljon valamilyen akciót az adott állapotától a múltbeli események alapján. Az RL célja, hogy az ügynök gyorsan megtanulja azt az eljárást, amivel a kiinduló állapotból el tud jutni a célállapotba. Egy tanulási folyamat az RL-ben a következőképpen néz ki:

a) az ügynök, jelen esetünkben a drón, választ egy akciót az elérhető akciók közül *t* időben, és elvégzi azt a környezetben;
b) az akció eredményeképpen a drón állapota megváltozik, ami annyit tesz, hogy a drón közelebb vagy távolabb kerül a céltól;
c) ezután kapja meg a jutalmat a célhoz való közelsége függvényében.

Az RL három fajtája:

a) az értékalapú megközelítésnél az ügynök célja, hogy megtalálja azt az eljárássorozatot, amely maximalizálja a pontjait;
b) az eljárásalapú megközelítés során az ügynöknek meg kell találnia az optimális értéket;
c) a modellalapú megközelítés úgy működik, hogy az ügynöknek adunk egy modellt a környezetről, vagy megkérjük az ügynököt, hogy tanulja meg a környezet modelljét, hogy utána feladatokat végezzen el abban a környezetben.

Amíg az első két variáció például egy mentőakciónál lehet sikeres, mert ott számukra ismeretlen helyen, környezetben kell manőverezniük, addig a harmadikat egy városban lehet optimálisan használni, ahol a város térképét a programba lehet táplálni, mert az nagyon ritkán fog változni. Az RL alapkoncepciója közé tartozik, hogy az irányítás zárt körfolyamatként működik, és a jutalmak jelentik a visszajelzést neki. Ha mindezt drónokra akarjuk alkalmazni, akkor felfogható módosított Markov-döntési folyamatként.

A legelterjedtebb és legtöbbet módosított változat a Deep Q-Network algoritmus, röviden DQN. Ez az algoritmus kiemelkedik a komplex problémák megoldásában. A Q függvény a következőképpen néz ki:

$$Q^{\pi}(s,a) = R(s,a) + \tau \sum_{s' \in S} P_{ss'} V^{\pi}(s') \tag{9}$$

ahol:

- π – az eljárás értéke;
- s – az adott állapot;
- α – az akció, amit tett a modell;
- τ – a csökkentési faktor.

Ez a rendszer úgy működik, hogy tudjuk a kezdő vagy éppen az adott állapotokat, *S* a helyzet, *R* a jutalom értéke, *P* a változási valószínűsége és *V* a használhatósági függvény (Utility Function). Miután ezek megvannak, a tanulást felbontjuk *t* időközökre, és minden lépésnél/minden időköznél egy új jutalomértéket számítunk, amíg el nem érjük az utolsó lépést, *T*-t. Mindezek után a maximum Q függvényt és a kumulált csökkentési faktort a következő képletekkel kell számolni [28]:

Maximum Q függvény:

$$Q^{\pi^*} = (s,a) = R(s,a) + \tau \sum_{s' \in S} P_{ss'} V^{\pi^*}(s') = \mathbb{E}[r + rmax\left(Q^{\pi^*}(s',a') \middle| s,a\right)] \tag{10}$$

Kumulált csökkentési faktor:

$$V^{\pi^*}(s) = max_{a \in A}[Q^{\pi^*}(s,a)] \tag{11}$$

Utóbbinak három fejlesztett változata van, amit használnak:

a) a Dupla DQN, ahol minden időben más értéket használ, hogy elkezdje az akciót.

$$Y_t^{Double} = r + \gamma Q(s_{t+1}, a_{t+1}, : \theta;)\theta^- \quad (12)$$

b) a küzdő DQN (Dueling DQN) esetén a hálózatot érték szerinti és előny szerinti hálózatokra bontjuk. Az értékhálózat megpróbálja kiértékelni a mostani és az összes állapotát a rendszernek. Miközben az előnyhálózat az elérhető akciók minőségét mutatja be, amit a következő egyenlet szemléltet:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + A(s_t, a_t; \theta, \alpha) - \frac{1}{A}\sum_{a^{t+1}} A(s_t, a_t; \theta, \alpha) \quad (13)$$

c) a Dupla Küzdő DQN (Double Dueling DQN) az előző kettő összeolvadásáról szól.

$$Y_t^{DDQ} = r_{t+1} + \gamma Q(s_{t+1}, argmaxQ(s_{t+1}, a_t; \theta, \alpha, \beta); \theta^-, \alpha^-, \beta^-) \quad (14)$$

Mivel a legtöbb esetben az UAV-kat olyan helyekre viszik, ahol a környezetük nem ismert, vagy csak részben ismert, olyan algoritmus kell, amely út közben tud tanulni és ezt automatikusan végzi. Ebben segít a mesterséges intelligencia és a Deep Learning, ezeknek a segítségével az UAV végig tud menni ismeretlen területeken is anélkül, hogy bármilyen ütközésveszély fennállna. A Deep Learning is az emberi viselkedésen alapul: az embernek sincs meg minden ismerete egy új környezetben, de az emlékeinek segítségével tud döntést hozni, hogy mit csinált hasonló helyzetben. A drónnál is azt kell elérni, hogy legyen valamilyen emléke, hogy tudja, adott helyzetekre hogyan kell reagálni [28].

Egy ilyen modellt hét fő változóval lehet leírni:

- *S* – a környezeti állapotot írja le;
- *A* – a lehetséges akciókat;
- *P* – egy akció valószínűsége a jelen és a múlt helyzetei, illetve döntései alapján;
- *R* – az UAV-hoz visszacsatolt jutalom, hogy milyen jól teljesített;
- *Ω* – a tényleges megfigyelések;
- *O* – a feltételes valószínűségi eloszlás;
- *δ* – a csökkentési érték.

Hogy bemutassa hatásosságát ennek a DQN-nek, két különböző kutatás is MATLAB segítségével szemléltette, hogy ez az algoritmus működőképes olyan környezetben, ahol nem minden ismert, és jól kiigazodik előre nem látható helyzetekben is. Kutatásaikban Xiaojian Hou és csapata létrehozott egy algoritmust, amelyben összekapcsolta a Q-Learning algoritmust, illetve a „potenciális mező"-metódust [5], továbbá Ender Cetin és csapata szimulálták a drónok navigációját ismeretlen környezetben megerősítéses tanulással párosítva [29]. Utóbbiak a kutatásban megállapították, hogy a drón a mesterséges intelligencia/mélytanulás segítségével képes kikerülni az álló és mozgó akadályokat, és keresztül tud menni az adott környezetben, mint például egy kisebb szomszédságban. A másik kutatócsapat is megállapította, hogy a legtöbb algoritmus nem elég effektíven alkalmazható dinamikusan változó környezetben. Kutatásukban a Q-Learning algoritmust használták, hogy a globális tervezést megcsinálják, de amikor valamilyen ismeretlen veszélyforrás jelent meg, vagy elakadt, akkor a „potenciális mező"-metódust használták. Kutatásukban megállapították, hogy ezzel a módszerrel a drón effektíven tud navigálni a környezetben.

7. Összefoglalás, következtetések, kitekintés

Kutatásom során megvizsgáltam, milyen kihívásokkal kell megküzdeni egyes pályatervező algoritmusoknak, hogy megfeleljenek a kritériumoknak. Bemutattam egyes megbízható katonai algoritmusokat, mint a Voronoi Roadmap algoritmus, egyes általános használatúakat, mint az A* algoritmust, illetve megvizsgáltam a legújabb típusú pályatervező algoritmusokat, mint a megerősítéses tanulásos algoritmus. A jövőben egyre pontosabb és jobb algoritmusokat fogunk tudni majd létrehozni a kettő kombinálásából, mint láttuk ebben a két kutatásban [5], [29]. Megállapításaim szerint egy mesterségesintelligencia-alapú (ebben az esetben megerősítéses tanulásos) algoritmus kombinálva a VHF, illetve a VHF+ sajátosságával jó módszer lehet, hogy még effektívebben tudjunk eljutni A-ból B-be, miközben figyel a drónunk a dinamikus környezetváltozásra, hirtelen felbukkanó akadályokra, és megtalálja a legrövidebb utat. További kutatásaim során ebbe az irányba fogok elindulni.

Felhasznált irodalom

[1] T. Amukele, „Using Drones to Deliver Blood Products in Rwanda," *The Lancet Global Health,* pp. e463–e464, 2022. Online: https://doi.org/10.1016/S2214-109X(22)00095-X

[2] A. A. Nyaaba, Matthew Ayamga, „Intricacies of Medical Drones in Healthcare Delivery: Implications for Africa," *Technology in Society,* 66. évf. 101624. 2021. Online: https://doi.org/10.1016/j.techsoc.2021.101624

[3] „The Verge," [Online]. Elérhető: www.theverge.com/sponsored/goldman-sachs-drones.

[4] „Statista," [Online]. Elérhető: https://shorturl.at/f0TuN

[5] H. Xiaojian et al., „A UAV Dynamic Path Planning Algorithm," in *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC),* pp. 127–131. 2020.

[6] J. F. Shortle et al., „Simulating Collision Probabilities of Landing Airplanes at Nontowered Airports," *Simulation,* 80. évf. 1. sz. pp. 21–31. 2004. Online: https://doi.org/10.1177/0037549704042028

[7] B. M. Sathyaraj et al., „Multiple UAVs Path Planning Algorithms: A Comparative Study," *Fuzzy Optimization and Decision Making,* 7. évf. pp. 257–267. 2008. Online: https://doi.org/10.1007/s10700-008-9035-0

[8] S. Aggarwal, N. Kumar, „Path Planning Techniques for Unmanned Aerial Vehicles: A Review, Solutions, and Challenges," *Computer Communications,* 149. évf. pp. 270–299. 2020. Online: https://doi.org/10.1016/j.comcom.2019.10.014

[9] R. Szabolcsi, „3D Flight Path Planning For Multirotor UAV," *Review of the Air Force Academy,* 18. évf. 1. sz. pp. 5–16, 2020. Online: https://doi.org/10.19062/1842-9238.2020.18.1.1

[10] R. Szabolcsi, „Multirotoros pilóta nélküli légijárművek háromdimenziós repülési pályáinak számítógépes tervezése és szimulációja," *Hadtudomány,* 30. évf. 4. sz. pp. 133–150. 2020. Online: https://doi.org/10.17047/HADTUD.2020.30.4.133

[11] R. Szabolcsi, „Flight Path Planning for Small UAV Low Altitude Flights," *Land Forces Academy Review,* 25. évf. 2. sz. pp. 159–167. 2020. Online: https://doi.org/10.2478/raft-2020-0019

[12] R. Szabolcsi, „Pilóta nélküli légi jármű kis magasságú repülési pályáinak tervezése," *Repüléstudományi Közlemények,* 32. évf. 1. sz. pp. 29–50. 2020. Online: https://doi.org/10.32560/rk.2020.1.2

[13] C. Goerzen, Z. Kong, B. Mettler, „A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems,* 57. évf. pp. 65–100. 2020. Online: https://doi.org/10.1007/s10846-009-9383-1
[14] H. Liu et al., „An Autonomous Path Planning Method for Unmanned Aerial Vehicle Based on a Tangent Intersection and Target Guidance Strategy," *IEEE Transactions on Intelligent Transportation Systems,* 23. évf. 4. sz. pp. 3061–3073. 2022. Online: https://doi.org/10.1109/TITS.2020.3030444
[15] Bortoff, Scott, „Path Planning for UAVs," American Control Conference, 2000. Proceedings of the 2000, pp. 364–368. 2000. Online: https://doi.org/10.1109/ACC.2000.878915
[16] Balampanis et al., *2017 International Conference on Unmanned Aircraft Systems (ICUAS),* 2017.
[17] M. Rhinehart, *Monte Carlo Testing of 2- and 3-dimensional Route Planners for Autonomous UAV Navigation in Urban Environments, Thesis (M.S.)* University of Minnesota, 2008.
[18] [Online]. Elérhető: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/basicmotion.html.
[19] Blasi et al., „Path Planning and Real-Time Collision Avoidance Based on the Essential Visibility Graph," *Applied Sciences,* 10 évf. 16. sz. p. 5613. 2020. Online: https://doi.org/10.3390/app10165613
[20] C. Xia, C. Xiangmin, „The UAV Dynamic Path Planning Algorithm Research Based on Voronoi Diagram," *The 26th Chinese Control and Decision Conference (2014 CCDC),* pp. 1069–1071. 2014.
[21] I. W. Geoffrey, C. Sammut, *Encyclopedia of Machine Learning.* Boston, MA: Springer US, 2010. Online: https://doi.org/10.1007/978-0-387-30164-8
[22] V. Jeauneau, A. Kotenkoff, L. Jouanneau, „Path Planner Methods for UAVs in Real Environment," *IFAC-PapersOnLine,* 51. évf. 22. sz. pp. 292–297. 2018. Online: https://doi.org/10.1016/j.ifacol.2018.11.557
[23] F. Daniel et al., „A Systematic Literature Review of A* Pathfinding," *Procedia Computer Science,* 179. évf. pp. 507–514. 2021. Online: https://doi.org/10.1016/j.procs.2021.01.034
[24] J. Borenstein, Y. Koren, „Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *Proceedings – IEEE International Conference on Robotics and Automation,* pp. 1398–1404. 1991.
[25] Online: http://www-personal.umich.edu/~johannb/vff&vfh.htm
[26] J. Borenstein, Y. Koren, „The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation,* 7. évf. 3. sz. pp. 278–288. 1991. Online: https://doi.org/10.1109/70.88137
[27] T. Ahmad et al., „Drone Deep Reinforcement Learning: A Review," *Electronics,* 10. évf. 9. sz. p. 999. 2021. Online: https://doi.org/10.3390/electronics10090999
[28] H. Xiaojian et al., „A UAV Dynamic Path Planning Algorithm," in *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC),* pp. 127–131. 2020. Online: https://doi.org/10.1109/YAC51587.2020.9337581
[29] Cetin et al., „Drone Navigation and Avoidance of Obstacles Through Deep Reinforcement Learning," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC),* 2019, pp. 1–7. Online: https://doi.org/10.1109/DASC43569.2019.9081749

The UAVs Path Planning Challenges and Possible Solutions

During my research I analysed the problems and the challenges of the UAV path planning. I am going to demonstrate the most common problems, which can come across during path planning. These problems include the Point Vehicle problem or the Jogger's Problem. I am going to present the state-of-art path planning algorithms and solutions like Visible Graph or A.*

Keywords: *Path planning, A* algorithm, Q-Learning, UAV*

Mihályi Géza	Géza Mihályi
hallgató	Student
Óbudai Egyetem	Óbuda University
Bánki Donát Gépész és Biztonságtechnikai Mérnöki Kar	Bánki Donát Faculty of Mechanical and Safety Engineering
mihgeza2000@gmail.com	mihgeza2000@gmail.com
orcid.org/0009-0005-4636-0516	orcid.org/0009-0005-4636-0516