

Óbudai Egyetem

Bánki Donát Gépészeti és Biztonságtechnikai kar

Az UAV pályatervezés kihívásai és lehetséges megoldásai
The UAVs path planning challenges and possible solutions

Konzulens:

Prof. Dr. habil. Szabolcsi Róbert
egyetemi tanár

Tartalom

Bevezetés.....	3
1. Problémafelvetés, motiváció.....	4
2. Szakmai – tudományos előzmények	4
3. A pályatervezés kihívásai.....	5
3.1. Pontszerű Test Probléma.....	8
3.2. Kocogó probléma	8
3.3. Bogár probléma	8
3.4. Az időben változó tér probléma	8
3.5. Sétáló Problémát.....	8
4. Pályatervezés lépés	8
5. Autonóm járművek (UGV, USV, UAV) általános pályatervezése.....	10
6. A Pályatervezés egyes megoldásai	13
6.1. Sejtekre lebontó algoritmus (cell decomposition method)	13
6.2. Visible Graph - Látható Gráf	14
6.3. Voronoi Roadmap.....	15
6.4. A-Star alapú algoritmusok	16
6.5. Potenciális mező metódus (Potential Field Method)	17
6.6. A Virtuális erő mező metódus (The Virtual Force Field (VFF) Method)	18
6.7. A Vector Field Histogram módszer	20
6.8. Mesterséges intelligencia	21
7. Összefoglalás, következtetések, kitekintés	24
Felhasznált irodalom.....	25

Bevezetés

Unmanned Aerial Vehicles vagy UAV¹-kat már használtak az 1800-as évek közepén, de akkor még nem úgy néztek ki mi a mai elterjedt drónok, hanem főleg hőlégballonok voltak, amiket felderítésre és gyakorlásra használtak. Térjünk vissza időben a 2013-as évbe, amikor is a drónok kezdtek elterjedni a köztudatban. Az Amazon cég bejelentette, hogy drónokat fog használni a termékek szállításához. Ez a mai napig még nem történt meg, az Amazon csak annyit jelentett be, 2022. június 13-án, hogy hamarosan elkezdik. Ettől függetlenül, hogy a szállítmányozásban még nem terjedt el, arra is láthatunk példát Afrikában, hogy drónokkal oldják meg a vér, illetve eszközszállítást. Ez 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 drón piac 2020-ra eléri a 100 milliárd dolláros 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 drón piac eléri a 26,3 milliárd dolláros értéket. A 2026-os előrejelzések szerint ez az érték elérheti az 41,3 milliárd dollárt [4]. Ahhoz, hogy az ilyen és ehhez hasonló drónok, akár szállítási, akár katonai céloknak megfeleljenek kell, hogy legyen bennük önvezető képesség, aminek egy fontos része a pályatervezés. A cikk felépítése a következő:

- A. 1. fejezet - Problémafelvetés, motiváció;
- B. 2. fejezet - Szakmai - tudományos előzmények;
- C. 3. fejezet - A pályatervezés kihívásai - Pontszerű Test Probléma, Kocogó Probléma, Bogár probléma, Súlyozott Régió probléma, Az időben váltakozó probléma, Sétáló probléma;
- D. 4. fejezet - Pályatervezés lépései;
- E. 5. fejezet - Vizsgált pályatervezési eljárások és módszerek - Sejtekre lebontó algoritmus, Visible Graph, Voronoi Roadmap, A* alapú algoritmusok;
- F. 6. fejezet – Konklúzió.

¹ Pilóta nélküli repülőgép

1. Problémafelvetés, motiváció

Ahhoz, hogy a drón eljusson A pontból B pontba egy jól meghatározott utat kell követnie. Miután ezt az utat meghatározta folyamatosan frissítenie kell ezt mivel a városi vagy akár a katonai terep egy kiszámíthatatlan terep. A drón megtervez egy olyan utat, ahol a környezetét részben vagy egyáltalán nem ismeri, emiatt folyamatosan alkalmazkodnia kell hozzá. A legtöbb esetben ilyenről beszélünk, a való életben ritkán van statikus közeg, ahol az UAV az előre meghatározott úton tudna haladni. Egy ilyen dinamikusan változó környezetre hozott egy megoldást Xiaojian Hou és csapata [5], amiről a későbbiekben részletesebben szó lesz, akik a mesterséges intelligencia, azon belül is a megerősítéses tanulást hívták segítségül, abból is a fejlesztett Q-Learning algoritmust. Ahhoz, hogy a drónokat elfogadja a társadalom biztonságossá/biztonságosabbá kell tenni őket. Városon felüli repülésnél több komolyabb baleset is történhet, ami rosszabb esetben ember életeket is követelhet, jobb esetben magántulajdont károsíthat. Emiatt, hogy minden ilyen nemű balesetet minimalizálni tudjunk Quan Shao és csapata készített egy cost-benefit assessment modelt,² ami vizsgálja a veszély lehetőségeit és a szolgáltatás előnyeit [6]. Ezen kívül általános értelemben az egyik leghasznosabb ütközés elemző modell a The Longitudinal Reich Collision Risk Model,³ amit az előbb említett kutatásban is használnak, és használják, nem csak drónok esetén, arra, hogy megállapítsák mekkora az esélye annak, hogy 2 légi jármű ütközik egymással. Amit egy kutatócsoport fel is használt, hogy kiderítse, mekkora valószínűséggel ütköznek össze a repülők leszálláskor [7]. Motivációmként szolgál, hogy mindezen ismeretekkel, mind a saját programozói tudásommal együtt szimuláljak egyes algoritmusokat és megpróbáljam továbbfejleszteni őket.

2. 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, amiben összehasonlítottak több algoritmust is, köztük a Distance vector algoritmust⁴, a Floyd-Warshall's algoritmust, illetve a manapság egyik legjobban elterjedtet, az A* algoritmust. Ebben

² Költség-haszon értékelési modell

³ Hosszúsági ütközési modell – ennek a modellnek a segítségével határozzák meg, hogy mekkora eséllyel ütközik össze 2 légi jármű.

⁴ Távoltság-vektoros algoritmus – Ahol is az egyes csomópontok tudják a távolságot közöttük és az összes többi lehetséges csomópont között.

megállapítják, hogy BFS5 akkor is abba az irányba halad tovább a cél felé, amikor már látszódik, hogy nem az legrövidebb irány, illetve, hogy az A* algoritmus mindig megtalálja a legrövidebb utat [8]. Egy 12 évvel későbbi kutatásban Shubhani Aggarwal és Neerej 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. Ebben a kutatásban kitérnek arra, hogy milyen jövőbeli kutatásokat lenne célszerű követni [9]:

- a) Habár a pályatervezés hatékonyságára már több kutatás is készült, még mindig tele van kihívásokkal és lehetőségekkel;
- b) Talán az egyik legfontosabb 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 a drónok az információt.

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

3. A pályatervezés kihívásai

A szakirodalom alapján a pályatervezés a legfontosabb eleme a szoftveres rendszernek és ennek kutatása, hogy megtalálják az ideális utat a kiinduló helyzet és a cél helyzet 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 már drónokkal. A sima 2D-s környezeti ábrázolás nem lenne megfelelő, mert nem lehetne pontosan elhelyezni a környezetben a tárgyakat, emiatt az kell, hogy legyen egy 3D-s komplex kép a környezetről. A pályatervező algoritmus eltérő lehet kereskedelmi drónoknál, illetve katonaiaknál. Míg a kereskedelmi

⁵ Breadth-First Search – Keresési algoritmus, ami először az adott csúcshoz legközelebbi csúcsokat vizsgálja át, amik egy szinten vannak vele, és csak utána folytatja a keresést mélyebben. Szélességi keresésnek hívják.

forgalomban lévő drónoknál, illetve a szállítási drónoknál 3 alapelem van, a Motion planning⁶, a Trajectory planning⁷ és a Navigation⁸. A katonai drónok abban térnek el, hogy náluk több bemeneti paraméter van, amit számításba kell venniük, amivel egy több célú optimalizálási probléma jön létre.

A fő célja a pályatervező technikáknak, hogy a számítási szükségletet és a számítás idejét lecsökkentsék, amíg megtalálja az optimális utat. Optimálisnak kell lenniük olyan szempontból is, hogy a minimum energiát fogyassza, rövidebb időbe kerüljön, amíg elér a célig és nem ütközik össze közben más drónokkal vagy környezeti tárgyakkal. A Fő kihívási a pályatervező algoritmusnak a magába foglalja a következőket [9].

- a) Pálya hossza: A pálya hossza az a drón által megtett távolságot jelenti, amíg eljut a kiinduló állapotból a végállapotba;
- b) Optimalitás: Ez azt jelenti, hogy a teljes rendszernek optimálisnak kell lennie, energia-, ár- és egyéb elvárás tekintetében. 3 módon lehet az optimalitást kifejezni: 1. optimális, 2. sub-optimális, 3. egyáltalán nem optimális;
- c) Teljesség: Akkor nevezzük teljesnek, ha a pályatervezésnél meghatározott feltételeket teljesíti, ha létezik olyan út. Ezzel egy megoldást rak a drón elé, amit követnie kell;
- d) Költséghatékonyság: Több összetevőből áll, és függ attól a teljes számítási költségtől. Összetevői közé tartozik az üzemanyag ára, akkumulátorok töltésének a költsége, szoftver és hardver árak;
- e) Időhatékonyság: Azt jeleneti, hogy a drón megteszi az adott utat a kiinduló ponttól a végpontig ütközés nélkül a minimális idő alatt;
- f) Energiahatékonyság: Energiahatékonyság azt jelenti, hogy a drón a lehető legkevesebb energiát használja, miközben teszi meg útvonalát és számolja a röppályáját a művelet közben.;
- g) Ütközés kerülés: Az a képesség, hogyha a röppályájába belekerül valami, akkor azt kikerülje és folytassa az útját, anélkül, hogy károsodna a szerkezet vagy az elektronika.

⁶ Pálya útjának a megtervezése

⁷ Az útnak megtervezése időben és térben, itt már kalkulálunk a jármű sebességével is

⁸ A kettő egybe illesztése, amikor is már a levegőben van teljesítse a rábízott utat a megadott sebességgel

Ezeket még mind nehezíti az is, hogy a legtöbb környezeti tárgyról csak hiányos információink vannak, amik főleg igazak a dinamikusan mozgó tárgyakról, ilyenek lehetnek az emberek, autók, más drónok.

Ezekkel a tárgyakkal a drónok tervező szoftverének 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 megtudja határozni, hogy az előtte lévő mozgó tárgy milyen messze van, merre mozog.

Meg kell még küzdeni a drónok kinematikai és dinamikai kötöttségükkel is. Az is megállapítható, hogy minden szituáció más, amiatt nagyon nehéz egy olyan algoritmust írni, ami megfelelné mindegyiknek. Emiatt minden feladathoz külön, ahhoz specifikus algoritmust kell írni. Mivel a környezet, amiben működik a drón ismeretlen emiatt az egyik megoldás, hogy használjunk valamilyen sejtekre lebontó algoritmust⁹.

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

Az UAV irányítási problémát háromdimenziós problémátérrel, limitált környezeti információval, limitált távolságú szenzorokkal, sebesség és gyorsaság kényszerekkel és a szenzor adat bizonytalanságával szokás jellemezni.

Az alábbi publikációban [14] a kutatók meghatározták, hogy milyen problémákkal találkozhatunk. Az egyik a statikus, ahol is a környezetet teljesen ismerjük, a másik pedig a dinamikus, ahol is a környezetet nem ismerjük teljesen vagy változik, ahogy haladunk előre az időben. Amikor a tárgyak fix helyen vannak az térben, akkor időfüggetlennek nevezzük őket, amikor tudnak mozogni akkor időfüggőnek. 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).

⁹Ezt a külföldi források cell decomposition method-nak hívják.

3.1. A 'pontoszerű test probléma'

Ebben a problémában a jármű egy pontként van modellezve a térben. Itt a konfigurációs tér az megegyezik az effektív térrel, amiatt nem kell semmilyen különleges cselekményt csinálni, szóval ez az egyik legegyszerűbb probléma, és az optimalitás megegyezik a kezdő és a start pont távolságával.

3.2. A 'kocogó probléma'

Ez a probléma egy kocogó dinamikai problémájával foglalkozik, amikor is a kocogónak limitált a látási tere. Ez a probléma azt reprezentálja, amikor egy jármű/drón egy változó, időfüggő környezetben mozog, limitált szenzor távolsággal. Az optimalitását úgy lehetne meghatározni, hogyha minimalizálja a repülési időt a kiinduló helyzet és a cél között, vagy minimalizál valamilyen másik attribútumát, mint például az energiafogyasztását.

3.3. A 'bogár probléma'

Ez a probléma egy különleges változata a Kocogó problémának, amikor is a betekintési szöge a járműnek 0. Ilyenkor az járműnek meg kell érinteni vagy nagyon közel kell kerülnie a tárgyakhoz, hogy érezze őket.

3.4. Az 'időben változó tér probléma'

Ebben a problémában a járműnek el kell kerülnie azokat a tárgyakat, amik időben mozognak. Ennek az optimális megoldása az, hogy minimalizáljuk az megtételéhez szükséges időt vagy az úthosszt.

3.5. A 'sétáló probléma'

Ez az a probléma amikor egy akadályokkal teli "mező"-n kell keresztülmenni, hogy elérjük a célt. Ilyenkor a járművet egy merev testnek vesszük. Feltételezzük, hogy az objektumnak nincs semmilyen dinamikai kötése. Ez a probléma a komplexitását vizsgálja a járműnek hozzáadva az akadálymező komplexitását, majd az ebből jövő számot m vagy M számnak nevez.

4. A pályatervezés lépései

A kereskedelmi és szállítási drónoknál már említett 3 dolog közül a Motion Planning felel az út optimalizálásért, hogy minimalizálja az út hosszt, a fordulási szöget és a maximális repülési távolságot, amit meg tud tenni az adott üzemanyaggal. A Trajectory Planning kiegészíti a 'Motion

Planning' részt azzal, hogy a tervezett úthoz hozzárendel értékeket, mint a sebességet, időt, illetve az UAV kinematikáját. A Navigation olyan részekből áll, mint a Motion Planning, Trajectory Planning, ütközés elkerülés és helyzetmeghatározás (lokalizáció). Az alap tervezés 2 fázisból áll.

Az első fázis az úgynevezett pre-processing¹⁰, azaz előkészítő számítások. Ebben a fázisban a már említett 3D-s környezetre rárajzoljuk a szükséges pontokat és vonalakat, majd a legeneráljuk a gráfokból álló térképet [9].

Habár az előző publikációban azt fejtegették, hogy nem lehet vagy nem olyan pontosan lehet 2D-s környezetben pontosan elhelyezni a tárgyakat, addig egy másik publikációban amellet érvelnek, hogy elegendő 2D-ben megtervezni és a 3D-s környezet pedig műveleti résznél kell figyelembe venni. Ebben a publikációban a drón különböző útvonalpontok között közlekedik.

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 is az l_i az a különböző waypontok közötti távolság, az L_{max} a maximális távolság, az m pedig a útvonalpontok számát jelöli. Az ütközés mentes 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 következő [15]

- a) x_i, y_i : Az adott waypontok koordinátái;
- b) a, b : A két féltengelye az ellipszisnek;
- c) θ : A dőlése a féltengelynek;
- d) x_k, y_k : Az adott tárgyak középpontja;
- e) $r_{biztonságos}$: A minimális távolsága a drónnak és a különböző természeti tárgynak.

¹⁰ Előkészületek. Amikor a nyers adatokat tisztítjuk és gép számára értelmezhető állapotba hozza.

Minden ilyen pályatervező technika és algoritmus ezek a pontok és vonalak alapján határozza meg az ideális pályát. A második fázis az úgy nevezett lekérdezés (query) fázis. Ebben a fázisban különböző algoritmusok futnak le, olyanok, mint az ant colony¹¹ vagy a Floyd-Warshall¹²

A katonai drónoknál több kitétel van, mert azok kivannak téve az ellenséges erőknek is, akik leakarják lőni, emiatt, hogy láthatatlan maradjon az ellenség számára, el kell, hogy kerüljék a radarokat szóval 'láthatatlanná' kell válniuk.

1. Emiatt az első lépése az, hogy először egy úgynevezett rejtett utat (stealthy path) kell kialakítania. Ennek a rejtett útnak a lényege, hogy úgy kormányozza a drónt, hogy annak a radar jelét az ellenséges radarok ne tudják befogni. Ez azért nehéz, mert nem minden drónnak ugyanolyan a radar jele, ami azt jelenti, hogy a radar által kibocsátott sugárzást nem ugyanúgy verik vissza, hanem az egyik irányba jobban veri vissza, mint a másikba. Amikor a visszavert jelben valamilyen tüske van, akkor az algoritmus azt irányítsa el a radaroktól;
2. Másodjára az algoritmusnak meg kell felelnie, hogy a pálya útja minimális legyen úgy, hogy számításba veszi ezt a 'rejtett utat és közben kielégíteni a repülő dinamikai kényszereit is;
3. Harmadjára pedig ennek a szoftvernek meg kell felelnie annak, hogy mi a misszió célja. Ez azt jelenti, hogy a legtöbb esetben egy akció több drónos támadást tartalmaz magában, szóval a szoftvernek figyelnie kell arra, hogy a drónok egyszerre érjenek célba;
4. Ennek a szoftvernek a magán az UAV-n processzorán kell futtatni, amiatt, hogyha valamiért a röppályát újra kell tervezni, akkor ezt valós időben (Real-Time) tudja megtenni.

[16]

5. Az autonóm járművek (UGV, USV, UAV) általános pályatervezése

Ebben a fejezetben még mielőtt kitérek az UAV-k pályatervező megoldásai szeretném bemutatni, hogy milyen általános megoldásai vannak egyes pilóta/vezető nélküli rendszereknek, illetve ezeken

¹¹ Hangyakolónia: Ez az algoritmus egy valószínűségszámító módszer, amelyek a grafikonok vagy gráfokon keresztül megtudják találni a jó utakat. A mesterséges hangyák az összes lehetséges megoldást reprezentáló paraméterben mozogva keresik meg az optimális megoldásokat.

¹² 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 csúcspár közötti legrövidebb útvonalra vonatkozóan, amíg a becslés nem lesz optimális.

belül egyet-egyét bemutatni. Ugye az UAV, amiről fentebb már volt szó az a Unmanned Aerial Vehicle-nek a rövidítése, ami azt jelenti, hogy pilóta nélküli légi jármű. Az UGV az a Unmanned Ground Vehicle-nek a rövidítése, ami a pilóta nélküli földi járműnek felel meg. Az ilyen típusúak a földön dolgoznak, illetve végzik a tevékenységüket. Őket használják felderítési-, illetve a kutatási és mentés feladatoknál is.

Amikor az UGV-ket tervezzük figyelembe kell vennünk, hogy milyen terepen fog dolgozni az eszköz, ugyanis a rossz időjárás, illetve a talaj nagyban hozzá fog járulni, hogy mennyire lesz sikeres a küldetés. Fontos még úgy tervezni a szoftverét, hogy tudja változtatni a pályáját, illetve sebességét annak érdekében, hogy elkerülje az ütközéseket.

Az USV az Unmanned Surface Vehicles kifejezésből származik, ami annyit jelent, hogy pilóta nélküli vízfelszíni járművek. Ezek általában olyan hajók, amik a víz felszínén mozognak. Ezeket olyan helyzetekben szokták alkalmazni, ami az ember számára kockázatos lehet és mivel ezek az eszközök viszonylag kompaktak, sok helyre beférnek és ezen felül még kevés a fenttartási költségük.

Először ezeket katonai céllal kezdték el használni megfigyelésre, azóta civil élet beli alkalmazása is van. Ezeket is használják kutatás és mentés feladatokra, de ugyanúgy autonóm szállításra is. Fontos, hogy megfelelő szenzorokkal legyen ellátva ugyanis ez 2 környezet között mozog, a víz, illetve levegő között. Mert ugyanúgy kell érzékelnie a víz alatti objektumokat, hogy ne ütközzön bele, illetve kell, hogy érzékelje, ha szárazföldre ér, hogy milyen messze is van. [17]

Általánosságban elmondható, hogy ezt a pályatervezést fel lehet bontani 2 részre, egy globálisra, illetve egy lokálisra. A globálisnak a nagy távolságokban gondolkozik emiatt lassabb is. Ezt úgy kell elképzelni mintha egy pályatervező szoftvernek beadnánk egy város vagy egy ország térképet vagy egy bolygónak a domborzatát, megadjuk neki az kezdő állapotot, illetve a vég állapotot és ő ezen paraméterek segítségével nagyságrendileg megadja, hogy hova is kéne eljutni. Nem foglalkozik olyannal, hogy egy fa ki van dőlve útra vagy éppen egy eddig fel nem térképezett kráter van a földben. Ezek az utóbbiak a lokális pályatervező rész feladata. Ahogy megy a szerkezet a globális pályatervező által meghatározott úton, az ilyen apróbb hibákat a lokális pályatervező kikerüli és visszatér a pályájára.

Ezek után a következő két lépés, hogy az összegyűjtött információkat betesszük egy konfigurációs térbe, majd ezután valamilyen kereső algoritmussal megoldjuk a pályakeresési problémát a

megadott feltételeinkkel. Három különböző fajta konfigurációs mező van, a Sejtekre osztó (Cell Decomposition), az Útiterv (Roadmaps) és a Potenciális mezők (Potential Fields).

Ezek közül a mind a háromnak az alkalmazásával a későbbiekben fogok többet foglalkozni. A sejtekre osztó konfigurációs térről el lehet mondani, hogy ez a legtöbbet használt és tanulmányozott főleg a kültéri robotikában. 3 csoportosítási módot tudunk ezeken belül létrehozni, a Hozzávetőleges osztódásos (Approximate Decomposition) tér, az Adaptív selt osztódásos tér (Adaptiv Cell Decomposition) és a pontos sejt osztódásos tér (Exact Cell Decomposition).

Az útiterv metódusokat is több irányba lehet bontani, mint a Látható gráfok (Visible Graphs), a Voronoi diagramok (Voronoi Diagrams) és még két olyanra, amikre ebben a dolgozatomban részletesebben nem térek ki. Az egyik a Valószínűségi útitervek (Probabilistic Roadmaps, továbbiakban PRM) ami egy továbbfejlesztett változata a Voronoi diagramoknak, mivel azokkal ellentétben ezek nem a teljes keresési téren próbálják megoldani a feladatot, hanem folyamatosan egy véletlenszerűen kiválasztott részt kutat át és ezek alapján illeszti össze a végleges utat.

A másik pedig a Véletlenszerű fák gyors felfedezésre (Rapidly Exploring Random Trees, a továbbiakban RERT), ahol is ezek a RERT-ek a PRM-eknek egy változata, itt van egy kiindulási pont és elkezd véletlenszerűen terjeszkedni egy irányba. Ezt lehet úgy is csinálni, hogy a kiinduló és végpontból egyszerre indítjuk el ezeket a RERT-eket és ezek valahol találkozni fogni, ennek köszönhetően már lesz egy út amin a robotunk végig tud menni, de ez nem feltétlenül lesz optimális.

Miután a világ leképezése megtörtént valamelyik műveletnek az egyikével, utána az útkereső algoritmusok segítségével megkeresik az optimális utat. Ilyen algoritmusok lehetnek a már kicsit régebbi, kevésbé szofisztikált algoritmusok, mint a Dijkstra algoritmus és a mélységi keresés, illetve modernebbek, amik heurisztikát vagy neurális hálózatokat használnak, hogy felgyorsítsák a keresést. Ezek az algoritmusok közé tartozik az A* algoritmus, illetve megerősítéses tanulós algoritmusok (Reinforcement Learning) amiket a későbbiek jobban kifejtek, illetve a D* algoritmus, ami az A*-nak egy olyan továbbfejlesztett verziója, ahol is a világot nem teljesen ismerjük és az új információk folyamatosan jönnek elő. [18]

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 egy hibrid megosztási technikát alkalmaztak, hogy a lefedettségi területet pontos formákra bontották, háromszögekre. Mások hasonló elvek mentén haladtak, de ők spirál-szerű formákra bontották fel az adott komplex lefedettségi területet [17].

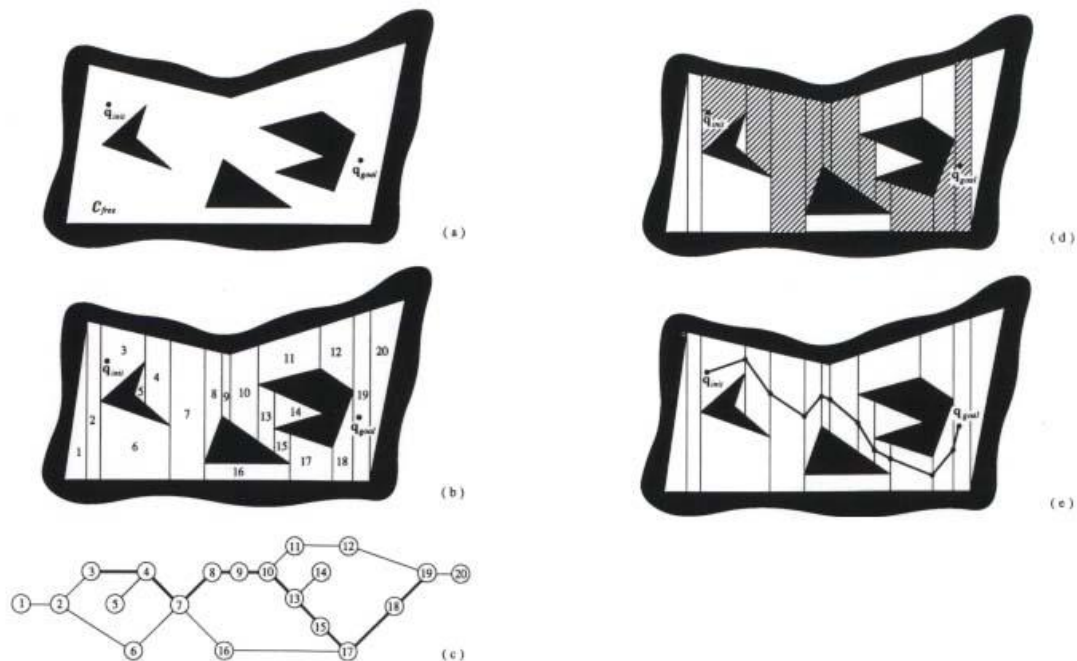
Azonban, ahogy feljebb már írtam, az, hogy milyen algoritmust választunk, az attól függ, hogy milyen problémát kell megoldanunk. Például, ha azt szeretnénk, hogy az adott drón átrepüljön a sivatag felett, vagy az alatta lévő utat kövesse, amiben esetleg kisebb kanyarok vannak és nincsen zavaró tényező, akkor sürgősséggel, hogy egy algoritmust használjunk, ami belekalkulálja a dinamikus változásokat, mert csak a számítási igényeket növeli.

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, amit N számnak neveznek.

Arról, hogy milyen mértékeket célszerű alkalmazni, arról Rhinehart írt ebben a kutatásában [18]. Annak érdekében, hogy egy algoritmus 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 energia felhasználásra és a megtett távolságra. Kell egy megbízható, valós időben történő számítás is, amiben nem történnek váratlan késések, mivel csak úgy lehet biztosítani a műveleti kritériumokat. Emiatt minél kisebbnek kell lenni a számítási komplexitásnak, minél gyorsabb az algoritmus, annál gyorsabban tud reagálni a kialakult helyzetre és változtatni az irányát, műveletét. A mostani State-of-Art megoldások.

6.1. A 'sejtekre lebontó algoritmus (cell decomposition method)'

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



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

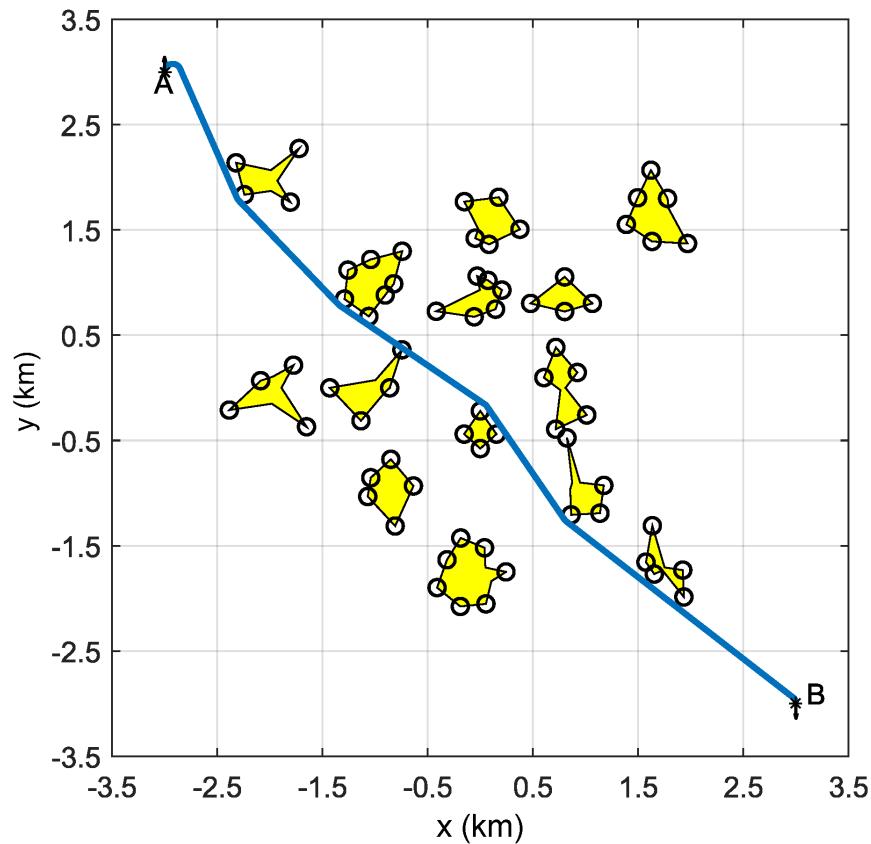
Ennek a lépései kifejtve a következők:

- Először fel kell bontani a szabad teret kisebb részekre, ebben a példában sokszögek által van határolva. Ezeket felbontjuk háromszögekre, illetve négyszögekre. Ezt úgy tesszük meg, hogy párhuzamosokat húzunk a belső alakzatok csúcsaihoz;
- Utána minden sejt, felbontott rész, kap egy számot, ami mostantól egy csomópontként lesz ábrázolva a kapcsolati gráfban;
- Azok a csomópontok, amelyek szomszédosok egymással a feldarabolt képen azokat összekötjük;
- A kapcsolati gráfból kijön egy út, amit fel lehet vetíteni a feldarabolt képre. A csomópontokhoz tartozó sejteket besatírozva jelölték.
- Ezután a sejthatárolók középpontját összekötötték és ebből megalakult az az út, amivel el lehet jutni a kezdőponttól a végpontba.

6.2. A 'Visible Graph - Látható Gráf' módszer

Ez a megoldás kifejezetten a Pontszerű test problémájának a megoldása. Viszont ezt csak 2D-s környezetben lehet számítani. Ebben az út úgy alakul ki, hogy az algoritmus "súrolja" a csúcsait a sokszögeknek, majd ezekből egy útvonaltervet csinál azokból a vonalakból, úgy, hogy minden

csúcsot összeköt minden csúccsal, amit lát. Mivel a minimális út nagyon közel kerül az akadályokhoz többször is az út folyamán, emiatt nem vagy alig garantál valamilyen baleset megelőzést olyan környezetnél, ahol valamilyen bizonytalanság van az akadályok pozíciójában. Az illusztráció a következő kutatásban szerepel [20].



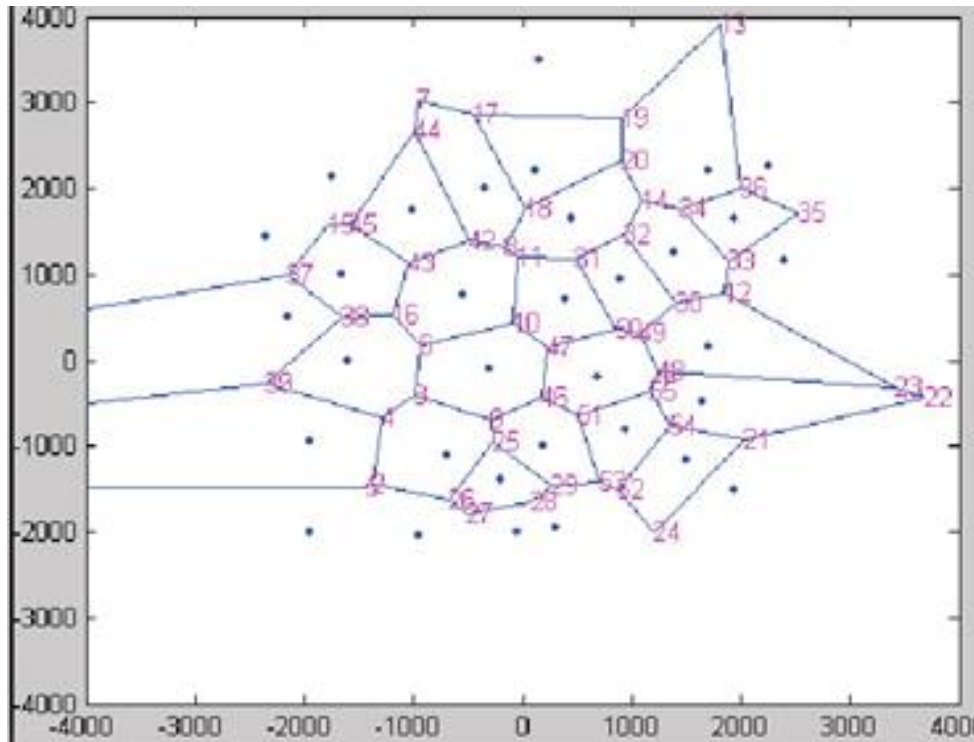
2. ábra Visible Graph, azaz Látható él módszer, amikor drón egyik élpontból megy a következő látható élpontba [20]

6.3. A 'Voronoi Roadmap' módszer

Ez a fajta algoritmus épít egy vázat, ami maximális távolságra van az akadályoktól majd megtalálja a minimális távolságot, amin tudja követni ezt a vázat. Ez az algoritmus is 2 dimenziós, viszont ezt próbálták áttenni 3D-be, ami sikerült is, de nem optimális.

Több kutatás is azzal foglalkozott, hogy olyan hierarchikus Voronoi gráfot építsen, amit általánosítani lehet több dimenzióra is. A diagram felépítése a következő: adott '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 kéne használnia ahhoz, hogy optimális legyen az útja, illetve, hogy elkerülje a veszélyeket katonai környezetben. Ezt a megoldást gyakran használják a katonai célokra. Az illusztráció a következő kutatásban jelent meg [21].



3. ábra Radar veszélyek Voronoi diagramja [21]]

6.4. Az 'A*' alapú algoritmusok

Ez az algoritmus régóta használatban van a pályakeresést kutató közösségek között. Ezt főleg egyszerűségének és könnyű változtathatósága miatt van így, ez a nagy előny egyes más algoritmusokkal szemben.

Ezt az algoritmust még amiatt is gyakran használják mert nem nagyon komplex, cserébe viszonylag nagy tárhelyre van szüksége, hogy tárolja az adatokat, amit használ. Ez egy determinisztikus algoritmus, amit először arra használtak, hogy megtalálja a legrövidebb utat egy 2D-s négyzetrácsban. Ez egy iterációs algoritmus, ami a csomópontokból mindig arra 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 $x = (x_1, x_2, \dots, x_n)$ és $y = (y_1, y_2, \dots, y_n)$ között n -dimenziós térben a pontok távolságának az összege minden dimenzióban.

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

Ezt azért hívják Manhattan távolságnak, mert ez az a távolság, amit egy autó megtesz 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 a heurisztika kielégíti a

$$h(x) \leq d(x, y) + h(y) \quad (4)$$

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 emiatt több kiegészítő algoritmus is kellhet hozzá.

Ez főleg akkor jön elő, amikor több ügynökös útkeresési problémát kell megoldani, mert ilyenkor gyakran előfordul, hogy a 2 ügynök ütközik egymással a nem megfelelő út miatt. Emiatt a mostani kutatások az A^* -al kapcsolatban arról szólnak, hogy lehetne az A^* -t módosítani, hogy tudjon kezelni ügynököket is.

Ezt egy újabb megoldással próbálják megoldani, ami az állatvilágból ered, ennek a neve a raj intelligencia (swarm intelligence). Erre már példát is láthattunk az Inteltől, amikor az égre küldtek 500 drónt és egy fényjátékot mutattak be. Ez módszer bebizonyította, hogy csökkenti a számítási igényt, azonban ez sem egy általános megoldás.

6.5. Potenciális mező módszer (Potential Field Method)

Az előzőektől kicsit eltérően, ámbár kicsit az A^* algoritmushoz hasonlítva megjelent a Potential Field Method is. Ennek egy 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 a lehetőségeit, realitásait.

Ebben a módszerben 2 fajta erő lép fel. Van a vonzó erő, amit a célok és állomások bocsátanak ki és vannak a taszító erők, amiket pedig az akadályok adnak. Ez is hasonlóan az A^* -hoz is egy közkedvelt algoritmus kutatásokban, mivel egyszerű és magas biztonsági szinttel dolgozik és ez használható valós időbeli alkalmazásra is mivel nincsenek benne bonyolult matematikai képletek, emiatt a számítási idő is kevés.

A módszer hatékonyságának vizsgálatára Krogh és Thorpe 1986-ban felállítottak egy általános potenciális mező metódust, amiben is kombinálják a globális és a lokális út tervezést. Ezen kívül 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 egy pontszerű testnek tekintjük, amire hatnak a külső pozitív és negatív erők. Ezeknek az erőknek az összeadásával megkapjuk az úgynevezett eredő erőt, aminek az irányába a drónunk el fog indulni. Viszont ennek az algoritmusnak a nagy hátránya, hogy könnyen tud a lokális minimumban beleesni 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 ütközni és ez az egész egy folytonos út 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 ebben az erőknek az erőssége számít.

Ebben is szerepet játszik azért a távolság, 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) Van, amikor eléri a lokális minimumot, ezt csapda szituációnak nevezik, ebben az esetben a drón nem fog tudni eljutni a célig, mert megáll a lokális minimum pontjánál;
- b) Leng, illetve kileng/kilenghet a drón amikor akadály közelébe kerül;
- c) Habár a drón a fizikai méreteit tekintve átférne 2 akadály között, az erők úgy határozzák, hogy ki kell kerülnie az objektumot, ezáltal nem az optimális elérési utat fogják választani;
- d) Szűk utak között is kilenghet, főleg akkor, ha változás áll be a falak távolsága között.

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

6.6. A Virtuális erő mező metódus (The Virtual Force Field (VFF) Method)

Ezt 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 egy koordináta rendszert használ, aminek a Hisztogram rács nevet

adja, amiben 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 értékeik megnőnek. Ezzel együtt megjelenik a Potenciális mező metódus is. Ahogy a robot mozog egy $w_s * w_s$ lefedő mező követi, ami lefed egy C régiót, ez a régió lesz az aktív régió és azok a cellák, amik beleesnek azok 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.

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

ahol is:

- a) F_{ij} – virtuális taszító erő;
- b) F_{cr} – Taszító erő állandó;
- c) $d(i,j)$ – Távolsága az i,j aktív cellának a robothoz képest;
- d) $C_{i,j}$ – Bizonyossági értéke az aktív cellának;
- e) W – A Robot szélessége;
- f) x_0, y_0 – a robot adott koordinátái;
- g) x_i, y_i – az aktív cellák koordinátái.

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

$$F_r = \sum_{i,j} F_{ij} \quad (6)$$

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

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

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

$$R = F_r + F_t \quad (8)$$

Majd nem sokkal ezek után megcsinálták a Potential Field Method módosított változatát, ami egy adott szinten kijavítja ennek a hibáit [25]. Kutatásukban kifejtik az előbb felsorolt 4 lehetséges hibát részleteiben. A Lokális minimumhoz kapcsolódva leírják, hogy a robot egy zsákutcába jut, ilyen például, ha bekerül egy U alakú akadályba. Amikor 2 akadály között nem fér át a robot, mert az eredő taszító erő ellentétes irányba mutat, a cél pedig mögötte van, amiatt ahelyett, hogy átmenne a 2 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 2 részletben zajlik, míg a VFF-ben csak 1 volt. Ezáltal létrejön az, hogy az 3 adat reprezentációs szint fog létezni, amit kutatásukban részletesen kifejtenek:

- a) A legmagasabb szinten van az a környezet, amiben a robot tevékenykedik. Itt a koordináta rendszer folyamatosan frissíti önmagát a fedélzeti szenzoroknak a segítségével;
- b) Középső szinten, egy egydimenziós Polár hisztogram H képződik a robot pillanatnyi helyzete köré. Ez a H magába foglal n szögletes szektort, aminek α a szélessége. Majd a C^* régiót egy transzformációval áttanszformáljuk ebbe a H -ban;
- c) A legalsó szinten pedig a VFH algoritmus kimenete van. A referencia értékek a vezetésre (gyorsítás, lassítás) és a kormányzásra.

Azonban, mint minden módszernek, 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 beragad egy zsákutcába és ott fog körözni.

Ezt viszont ki lehet azzal küszöbölni, hogy egy adat Heurisztikus szabályt definiálunk neki, de abban az esetben sem lesz optimális az út. Azonban erre kitaláltak egy megoldást, amiben van egy út monitorozás, ami jelző, hogyha a robot zsákutcába kerül. Ezután beállít magának valamilyen elterelő módot és addig nem fog törölni 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 algoritmus felfüggesztésre kerül. Egy Globális Úttervező (Global Path Planner) algoritmus átveszi az

irányítást, hogy csináljon egy új utat a hisztogram rácson 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 Learnig (RL), azaz megerősítéses tanulás. Ennek a lényege, hogy adott szituációkhoz rendelünk adott akciókat és a végén ezeket az akciókat hatásosságuk alapján számszerűen értékeljük.

Az RL-ben egy ügynök szerepel, akinek egy jól előre meghatározott célja van. Ennek a fajta tanulásnak 3 fő komponense van, a környezeti modell, hogy mi, hogyan helyezkedik el, mi hol van, az eljárás módok, hogy menjen egyenesen vagy forduljon jobbra, illetve a jutalmak, amik attól függően, hogy milyen eljárást választott és hogy az közelebb vitte-e a céljához lehet negatív vagy pozitív.

Ezeket a jutalmakat minden akció/eljárás után kap az ügynökünk. Ez a bizonyos ügynök egy olyan entitás, amit megkérünk, hogy csináljon valamilyen akciót az adott állapotától és a múltbeli események alapján. A célja az RL-nek, 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) Ezután a végeredménye ennek az akciónak az, hogy megváltozik az állapota a drónnak, ami annyit tesz, hogy a drón közelebb vagy távolabb kerül a céltól;
- c) Ezután kapja meg a jutalmat a célhoz való közelsége függvényében.

RL-nek három fajtája van, amelyek az alábbiak:

- a) Az érték alapú megközelítés, ahol is a célja az ügynöknek, hogy megtalálja azt az eljárás sorozatot, ami maximalizálja a pontjait;
- b) Az eljárás alapú megközelítés, ahol is az ügynöknek meg kell találnia az optimális értéket;
- c) A modell alapú megközelítés, ami 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 kettő például egy mentőakciónál lehet sikeres, mert ott számokra ismeretlen helyen, környezetben kell manőverezniük, addig a harmadik például egy városban lehet optimálisan használni, ahol is a város térképét bele lehet táplálni a programba, mert az nagyon ritkán fog változni.

Az RL alap koncepciója közé tartozik, hogy az irányítás egy zárt körfolyamatként működik és a jutalmak jelentik a visszajelzést neki. Ha ezt drónokra akarjuk alkalmazni akkor fel lehet úgy is fogni, mint egy módosított Markov döntési folyamat.

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:

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

ahol:

- a) π az eljárás értéke;
- b) s az adott állapot;
- c) a az akció amit tett a modell;
- d) 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 időköznel egy új jutalom értéket 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 alakja az alábbi egyenlettel írható le:

$$Q^{\pi^*}(s, a) = R(s, a) + \tau \sum_{s' \in S} P_{ss'} V^{\pi^*}(s') = \mathbb{E}[r + \gamma \max_{a'} (Q^{\pi^*}(s', a')) | s, a] \quad (10)$$

A kumulált csökkentési faktor:

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

Ennek 3 fejlesztett változata van, amit használnak:

- a) A Dupla DQN, ahol is minden időben más értéket használ, hogy elkezdje az akció.

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

- b) A küzdő DQN (Dueling DQN), ahol is a hálózatot érték szerint és előny szerinti hálózatokra bontjuk. Ahol is az érték hálózat megpróbálja kiértékelni a mostani és az összes állapotát a rendszernek. Miközben az előny hálózat pedig az elérhető akciók minőségét mutatja be, amit a következő egyenlet mutat be:

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

- c) A Dupla Küzdő DQN (Double Dueling DQN), ami az előző kettő összeolvadásáról szól.

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

Mivel a legtöbb esetben az UAV-kat olyan helyekre viszik, ahol a környezetük nem ismert, vagy csak részben ismert amiatt egy olyan algoritmus kell, ami útközben tud tanulni és ezt automatikusan tudja csinálni.

Ebben segít a mesterséges intelligencia és a Deep Learning, mivel ezeknek a segítségével az UAV végi tud menni ismeretlen területeken is, anélkül, hogy bármilyen ütközés veszély fent állna. Ez a Deep Learning is az emberi viselkedésen alapul. Mivel az embernek sincs meg minden ismeret egy új környezetben, de az emlékeinek segítségével tud döntést hozni, hogy mit csinált hasonló helyzetben. Ezt kell a drónnál is elérni, hogy legyen valamilyen “emléke”, hogy tudja, hogy adott helyzeteket hogyan kell leereagálni. [28]

Egy ilyen modellt 7 fő változóval lehet leírni:

- a) S, ami a környezeti állapotot írja le;
- b) A, a lehetséges akciókat;
- c) P, a valószínűsége egy akciónak a jelen és a múlt helyzetei, illetve döntései alapján;
- d) R, a jutalom, ami visszacsatolásra kerül az UAV-hoz, hogy milyen jól teljesített;
- e) Ω , a tényleges megfigyelések;
- f) O, a feltételes valószínűségi eloszlás;
- g) δ , a csökkentési érték.

A DQN hatásosságát igazolására több, különböző kutatás is készült a MATLAB segítségével, amelyek bemutatták, hogy ez az algoritmus működőképes olyan környezetben, ahol nem minden ismert és jól tud kiigazodni előre nem látható helyzetben is. Kutatásaikban Xiaojian Hou és csapata létrehozott egy algoritmust, amiben összekapcsolta a Q-Learning algoritmust, illetve a Potenciális mező metódust (Potential field method) [29], Illetve Ender Cetin és csapata szimulálták a drónok navigációját ismeretlen környezetben megerősítéses tanulással párosítva. [30]

Utóbbiak a kutatásban megállapították, hogy a drón a mesterséges intelligencia/mélytanulás segítségével ki tudja kerülni az álló és mozgó akadályokat és keresztül tud menni az adott környezetben, mint például egy kisebb szomszédságban. Míg a másik kutatócsapat is megállapította, hogy a legtöbb algoritmus nem elég effektíven alkalmazható olyan környezetben, ami dinamikusan változik. 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 algoritmust. A jövőre nézve egyre pontosabb és jobb algoritmusokat fogunk tudni majd létrehozni a kettőnek a kombinálásából, mint láttuk ebben a 2 kutatásban [29] [30]. Megállapításaim szerint egy mesterséges intelligencia alapú (ebben az esetben megerősítéses tanulós) algoritmus kombinálva a VHF, illetve a VHF+ sajátosságával egy jó módszer lehet, hogy még effektívebben tudjunk eljutni A-ból B-be, miközben figyel a drónunk a dinamikus környezet változásra, hirtelen felbukkanó akadályokra és megtudja találni 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.
- [2] A. N. Albert és et al, „Intricacies of medical drones in healthcare delivery: Implications for Africa,” *Technology in Society*, %1. szám66, p. 101624, 2021.
- [3] „The Verge,” [Online]. Available: <https://www.theverge.com/sponsored/goldman-sachs-drones>.
- [4] „Statista,” [Online]. Available: <https://www.statista.com/statistics/1234521/worldwide-drone-market/#:~:text=Drone%20economy%20worldwide%202021%2D2026&text=According%20to%20ODRONEII%2C%20the%20global,safety%2C%20logistics%2C%20and%20passenger>.
- [5] H. Xiaojian és et al, „A UAV Dynamic Path Planning Algorithm,” in 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2020, pp. 127-131.
- [6] H. Xiaojian és et al, „A UAV Dynamic Path Planning Algorithm,” in 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2020, pp. 127-131.
- [7] J. F. Shortle és et al, „Simulating Collision Probabilities of Landing Airplanes at Nontowered Airports,” *Simulation*, %1. kötet80, pp. 21-31, 2004.
- [8] B. M. Sathyaraj és et al, „Multiple UAVs path planning algorithms: A comparative study,” *Fuzzy Optimization and Decision Making*, %1. kötet7, pp. 257--267, 2008.
- [9] S. Aggarwal és N. Kumar, „Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Computer Communications*, %1. kötet149, pp. 270-299, 2020.
- [10] R. Szabolcsi, „3D FLIGHT PATH PLANNING FOR MULTIROTOR UAV,” *Review of the Air Force Academy*, 1. kötet18, pp. 5-16, 2020.
- [11] R. Szabolcsi, „Multirotoros pilóta nélküli légijárművek háromdimenziós repülési pályáinak számítógépes tervezése és szimulációja,” *Hadtudomány*, 1. kötet30, pp. 133-150, 2020.
- [12] R. Szabolcsi, „Flight Path Planning for Small UAV Low Altitude Flights,” *Repüléstudományi Közlemények*, 1. kötet25, pp. 159-167, 06 2020.
- [13] R. Szabolcsi, „Pilóta nélküli légi jármű kis magasságú repülési pályáinak tervezése,” *Repüléstudományi Közlemények*, 1. kötet32, 2020.
- [14] C. Goerzen, Z. Kong és B. Mettler, „A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance,” *Journal of Intelligent and Robotic Systems*, %1. kötet57, pp. 65-100, 2020.
- [15] H. Liu és 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*, %1. kötet23, pp. 3061-3073, 2022.
- [16] Bortoff és Scott, „Path planning for UAVs,” *American Control Conference*, 2000. Proceedings of the 2000, %1. kötet1, pp. 364 - 368, 2000.
- [17] E. Balestrieri, P. Daponte, L. De Vito és . F. Lamonaca, „Sensors and Measurements for Unmanned Systems: An Overview,” *Sensors*, %1. kötet4, %1. szám21, pp. 2-4, 2021.
- [18] J. Giesbrecht, „Global Path Planning for Unmanned Ground Vehicles,” *Defence Research and Development Canada*, Kanada, 2004.
- [19] Balampanis és et al, 2017 International Conference on Unmanned Aircraft Systems (ICUAS), 2017.
- [20] 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.

- [21] [Online]. Available: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/basicmotion.html>.
- [22] Blasi és et al, „Path Planning and Real-Time Collision Avoidance Based on the Essential Visibility Graph,” *Applied Sciences*, %1. kötet10, p. 5613, 2020.
- [23] C. Xia és 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.
- [24] I. W. Geoffrey és C. Sammut, *Encyclopedia of Machine Learning*, Boston, MA: Springer US, 2010.
- [25] V. Jeaneau, A. Kotenkoff és L. Jouanneau, „Path planner methods for UAVs in real environment,” *FAC-PapersOnLine*, %1. kötet51, pp. 292-297, 2018.
- [26] F. Daniel és et al, „A Systematic Literature Review of A* Pathfinding,” *Procedia Computer Science*, %1. kötet179, pp. 507-514, 20221.
- [27] J. Borenstein és Y. Koren, „Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation,” *Proceedings - IEEE International Conference on Robotics and Automation*, %1. kötet2, pp. 1398 - 1404, 1991.
- [28] [Online]. Available: <http://www-personal.umich.edu/~johannb/vff&vfh.htm>.
- [29] J. Borenstein és Y. Koren, „The vector field histogram-fast obstacle avoidance for mobile robots,” *IEEE Transactions on Robotics and Automation*, %1. kötet3, %1. szám7, pp. 278-288, 1991.
- [30] T. Ahmad és et al, „Drone Deep Reinforcement Learning: A Review,” *Electronics*, %1. kötet10, 2021.
- [31] H. Xiaojian és et al, „A UAV Dynamic Path Planning Algorithm,” in *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2020, pp. 127-131.
- [32] Cetin és 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.
- [33] Mihályi, G.: „Az UAV pályatervezés kihívásai és lehetséges megoldásai,” *Repüléstudományi Közlemények, Megjelenés alatt*.