

Debreceni Egyetem
Műszaki Kar
Villamosmérnöki és Mechatronikai Tanszék



Mobil robotok autonóm navigációja

Szerző(k) **Elek Donát Levente BSc hallgató**

Témavezető(k): dr. Almusawi Husam Abdulkareem
Tanszékvezető: prof. dr. Husi Géza

Debrecen
2024.11.11.

Tartalomjegyzék

1. Szakirodalmi áttekintés	5
1.1. Globális navigáció	5
1.2. Lokális navigáció.....	5
2. Navigációs algoritmus.....	7
2.1. Általános struktúra.....	7
2.2. Definiált általános érvényű változók és műveletek	10
2.3. Akadálycsoportosítás.....	11
2.4. Csapdadetektálás	12
2.5. Csapdaelkerülés.....	14
2.6. Statikus komponens	16
2.7. Pályabecslő algoritmus	17
2.8. Dinamikus komponens	21
2.9. Utazási irány és sebesség	23
2.10. Ütközésselkerülés megállással.....	25
3. Eredmények és azok értékelése.....	26
3.1. Robot modell	26
3.2. Teljesítménymutatók	30
3.3. Eredmények statikus környezetben	32
3.4. Eredmények dinamikus környezetben	36
3.5. Eredmények komplex környezetben	39
4. Következtetések és javaslatok	47
Megjegyzés.....	48
Irodalomjegyzék.....	48

Ábrajegyzék

1. ábra: A tervezett navigációs algoritmus blokkdiagramja	8
2. ábra: Memóriefunkció szükségessége az univerzális csapdaelkerülés megvalósításához [31]	9
3. ábra: Akadálycsoportok (piros) és akadályszünetek (zöld) [31].....	12
4. ábra: Az akadálycsoport ívessége [32]	13
5. ábra: Csapdahelyzet detektálás	14
6. ábra: A statikus taszítóerő-távolság függvény.....	17
7. ábra: Mozgó akadály becsült pályája által keltett taszító hatás vektormezője	23
8. ábra: A türelem paraméter értékének hatása a robot pályájára [31]	26
9. ábra: Egy hajtott kerék modellje [32].....	28
10. ábra: A modellezett rendszert alkotó merev testek egymáshoz viszonyított helyzete és kölcsönhatása [32]	29
11. ábra: Statikus tesztpálya U-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)	32
12. ábra: Statikus tesztpálya W-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)	33
13. ábra: Statikus tesztpálya L-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)	34
14. ábra: Statikus tesztpálya T-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)	34
15. ábra: Statikus tesztpálya W-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)	35
16. ábra: Első dinamikus teszt eredményei.....	37
17. ábra: A második dinamikus teszt eredményei	38
18. ábra: A robot sebesség-idő függvénye az első (balra) és a második (jobbra) dinamikus teszt során.....	38
19. ábra: A tervezett navigációs algoritmus az első komplex tesztkörnyezetben	41
20. ábra: Sebesség-idő (balra) és orientáció-idő (jobbra) karakterisztikák az első komplex tesztpályán.....	41
21. ábra: A tervezett navigációs algoritmus a második komplex tesztkörnyezetben.....	43
22. ábra: Sebesség-idő (balra) és orientáció-idő (jobbra) karakterisztikák a második komplex tesztpályán.....	43
23. ábra: A tervezett navigációs algoritmus a harmadik komplex tesztkörnyezetben	45
24. ábra: Sebesség-idő (balra) és orientáció-idő (jobbra) karakterisztikák a harmadik komplex tesztpályán.....	45
25. ábra: Az összehasonlításul használt kétszintű navigációs algoritmusok által bejárt pálya a komplex tesztkörnyezetekben	47

Bevezetés

Az autonóm mobil robotok piaca a következő 5-10 éves előrejelzések alapján exponenciális növekedésben van. A technológia fejlődésével egyre több felhasználási területen (raktárak, gyártósorok, logisztika, drónok, fuvarozás, éttermek, irodák, stb.) jelennek meg mobil robotok. Ennek a fejlődésnek egyik alapvető feltétele az autonóm navigációs módszerek képességeinek és megbízhatóságának növekedése. Az emberi beavatkozás nélküli munkavégző képesség teszi értékessé a technológiát számos felhasználási területen (extrém példák: űrkutatás, katasztrófa mentési folyamatok). Az igazi áttöréshez azonban szükség van olyan megbízhatóan működő robosztus navigációs módszerekre, melyek képesek a munkaterületükről szolgáltatott előzetes információk nélkül is hatékonyan működni. Ezen a területen a navigációs algoritmusok két fő kihívása a csapdahelyzetek kezelése és a mozgó akadályokkal történő ütközések elkerülése.

A projektem célja egy olyan navigációs algoritmus fejlesztése mobil robotok számára, ami képes előre nem ismert környezetben – ahol statikus és mozgó akadályok is találhatóak – önállóan navigálni és a kijelölt célt elérni ütközések nélkül. A cél egy olyan rendszer készítése, mely a lehelyezése után a saját szenzorainak méréseire támaszkodva további segítség nélkül képes elérni a megadott célkoordinátákat. A célkoordináták a kiindulási helyzethez képesti kívánt elmozdulást definiálják és megadhatóak manuálisan vagy egy magasabb szintű programból (ez már kívül esik a projekt céljain, a készített navigációs algoritmus bemenete a célpont koordinátái).

A munka során a célzott felhasználási terület alapján az alábbi feltételezésekkel és egyszerűsítésekkel éltem:

- A munkaterület teljes egészében (közel) sík felületű (pl. beltér, aszfaltozott vagy betonozott kültéri környezet). A talajegyenetlenségek nem kerülnek figyelembe vételre a döntéshozás során az algoritmusban, így az nem alkalmas terepen történő alkalmazásra jelen állapotában.
- A munkaterület nem tartalmaz labirintusszerű részeket, amik magasabb szintű feltérképezést/útkeresést igényelnének. A célzott működési környezet komplexitása egy ésszerű irodai, raktári, gyárcsarnoki, éttermi, stb. elrendezéssel egyezik meg.

A feladat tehát egy $\mathcal{F}_v$ világkoordináta rendszerben ismert $\mathbf{P}_{cél}$ célkoordináták és a robot $\mathbf{P}_r$ pozíciója alapján olyan $\delta_{vál}$ utazási irány és $u_{vál}$ utazási sebesség parancsok kiszámítása a robot távolságmérő szenzorai által szolgáltatott $\mathbf{R}_{lid}$ távolságpontok és ϵ_{din} irányokban detektált mozgó akadályok alapján, melyek garantáltan a $\mathbf{P}_{cél}$ pozícióba konvergálják a robotot, miközben mindenfajta ütközést elkerülnek az akadályokkal. További cél, hogy a döntéshozó

algorithmus számítási igénye elég alacsony legyen, ahhoz hogy legalább 10Hz-es frekvencián (0,1s/iteráció) lehessen működtetni a valós idejű rendszerben való felhasználhatósága céljából.

1. Szakirodalmi áttekintés

1.1. Globális navigáció

Globális navigáció során a rendszernek teljes ismerete van a munkaterről és ez alapján valamilyen kereső algoritmus segítségével kell egy útvonalat terveznie, ami összeköti a célt a robot pillanatnyi pozíciójával. Ez az útvonal valamilyen paraméter (pl. úthossz, utazási idő, felhasznált energia stb.) alapján optimalizálva lehet és megfelelően hosszú futásidő esetén garantáltan eléri a célt, amennyiben ez a megadott paraméterek (pl. akadályoktól való minimális távolság) betartásával lehetséges.

A fentiek alapján a globális navigációs módszerek lényegében kereső algoritmusok, például a hagyományos A* algoritmus [1].

A napjainkban legnépszerűbb két megközelítés a gyorsan-növekvő fagráf, angolul „Rapidly-Exploring Random Tree” (RRT) [2], illetve a valószínűség alapú úterv, angolul „Probabilistic Roadmap” (PRM) [3]. Ezen két módszerhez rengeteg fejlesztés és kiegészítés lett publikálva az évek során [4].

1.2. Lokális navigáció

Lokális navigáció során a robotnak nincsen előzetes információja a munkaterületről, kizárólag a szenzorai által szolgáltatott pillanatnyi adatok alapján hoz döntéseket a pillanatnyi haladási irányról. Mivel nem tervezi előre a pályát az eredmény nincs optimalizálva, illetve nem garantálható a cél elérése (nincs meg az ehhez szükséges globális információ). Ellenben a döntéshozás nagyon gyorsan történik, így a rendszer képes valós időben reagálni a környezet változásaira (pl. mozgó akadályok).

A lokális navigációs módszerek közül az egyik legkorábbiak a potenciálmező alapú algoritmusok, angol néven „Artificial Potential Field” (APF) [5]. A módszer elvi megközelítése, hogy a robot környezetét egy potenciálfüggvény segítségével reprezentálja, aminek lokális maximumai akadályokat, globális minimuma pedig a célt jelképezi. A robot ez alapján egy gradiens ereszkedés módszerrel érheti el a célt. A gyakorlati megvalósításnál általában kihagyják a potenciálmező felépítését és a deriválás additív tulajdonságát kihasználva közvetlenül a szenzorok távolságméréseiből számolnak virtuális tasztító erőkomponenseket. A cél irányába mutató vonzókomponenssel összeadva az eredő erővektor meghatározza az

elméleti potenciálmező pillanatnyi gradiensét, ami egyben a robot pillanatnyi haladási irányát adja meg. Ennek a megközelítésnek az előnye, hogy a detektált akadályok által kifejtett taszítóerő komponensek meglehetősen robusztussá teszik az ütközések elkerülése szempontjából. Továbbá a szenzorok méréseiből egyetlen függvénnyel és vektorösszeggel megadja a szükséges haladási irányt. Ez rendkívül gyors működést tesz lehetővé, ugyanakkor az egy lépcsőben történő adatredukálás információvesztéshez és nem-optimális döntéshozáshoz vezet. Az ebből adódó legnagyobb hiányosság az úgynevezett „lokális minimum” probléma. Ez lényegében a csapdába esésnek megfeleltethető helyzet, amikor a gradiens ereszkedés során a robot a célt jelképező globális minimum helyett egy hozzá közelebb eső lokális minimumba navigál és ott ragad. Gyakorlati megvalósításnál ez olyan formában jelenik meg, hogy a taszító és vonzó erők kioltják egymást és eredőjük zérus. Ilyenkor az algoritmus nem tud dönteni a követendő mozgásirányról és egy helyben ragad. A csapdába esés jelensége egyébként minden lokális navigációs algoritmus esetén fennáll, viszont az APF módszer különösen érzékeny rá. Emiatt, valamint mert a jelenség egzakt matematikai megfogalmazását először ennél a módszernél definiálták és vizsgálták a csapdába esés problémáját a lokális navigáció területén gyakran hívják lokális minimum problémának más algoritmusok esetén is. Az APF módszer további hiányosságai az erősen oszcilláló mozgás falakkal párhuzamos mozgás esetén, a szűk átjárókon (pl. ajtó) való áthaladás nehézsége, illetve a cél elérhetetlensége akadály közvetlen közelében [6]. Ezen jól ismert problémák ellenére az APF módszer a mai napig gyakran használt különböző formákban, valamint komplexebb rendszerek komponenseként [7], [8], [9], [10]. Ennek oka, hogy egyszerűen implementálható, rendkívül alacsony a számítási igénye, könnyen vizualizálható és megérthető és a taszító erőfüggvények változtatásával könnyen lehet új viselkedésformákat implementálni, mint például a fal-követő viselkedés [11].

Az APF módszer hiányosságainak elemzése vezetett a vektormező hisztogram, angolul „Vector Field Histogram” (VFH) algoritmus kidolgozásához [12]. A módszer fő jellemzője az APF-hez képest a két lépcsőben történő adatredukció, ami lehetővé teszi a csapdahelyzetekre való érzékenység csökkentését. A VFH a szenzorokból származó mérések alapján egy polárkoordináta rendszerbeli hisztogramot hoz létre, ami a környezet pillanatnyi állapotát reprezentálja. Az akadályok irányába eső szögeknél a hisztogram értéke magas, a szabad haladási irányokba pedig alacsony, úgynevezett völgyeket hozva létre. Az adatredukció második lépésében ezekből a detektált völgyekből (azaz az elérhető szabad haladási irányokból) választ ki az algoritmus egyet a cél iránya, a pillanatnyi sebesség és az előző haladási irány figyelembe vételével. Ezzel a megközelítéssel a VFH módszer elkerüli az oszcilláló viselkedést falak mentén és képes áthaladni szűkebb átjárókon a völgyek detektálásának köszönhetően. A

VFH módszer legelterjedtebb továbbfejlesztett változatai a VFH+ [13], [14] és a VFH* [15] navigációs algoritmusok. A VFH+ egy egyszerűsített kinematikai modellel figyelembe veszi a kiadott parancsok megvalósíthatóságát, illetve az akadályok megnagyobbításával a robot méretét. A VFH* pedig egy A* alapú rövidtávú előrejelzőt alkalmazva a kiadott utasítások következményeit veszi figyelembe és ez alapján igyekszik optimalizálni a döntést.

A dinamikus ablak alapú megközelítés, angolul „Dynamic Window Approach” (DWA) [16], [17], [18], [19] a robot kinematikai limitációit figyelembe véve választ egy optimális utazási sebességet. Ennek köszönhetően a kapott utazási parancsok minden esetben fizikailag megvalósíthatóak és kihasználják az elérhető maximális sebességet. Azonban a DWA módszer szintén kiemelten érzékeny a csapdába esés jelenségére.

A közelségi diagram, angolul „Nearness Diagram” (ND) módszer [20], [21] egy „oszd meg és uralkodj” elven alapulva elsősorban a szűk helyeken való navigációt célozza meg. Hátránya viszont, hogy nincs felkészítve dinamikus akadályokra és hajlamos az elérhető nagyobb méretű szabad területek felé terelődni.

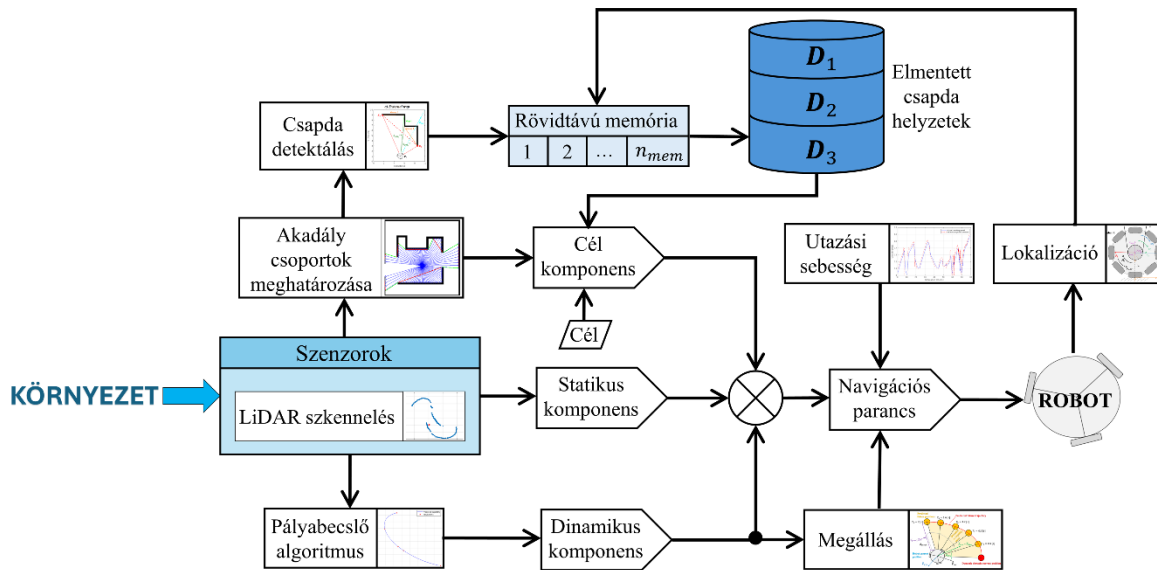
Később, a pillanatnyi szenzoradatok feldolgozásának további vizsgálatával megjelentek a különböző réskövető algoritmusok [22], [23], [24], [25] a közelségi diagramhoz hasonló navigációs módszerrel. Viszont nem lettek adaptáltak a prediktív dinamikus akadályelkerüléshez, továbbá ezekből is hiányzik mindenféle memória, ami az általuk elkerülhető csapdahelyzetek körét a közvetlenül észlelhető csapda forgatókönyvekre korlátozza, látószög-korlátozás nélkül. A memória funkció szükségét az univerzális csapdaelkerülő megoldáshoz további vizsgálatok bizonyították [26].

A globális és lokális módszerek egyéni hiányosságai miatt gyakori megoldás a kétszintű navigációs algoritmusok alkalmazása [27], [28], [29], [30]. Itt egy alacsonyabb frekvencián futó globális tervező felel a csapdák elkerüléséért és egy nagy frekvencián működő lokális algoritmus valósítja meg az ütközések elkerülését. A rendszer előnye hogy kihasználja mindkét megközelítés erősségeit, hátránya viszont, hogy ismeretlen munkaérben való működéséhez szüksége van egy feltérképező algoritmusra. Ez a globális tervezővel együtt jelentős számítási többletet igényel, ami online működés esetén komoly kihívást jelenthet.

2. Navigációs algoritmus

2.1. Általános struktúra

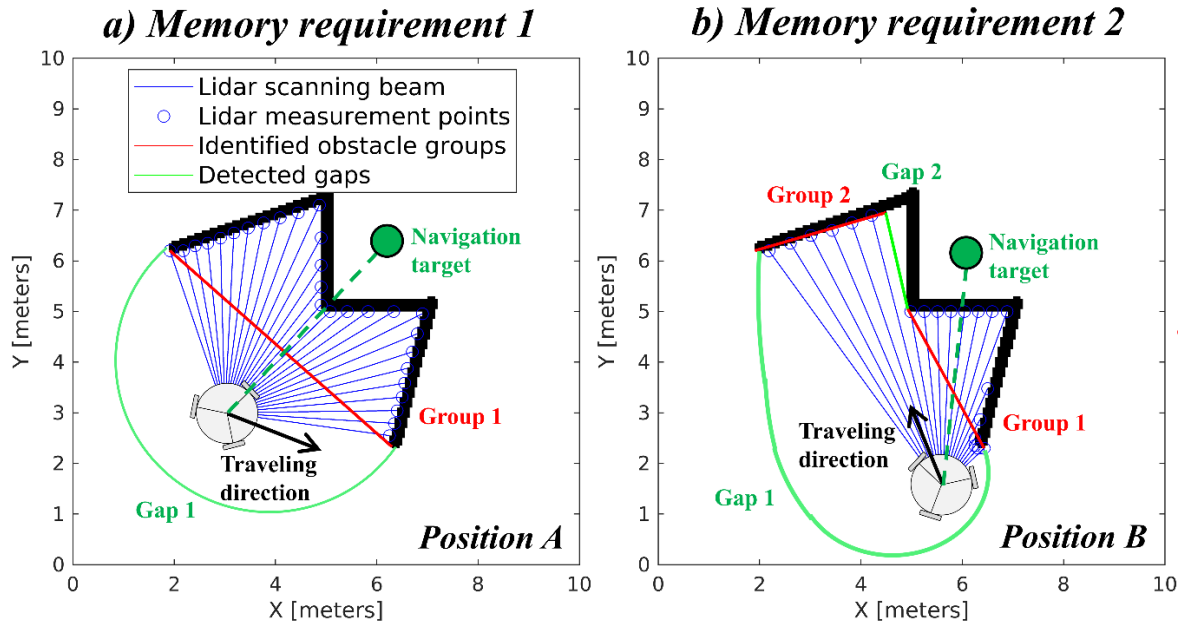
A létrehozott navigációs módszer felépítése az alábbi blokkdiagrammon látható.



1. ábra: A tervezett navigációs algoritmus blokkdiagramja

A rendszer a hagyományos lokális módszereknek megfeleltethető statikus komponens mellett tartalmaz egy csapdák elkerülését szolgáló, múltbeli információkon alapuló célkomponenst, illetve egy mozgó akadályokkal való ütközés elkerülését szolgáló becsült jövőbeli információkon alapuló dinamikus komponenst.

A múltbeli információk feldolgozása a csapdahelyzetek univerzális elkerüléséhez elengedhetetlen. A pillanatnyi szenzorinformációk átgondolt vizsgálatával készíthető olyan algoritmus, amely képes a csapdahelyzetek egy részének feltételes elkerülésére, de a minden esetben működő megoldások elérése lehetetlen valamilyen formájú memóriefunkció nélkül. Ennek belátására tekintsük az 1. ábra által bemutatott példát.



2. ábra: Memória funkció szükségessége az univerzális csapdaelkerülés megvalósításához [31]

Itt az A pozícióban a robot a pillanatnyi szenzormérések alapján detektálja az akadályt a helyzete és a cél között és dönt az elkerülő manőverről. Az akadály széléhez közelítve azonban, a B pozícióban a pillanatnyi szenzordatok a látószög korlátok miatt olyan információt közölnek az algoritmus felé, amely szerint az akadály közepén egy átjáró található, ami közelebb van a célhoz, mint a jelenleg célba vett útvonal. Amennyiben a döntés kizárólag ezen pillanatnyi méréseken alapulva születik meg a zsákutcába vezető útvonal fog előnyösebbnek tűnni és ezt fogja választani a robot. Ennek következtében egy oszcilláló viselkedés jön létre az A és B pontok között. Ennek a helyzetnek a megoldása az, hogy a robot „emlékszik” az A pozíció méréseire a B pontban és ez alapján tudja, hogy az akadály közepén feltűnő útvonal csak egy hamis detektálás és nem kell figyelembe venni. Sok navigációs algoritmus a cél és a jelenlegi haladási irányt súlyozással figyelembe véve választ az elérhető szabad útirányok közül. Ez a megközelítés képes sok hasonló helyzetet megoldani, viszont mindig található 2 olyan elrendezés, amely azonos súlyozási paraméterekkel nem megoldható, más szóval ez a módszer nem vezet általános eredményre.

A pályabecslő algoritmus a mozgó akadályok elkerülésének megkönnyítését szolgálja. A gyors működési sebesség és a pillanatnyi méréseken alapuló döntéshozás miatt minden lokális navigációs módszer képes reagálni a környezet változásaira. Ez lehetővé teszi a dinamikus akadályok reaktív elkerülését. Viszont a robot mozgásának megváltoztatásához szükség van egy bizonyos reakcióidőre, amely gyorsabb akadályok esetén nem teszi lehetővé az ütközés

elkerülését, amennyiben csak utólag reagálunk a mozgásra. A pályabecslő algoritmus lényege, hogy képet kapjuk a közeljövőbeli változásokról és úgy hozzuk meg a navigációhoz szükséges döntéseket, hogy ezt a jövőt figyelembe vesszük. Ezáltal hamarabb kezd a robot bele az elkerülő manőverbe, ami megnöveli az esélyét annak, hogy a reakcióidő rövidebb lesz, mint az ütközésig hátralévő idő. Ezzel a prediktív elkerülési stratégiával növelhető a mozgó akadályok elkerülésének valószínűsége.

2.2. Definiált általános érvényű változók és műveletek

A továbbiakban definiálok néhány változót és műveletet az általam készített navigációs módszer levezetésének megkönnyítése érdekében. Az alábbi táblázat tartalmazza a dokumentum terjedelmében általános érvényű változók listáját.

1. Táblázat: A dokumentumban általános érvényű változók listája

$\mathcal{F}_v$	Világkoordináta-rendszer	$\mathcal{F}_t$	Testkoordináta-rendszer
$\mathbf{P}_r = [x_r, y_r, \varphi_r]^T$	Robot pozícióvektora világkoordinátában (φ_r az orientáció)		
$\mathbf{v}_r = [v_x, v_y, \omega_r]^T$	Robot sebességvektora világkoordinátában (ω_r a szögsebesség)		
$\mathbf{a}_r = [a_x, a_y, \beta_r]^T$	Robot gyorsulásvektora világkoordinátában (β_r a szöggyorsulás)		
$\mathbf{R}_{lid}$	Távolságpontok vektora	$\boldsymbol{\theta}_{lid}$	Polár szögkoordináták vektora
n_{lid}	Távolságpontok száma	n_{gr}	Akadálycsoportok száma
$\mathbf{P}_{cél} = (x_{cél}, y_{cél})$	Célpont koordinátái	n_{din}	Mozgó akadályok száma
$\delta_{vál}$	Választott utazási irány	$u_{vál}$	Választott utazási sebesség

Műveletek közül elsőként a robot testkoordináta-rendszere ($\mathcal{F}_t$) és a világkoordináta-rendszer ($\mathcal{F}_v$) közötti átváltást adom meg. A polárkoordinátákkal leírt testkoordináta-rendszerből a Descartes-féle világ koordináta-rendszerbe váltást a $(x, y) = (r, \theta)^{p \rightarrow D}$ művelet írja le, ahol:

$$\begin{aligned} x &= r \cos(\theta + \varphi_r) + x_r \\ y &= r \sin(\theta + \varphi_r) + y_r \end{aligned} \quad (1)$$

Az inverz művelet $(r, \theta) = (x, y)^{D \rightarrow p}$ definíciója:

$$\begin{aligned} r &= \sqrt{(x - x_r)^2 + (y - y_r)^2} \\ \theta &= \tan^{-1} \left(\frac{y - y_r}{x - x_r} \right) - \varphi_r \end{aligned} \quad (2)$$

A robot környezetének polárkoordináta rendszerbeli leírásához definiálok az óramutató járásával ellentétes irányú szögtartományt két határszög között az $A = [-\pi; \pi]$ alaphalmazon a $\Lambda(\theta_1, \theta_2)$ műveletet az alábbiak szerint:

$$\Lambda(\theta_1, \theta_2) = \begin{cases} [\theta_1; \theta_2], & \text{ha } \theta_1 \leq \theta_2 \\ [\theta_1; \pi] \cup [-\pi; \theta_2], & \text{ha } \theta_1 > \theta_2 \end{cases} \quad (3)$$

Továbbá definiálom a minimum szögtartományt két határszög között a $V(\theta_1, \theta_2)$ művelettel, az alábbiak szerint:

$$V(\theta_1, \theta_2) = \begin{cases} \Lambda(\theta_-, \theta_+), & \text{ha } |\theta_1 - \theta_2| \leq \pi \\ \Lambda(\theta_+, \theta_-), & \text{ha } |\theta_1 - \theta_2| > \pi \end{cases}$$

$$\begin{aligned} \theta_+ &= \theta_1, & \theta_- &= \theta_2, & \text{ha } \theta_1 &\geq \theta_2 \\ \theta_+ &= \theta_2, & \theta_- &= \theta_1, & \text{ha } \theta_1 &< \theta_2 \end{aligned} \quad (4)$$

Végezetül két szög közti különbséget a $\Delta\varphi(\theta_1, \theta_2)$ művelet ír le:

$$\Delta\varphi(\theta_1, \theta_2) = \begin{cases} \theta_1 - \theta_2 + 2\pi, & \text{ha } \theta_1 - \theta_2 < -\pi \\ \theta_1 - \theta_2 - 2\pi, & \text{ha } \theta_1 - \theta_2 > \pi \\ \theta_1 - \theta_2, & \text{egyébként} \end{cases} \quad (5)$$

2.3. Akadálycsoportosítás

Az algoritmus a szenzorokból beérkező távolságmérésekből először akadálycsoportokat hoz létre a pillanatnyi helyzet elemzésének megkönnyítésére. Azokat a mért távolságpontokat, amelyek a szomszédjuktól egy adott távolságnál ($2R_r$) nincsenek messzebb egy csoportba soroljuk.

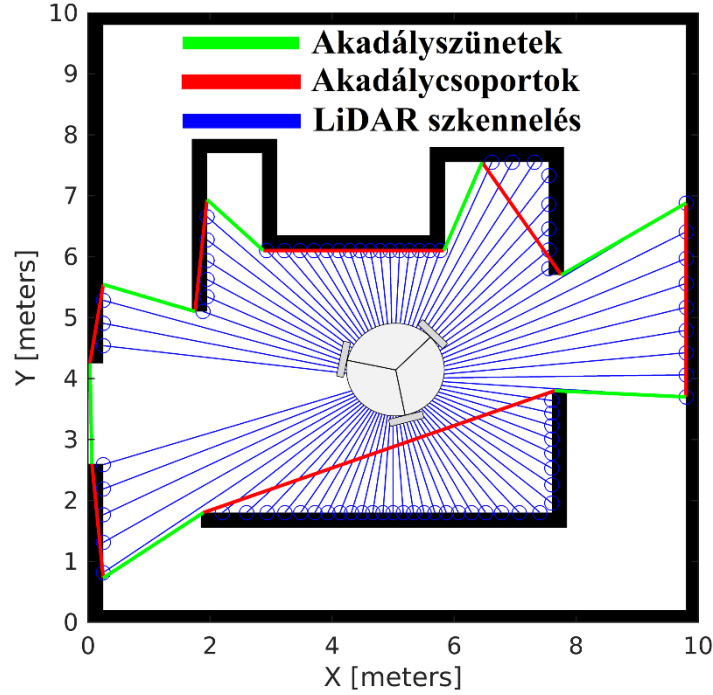
Az egyes csoportok a testkoordináta-rendszerben a $\mathbf{g}_1$ és $\mathbf{g}_2$ határszögeikkel, míg a világkoordináta-rendszerben a $\mathbf{G}_1$ és $\mathbf{G}_2$ végpontjaikkal vannak reprezentálva, melyek között az alábbi összefüggés teremt kapcsolatot:

$$\mathbf{G}_1 = (\mathbf{R}_{lid}(\mathbf{g}_1), \mathbf{g}_1)^{p \rightarrow D} \quad (6)$$

A csoportok közötti elválasztó területek jelképezik a szabad haladási irányokat a környezet pillanatnyi állapotában, melyeket akadályszünetnek fogok nevezni a továbbiakban. Az akadályszünetek reprezentálására egy bináris vektort hozunk létre az alábbiak szerint:

$$\mathbf{H}_{szün}(i) = \begin{cases} 0, & \text{ha } \theta_{lid}(i) \in \Lambda(\mathbf{g}_1(k), \mathbf{g}_2(k)) \mid \forall k \leq n_{gr} \\ 1, & \text{egyébként} \end{cases} \quad (7)$$

Az akadálycsoportok és -szünetek elhelyezkedésére egy adott pillanatban a 3. ábra mutat egy példát.



3. ábra: Akadálycsoportok (piros) és akadályszünetek (zöld) [31]

2.4. Csapdadetektálás

A memóriefunkció lényege, hogy a robot mozgása során folyamatosan elemzi a pillanatnyi szenzor méréseket, detektálja a potenciális csapdahelyzeteket és a felismerésükhöz szükséges információt menti el, minden mást elhanyagol. Így a későbbi iterációkban már csak a lényegi adatok maradnak vissza, jelentősen csökkentve a feldolgozásukhoz szükséges számítási igényt. A potenciális csapdahelyzetek felismerésének alapja a korábban ismertetett akadálycsoportosítás. Amennyiben egy adott pillanatban létezik egy akadály a robot és a cél között, más szóval a feltétel

$$\varphi_{cél} \in \Lambda(\mathbf{g}_1(k_{bl}), \mathbf{g}_2(k_{bl})) \quad (8)$$

$$(d_{cél}, \varphi_{cél}) = (x_{cél}, y_{cél})^{D \rightarrow p} \quad (9)$$

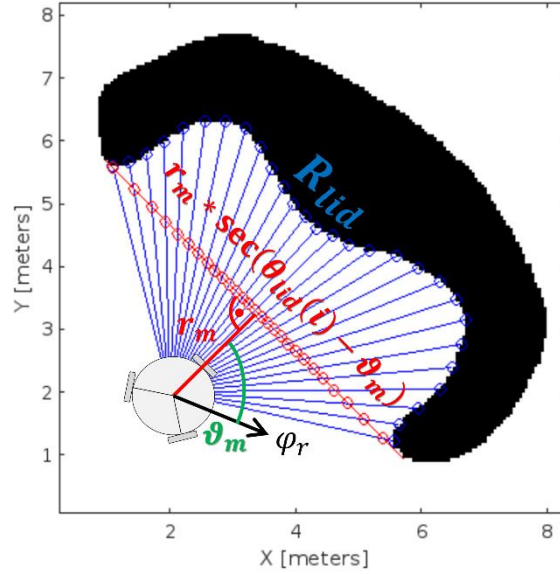
bármely $k_{bl} \leq n_{gr}$ -re teljesül, akkor megvizsgáljuk a csapdába esés szempontjából jelentett kockázatát. Két esetben minősül a cél és robot közötti akadálycsoport potenciális csapdának: elsőként amennyiben több mint a horizont felét elfoglalja, vagyis teljesül az alábbi feltétel:

$$|\Lambda(\mathbf{g}_1(k_{bl}), \mathbf{g}_2(k_{bl}))| > \pi \quad (10)$$

A második eset, ha az akadálycsoport elég íves ahhoz, hogy csapdának lehessen tekinteni. Ezt úgy tudjuk megállapítani, hogyha az akadálycsoport távolságpontjainak több mint 80% teljesíti az alábbi feltételt:

$$r_m * \sec(\theta_{lid}(i) - \vartheta_m) + 0.1 < \mathbf{R}_{lid}(i) \quad (11)$$

Itt (r_m, ϑ_m) a testkoordináta-rendszerbeli polárkoordinátái az akadálycsoport két végpontját összekötő egyenes robot pozíciójához legközelebbi pontjának. Az ívesség vizsgálatot a 4. ábra mutatja be.



4. ábra: Az akadálycsoport ívessége [32]

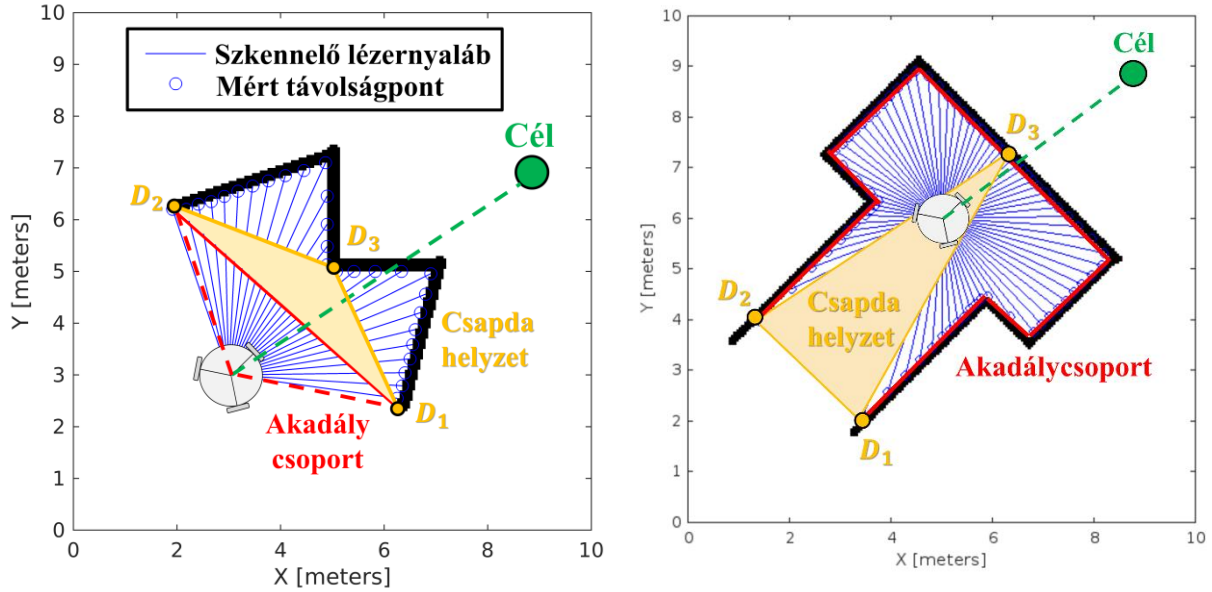
Amennyiben a két feltétel közül valamelyik teljesül az akadálycsoport végpontjait elmentjük egy rövidtávú memóriába, az alábbiak szerint:

$$\mathbf{D}_1(1) = \mathbf{G}_1(k_{bl}), \quad \mathbf{D}_2(1) = \mathbf{G}_2(k_{bl}) \quad (12)$$

$$\mathbf{D}_3(1) = (\mathbf{R}_{lid}(d_3), d_3)^{p \rightarrow D} \quad (13)$$

$$d_3 = \text{med} \left(\Lambda(\mathbf{g}_1(k_{bl}), \mathbf{g}_2(k_{bl})) \right) \quad (14)$$

Itt $\text{med}(\cdot)$ a medián függvény (a szögtartomány középértéke). A potenciális akadályok detektálását a 5. ábra illusztrálja.



5. ábra: Csapdahelyzet detektálás

A rövidtávú memória célja, hogy kiszűrjük a dinamikus akadályok okozta esetleges hamis detektálásokat. Ennek érdekében a minden iteráció során elmentett új elem végpontjait összehasonlítjuk a memóriában lévő korábbi detektálásokkal. Amennyiben az új elem egyezik (hibahatáron belül) legalább 10 korábbi elemmel, akkor megerősített csapdahelyzetnek tekintjük és elmentjük a végleges memóriába. A redundancia elkerülése végett a végleges memóriába mentéskor megnézzük, hogy nincs-e hasonló elem már elmentve. A rövidtávú memória lényegében egy FIFO tömb, adott n_{mem} elemszámmal, amibe minden iteráció során beírjuk az újonnan detektált $D_{1,2,3}$ végpontokat és töröljük a legrégebbi elemeket. Ha az adott iterációban nincs új csapdahelyzet detektálva, akkor egy üres sort mentünk el a rövidtávú memóriába. A végleges memória n_{csap} elemből áll, amik a megerősített csapdahelyzetekről szolgáltatnak információt a döntéshozó modulnak. A végleges memóriában a csapdahelyzetek végpontjait $C_{1,2,3}$ pontokkal jelöljük.

2.5. Csapdaelkerülés

A végleges memóriában szereplő elemek alapján a $H_{szün}$ akadályszünet vektorból ki kell vonnunk a csapdahelyzetek felé mutató irányokat mielőtt meghatározzuk a követendő navigációs irányt. Ehhez először végig nézzük az összes elmentett megerősített csapdahelyzetet és ezek alapján készítünk egy csapdavektort. Amennyiben a robot pillanatnyi pozíciója az m indexű csapdahelyzetben belül található, belső csapdaként kezeljük azt. A pozíció csapdán belül elhelyezkedését háromszög területek segítségével ellenőrizhetjük:

$$T_{C_1C_2C_3} \approx T_{P_rC_2C_3} + T_{P_rC_1C_3} + T_{P_rC_1C_2} \quad (15)$$

Ha a fenti feltétel teljesül, az m indexű csapdahelyzet $\mathbf{A}_{csap}$ csapdavektorra gyakorolt hatása az alábbiak alapján számolható:

$$\mathbf{A}_{csap}(i) = \begin{cases} -2, & \text{ha } \boldsymbol{\theta}_{lid}(i) \in V(c_1, c_3) \cup V(c_3, c_2) \\ 0, & \text{egyébként} \end{cases} \quad (16)$$

$$(r_l, c_l) = (\mathbf{C}_l(m))^{D \rightarrow p} \mid l = 1, 2, 3 \quad (17)$$

Amennyiben a robot a megerősített csapdahelyzeten kívül helyezkedik el, külső csapdaként kezeljük az esetet. Ennek feltétele a következő:

$$T_{C_1C_2C_3} < T_{P_rC_2C_3} + T_{P_rC_1C_3} + T_{P_rC_1C_2} \quad (18)$$

A külső csapdák nem minden esetben okoznak problémát a navigáció során, csak ha közvetlenül a robot és a cél között helyezkednek el. Ezt két további feltétel segítségével állapíthatjuk meg:

$$d_l < \mathbf{R}_{lid}(c_l) + 0.6 \quad \forall l = 1, 2, 3 \quad (19)$$

$$|P_r M_m| + |M_m P_{cél}| = d_{cél} \quad (20)$$

Itt M_m a metszéspontja a robotot a céllal összekötő $P_r P_{cél}$, valamint a megerősített csapdahelyzet két végpontját összekötő $\mathbf{C}_1(m) \mathbf{C}_2(m)$ egyeneseknek, azaz:

$$M_m = P_r P_{cél} \cap \mathbf{C}_1(m) \mathbf{C}_2(m) \quad (21)$$

Ekkor az m indexű csapdahelyzet csapdavektorra gyakorolt hatása az alábbiak szerint alakul:

$$\mathbf{A}_{csap}(i) = \begin{cases} -2, & \text{ha } \boldsymbol{\theta}_{lid}(i) \in V(c_1, c_2) \\ 0, & \text{egyébként} \end{cases} \quad (22)$$

Miután elkészítettük a csapdavektort az akadályszűnet vektorral vette eredőjét ($\mathbf{A}_{er}$) felhasználva definiálunk egy $\mathbf{E}$ költségfüggvényt a robot megengedett haladási irányaira:

$$\mathbf{E}(i) = \begin{cases} 10^5, & \text{ha } \mathbf{A}_{er}(i) < 0 \\ \Delta\varphi(\boldsymbol{\theta}_{lid}(i), \varphi_{cél}) + \frac{\Delta\varphi(\boldsymbol{\theta}_{lid}(i), \delta_{vál}')}{4}, & \text{egyébként} \end{cases} \quad (23)$$

$$\mathbf{A}_{er}(i) = \mathbf{A}_{csap}(i) + \mathbf{H}_{szűn}(i) \quad (24)$$

Itt $\delta_{vál}'$ az előző iterációban kiválasztott haladási irány. Az a szög, amelynél a költségfüggvény értéke a minimum lesz kiválasztva a célkomponens irányának:

$$\delta_{cél} = \boldsymbol{\theta}_{lid}(i_{min}), \text{ ahol } \mathbf{E}(i_{min}) = \min(\mathbf{E}) \quad (25)$$

Az akadálycsoportok létrehozását, a csapdadetektálást és a célkomponens meghatározását az I. Algoritmus mutatja be.

I. Algoritmus: A célkomponens meghatározása
Input: $\mathbf{R}_{lid}$; $\boldsymbol{\theta}_{lid}$; $\mathbf{P}_{cél}$; $\mathbf{P}_r$
Output: $\delta_{cél}$
$(d_{cél}, \varphi_{cél}) = D \rightarrow p(\mathbf{P}_{tar})$
for $i = 1$ to n_{lid} do //akadálycsoport készítés

```

     $P_i = p \rightarrow D(R_{lid}(\theta_{lid}), \theta_{lid})$ 
    if  $|P_i P_{i+1}| < 2r_r$  then  $G_2(n_{gr}) \leftarrow P_{i+1}$ 
    else  $n_{gr} ++$ ;  $G_1(n_{gr}) \leftarrow P_{i+1}$ 
for  $i = 1$  to  $n_{lid}$  do //akadályszűnet vektor készítés
    if  $\theta_{lid}(i) \in \Lambda(g_1(i), g_2(i))$  then  $H_{szün}(i) = 1$ 
    else  $H_{szün}(i) = 0$ 
find  $k_{bl}$  by  $\varphi_{tar} \in \Lambda(g_1(k_{bl}), g_2(k_{bl}))$ 
if  $|\Lambda(g_1(k_{bl}), g_2(k_{bl}))| > \pi$  OR ívesesség is TRUE //csapda detektálás
     $D_{1,2,3}(1) \leftarrow G_{1,2}(k_{bl})$ ; shift  $D_{1,2,3}$  by 1; delete  $D_{1,2,3}(n_{mem})$ 
    for  $i = 1$  to  $n_{mem}$  do
        if  $D_{1,2,3}(1) \approx D_{1,2,3}(i)$  then  $t_{mem} ++$ 
    if  $t_{mem} > 1s$  then  $n_{csap} ++$ ;  $C_{1,2,3}(n_{csap}) \leftarrow D_{1,2,3}(1)$ 
for  $i = 1$  to  $n_{lid}$  do //korábbi csapdahelyzetek feldolgozása
    for  $m = 1$  to  $n_{csap}$  do
        if  $P_r \subset C_1(m)C_2(m)C_3(m)\Delta$ 
            if  $\theta_{lid}(i) \in V(c_1, c_3) \cup V(c_3, c_2)$  then  $A_{csap}(i) = -2$ 
            else  $A_{csap}(i) = 0$ 
        else
            if  $|P_r M_m| + |M_m P_{cél}| = d_{cél}$  OR  $d_{1,2,3} < R_{lid}(c_{1,2,3}) + 0.6$ 
                if  $\theta_{lid}(i) \in V(c_1, c_2)$  then  $A_{csap}(i) = -2$ 
                else  $A_{csap}(i) = 0$ 
     $E \leftarrow H_{szün}, A_{csap}$ 
     $\delta_{cél} \leftarrow \min(E)$ 

```

2.6. Statikus komponens

A statikus komponens lényegében egy hagyományos lokális navigációs algoritmusnak felel meg, ami a környezet jelen helyzetének feldolgozásáért felel. A fő célja a detektált akadályokkal való ütközések elkerülésének biztosítása. Ebből a célból minden mért távolságpont egy virtuális taszítóerőt fejt ki a robotra, amelyek eredője lesz a statikus komponens (F_{stat}) az alábbiak szerint:

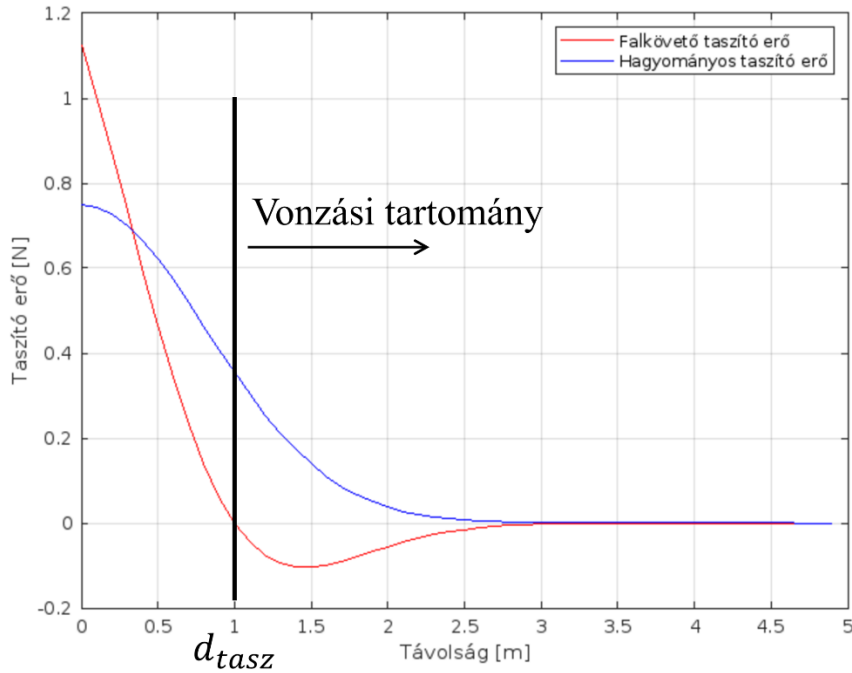
$$F_{stat} = \sum_{i=1}^{n_{lid}} -f_{st}(i) * \begin{pmatrix} \cos(\theta_{lid}(i) + \varphi_r) \\ \sin(\theta_{lid}(i) + \varphi_r) \end{pmatrix} \quad (26)$$

Az egyes távolságpontok által kifejtett virtuális erő nagyságát ($f_{st}(i)$) egy olyan függvénnyel írjuk le, ami lehetővé teszi a falkövető viselkedést és csökkenti az oszcillációt a mozgásban:

$$f_{st}(i) = \begin{cases} 0, & \text{ha } R_{lid}(i) > d_{max} \\ \frac{g_{st}}{n_{lid}} * h_{st} e^{-h_{st} * R_{lid}^2(i)} * (d_{tasz} - R_{lid}(i)), & \text{egyébként} \end{cases} \quad (27)$$

Ezen taszítóerő függvény fő tulajdonsága, hogy adott d_{tasz} távolságtól messzebb lévő távolságpontok által kifejtett erő vonzó irányú lesz a taszító helyett. Ez a megközelítés lehetővé

teszi, hogy a d_{tasz} paraméterrel beállítható távolságban a faltól egy lokális minimum „csatornát” hozunk létre, ami egy stabil mozgást biztosít a robot számára. A statikus taszítóerő-távolság függvényt a 6. ábra mutatja.



6. ábra: A statikus taszítóerő-távolság függvény

A szimulációban a függvényhez használt konstansok értékei: $d_{tasz} = 0.75m$; $d_{max} = 2m$; $h_{st} = 0.75$ és $g_{st} = 500$.

2.7. Pályabecslő algoritmus

Az itt ismertetett pályabecslő algoritmus szintén be lett mutatva a [33] konferencián. A pályabecslő algoritmusok lényege, hogy a mozgó akadályok múltban megfigyelt pozíciója alapján megjósolja a közeljövőben leírt pályájukat. A bemenetként szolgáló múltbeli pozíciók a $\mathbf{D}_{in}$ tömbben kerülnek tárolásra, amelynek összesen n_{in} eleme van. Amennyiben az adott iterációban ε_{din} szögnek megfelelő irányban detektálunk egy mozgó akadályt, a robothoz képest számított helyvektora az alábbi módon határozható meg:

$$\mathbf{r}_{din} = \begin{pmatrix} x_{din} \\ y_{din} \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \end{pmatrix} + \mathbf{R}_{lid}(\varepsilon_{din}) * \begin{pmatrix} \cos(\varepsilon_{din} + \varphi_r) \\ \sin(\varepsilon_{din} + \varphi_r) \end{pmatrix} \quad (28)$$

Ezt a műveletet egymás utáni iterációkban megismételve megkapjuk a dinamikus akadály közelmúltbeli pozícióinak vektorát az alábbiak szerint:

$$\mathbf{D}_{in} = \begin{bmatrix} \mathbf{r}_{din}[n_{in}] \\ \mathbf{r}_{din}[n_{in} - 1] \\ \vdots \\ \mathbf{r}_{din}[1] \end{bmatrix} \quad (29)$$

A $\mathbf{D}_{in}$ tömb minden eleméhez tartozik egy T_k detektálási idő (ahol k az elem indexe). A jövőbeli pálya megbecsléséhez valamilyen görbét kell illeszteni a bemenetként megadott $\mathbf{D}_{in}$ pontokra. Ehhez először redukáljuk az adatsort 5 vezérpontra ($P_l \mid l = 1,2,3,4,5$) az alábbiak szerint:

$$P_i = \frac{5}{n_{in}} * \sum_{k=1+(i-1)0.2n_{in}}^{i*0.2n_{in}} \mathbf{D}_{in}(k) \quad (30)$$

Az egyes vezérpontok detektálási ideje a következőképpen adható meg:

$$T_i = \frac{5}{n_{in}} * \sum_{k=1+(i-1)0.2n_{in}}^{i*0.2n_{in}} T_k \quad (31)$$

Az 5 vezérpontra így már alkalmazható a következő matematikai tétel [34]: a síkban minden 5 pontból álló halmaz esetén létezik egy a pontokra illeszkedő másodrendű görbe M_{fit} , ami az alábbi implicit egyenlettel írható le:

$$M_{fit} = Ax^2 + By^2 + Cxy + Dx + Ey + F = 0 \quad (32)$$

Az egyenlet együtthatóit az alábbi módszerrel lehet meghatározni. Először is 4 vezérpont felhasználásával felírunk 4 egyenest, melyek mindegyike 2 vezérponton halad át a következő elrendezésben: $L_1 = P_2P_3$; $L_2 = P_1P_4$; $L_3 = P_3P_4$ és $L_4 = P_1P_2$. Ezen L_l egyenesek egyenletei felírhatóak $\mathbf{x}_h = (x_{h_1}, x_{h_2}, x_{h_3})^T = (x, y, 1)^T$ homogén koordináták segítségével:

$$L_l = g_l x_{h_1} + h_l x_{h_2} + k_l x_{h_3} \mid l = 1,2,3,4 \quad (33)$$

Itt a g_l , h_l és k_l változók az egyenlet együtthatói. Két ilyen egyenes egyenletének összesorzásával egy-egy másodrendű görbét kapunk, amely az 5 vezérpontból 4-en keresztülhalad és tagja ugyanannak a másodrendű görbesornak, mint a keresett illesztés:

$$M_1 = L_1 * L_2 = 0 \quad (34)$$

$$M_2 = L_3 * L_4 = 0 \quad (35)$$

Mátrixos alakban felírva a kapott másodrendű görbék egyenletét, példaként az M_1 görbén bemutatva:

$$Q_{M_1} = \begin{bmatrix} g_1g_2 & \frac{g_1h_2 + g_2h_1}{2} & \frac{g_1k_2 + g_2k_1}{2} \\ \frac{g_1h_2 + g_2h_1}{2} & h_1h_2 & \frac{h_1k_2 + h_2k_1}{2} \\ \frac{g_1k_2 + g_2k_1}{2} & \frac{h_1k_2 + h_2k_1}{2} & k_1k_2 \end{bmatrix} \quad (36)$$

A másodrendű görbesor általános egyenlete az M_1 és M_2 görbék lineáris kombinációjaként írható fel:

$$Q_{M_{fit}} = \begin{bmatrix} A & 0.5C & 0.5D \\ 0.5C & B & 0.5E \\ 0.5D & 0.5E & F \end{bmatrix} = \beta Q_{M_1} - \alpha Q_{M_2} \quad (37)$$

Ennek a görbesornak keressük azt a tagját, amely áthalad az 5. vezérponton. Ez a feltétel a következő α, β együtthatók mellett teljesül:

$$\alpha = [x_5 \quad y_5 \quad 1] \times Q_{M_1} \times \begin{bmatrix} x_5 \\ y_5 \\ 1 \end{bmatrix} \quad (38)$$

$$\beta = [x_5 \quad y_5 \quad 1] \times Q_{M_2} \times \begin{bmatrix} x_5 \\ y_5 \\ 1 \end{bmatrix} \quad (39)$$

Az így meghatározott implicit egyenlet nem túl könnyen alkalmazható további számításokra, így szükséges átkonvertálni egy jóval könnyebben kezelhető explicit koordinátafüggvény alakba. Első lépésként φ_m szöggel elforgatjuk a kapott implicit egyenletet az alábbi módon:

$$\begin{pmatrix} x_R \\ y_R \end{pmatrix} = \begin{bmatrix} \cos\varphi_m & -\sin\varphi_m \\ \sin\varphi_m & \cos\varphi_m \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \quad (40)$$

$$ax_R^2 + by_R^2 + cx_Ry_R + dx_R + ey_R + F = 0 \quad (41)$$

Itt az elforgatás szögét úgy határozzuk meg, hogy az eredeti implicit egyenlet $C * xy$ tagját kiejtsük, azaz a $c * x_Ry_R = 0$ egyenlet teljesüljön. A c együttható egyenlete az elforgatásból való levezetéssel kapható meg, melynek eredménye:

$$c = (B - A) * \sin(2\varphi_m) + C * \cos(2\varphi_m) = 0 \quad (42)$$

Ebből átrendezéssel meghatározható a szükséges elforgatási szög:

$$\varphi_m = \frac{1}{2} \tan^{-1} \left(\frac{C}{A - B} \right) \quad (43)$$

Az elforgatás után kapott egyenlet már kanonikus alakra hozható. Amennyiben az 5 vezérpont közel semelyik 3 nem kollineáris az a, b együtthatók egyike garantáltan nem zérus. Ha valamely 3 vezérpont kollineáris, akkor a becsült pályát egy rájuk illesztett egyenessel közelítjük. Ha az $a \neq 0$ és $b \neq 0$ feltételek mindegyike teljesül, akkor a kapott pálya egy ellipszis vagy hiperbola, melynek általános kanonikus alakja:

$$\frac{a}{f} \left(x_R + \frac{d}{2a} \right)^2 + \frac{b}{f} \left(y_R + \frac{e}{2b} \right)^2 = 1 \quad (44)$$

Ha $a/f > 0$ és $b/f > 0$ mindegyike teljesül akkor a pálya ellipszis, az alábbi specifikus kanonikus alakkal:

$$\frac{(x_R - u)^2}{a_c^2} + \frac{(y_R - v)^2}{b_c^2} = 1 \quad (45)$$

Ha vagy $a/f < 0$ vagy $b/f < 0$, akkor a pálya egy hiperbola, kanonikus alakja pedig:

$$\pm \frac{(x_R - u)^2}{a_c^2} \mp \frac{(y_R - v)^2}{b_c^2} = 1 \quad (46)$$

Amennyiben pedig $a = 0$ vagy $b = 0$ határesetre fennáll, akkor a pálya parabola az alábbi kanonikus alakkal:

$$y_R = m(x_R - u)^2 + v \quad \text{vagy} \quad x_R = m(y_R - u)^2 + v \quad (47)$$

A kanonikus alakok előnye, hogy az együtthatókat felírva közvetlenül felírhatóak a másodrendű görbék koordinátafüggvényei az alábbiak szerint. Ellipszis esetén:

$$\begin{pmatrix} x_R(p) \\ y_R(p) \end{pmatrix} = \begin{pmatrix} a_c \cdot \cos p + u \\ b_c \cdot \sin p + v \end{pmatrix} \quad (48)$$

Hiperbola pálya esetén (kelet-nyugati irányba nyitott vagy észak-déli irányba nyitott esetekben):

$$\begin{pmatrix} x_R(p) \\ y_R(p) \end{pmatrix} = \begin{pmatrix} s \cdot a_c \cdot \cosh(p) + u \\ b_c \cdot \sinh(p) + v \end{pmatrix} \quad \text{vagy} \quad \begin{pmatrix} x_R(p) \\ y_R(p) \end{pmatrix} = \begin{pmatrix} a_c \cdot \sinh(p) + u \\ s \cdot b_c \cdot \cosh(p) + v \end{pmatrix} \quad (49)$$

Itt az $s = \pm 1$ változó értéke adja meg, hogy a hiperbola melyik ága a becsült pálya, amely az alábbiak szerint számolható:

$$s = \text{sign}\left(\frac{x_R(5) - u}{a_c}\right) \quad \text{vagy} \quad s = \text{sign}\left(\frac{y_R(5) - v}{b_c}\right) \quad (50)$$

Végezetül parabola esetén (kelet-nyugati irányba nyitott vagy észak-déli irányba nyitott esetekben):

$$\begin{pmatrix} x_R(p) \\ y_R(p) \end{pmatrix} = \begin{pmatrix} m(p - u)^2 + v \\ p \end{pmatrix} \quad \text{vagy} \quad \begin{pmatrix} x_R(p) \\ y_R(p) \end{pmatrix} = \begin{pmatrix} p \\ m(p - u)^2 + v \end{pmatrix} \quad (51)$$

Visszaforgatva a kapott egyenletet $-\varphi_m$ szöggel az eredeti koordináta rendszerbe:

$$\begin{pmatrix} x_{pred}(p) \\ y_{pred}(p) \end{pmatrix} = \begin{bmatrix} \cos \varphi_m & \sin \varphi_m \\ -\sin \varphi_m & \cos \varphi_m \end{bmatrix} \times \begin{pmatrix} x_R(p) \\ y_R(p) \end{pmatrix} \quad (52)$$

A becsült pályára kapott végleges koordinátafüggvények:

$$x_{pred}(p) = \begin{cases} q_{11} \cos(p) + q_{12} \sin(p) + q_{13}, & \text{ha ellipszis} \\ q_{11} \cosh(p) + q_{12} \sinh(p) + q_{13}, & \text{ha hiperbola} \\ q_{11} p^2 + q_{12} p + q_{13}, & \text{ha parabola} \end{cases} \quad (53)$$

$$y_{pred}(p) = \begin{cases} q_{21} \cos(p) + q_{22} \sin(p) + q_{23}, & \text{ha ellipszis} \\ q_{21} \cosh(p) + q_{22} \sinh(p) + q_{23}, & \text{ha hiperbola} \\ q_{21} p^2 + q_{22} p + q_{23}, & \text{ha parabola} \end{cases} \quad (54)$$

Megfigyelhető, hogy a másodrendű görbe típusától függetlenül a pálya a $Q_{pred} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \end{bmatrix}$ együtthatók segítségével megadható.

A koordinátafüggvény alakban felírás legnagyobb előnye, hogy a függvény p paramétere szinkronizálható a valós idővel, így egy explicit időfüggvényt kaphatunk a becsült mozgásról.

Ehhez ki kell számolnunk a p paraméter értékét a 4. és 5. vezérpont esetén az alábbi egyenletek segítségével:

$$x_5 = x(p_5) \text{ és } y_5 = y(p_5) \quad (55)$$

$$x_4 = x(p_4) \text{ és } y_4 = y(p_4) \quad (56)$$

A megoldás ellipszis esetén:

$$p_i = \begin{cases} \arccos\left(\frac{x_R(i) - u}{a_c}\right), \text{ ha } \frac{y_R(i) - v}{b_c} \geq 0 \\ 2\pi - \arccos\left(\frac{x_R(i) - u}{a_c}\right), \text{ ha } \frac{y_R(i) - v}{b_c} < 0 \end{cases} \mid i = 4,5 \quad (57)$$

Hiperbola pálya esetén (kelet-nyugati vagy észak-déli):

$$p_i = \operatorname{asinh}\left(\frac{y_R(i) - v}{b_c}\right) \text{ vagy } p_i = \operatorname{asinh}\left(\frac{x_R(i) - u}{a_c}\right) \mid i = 4,5 \quad (58)$$

Parabola esetén (kelet-nyugati vagy észak-déli):

$$p_i = y_R(i) \text{ vagy } p_i = x_R(i) \quad (59)$$

A vezérpontok mérési idejét ismerve pedig már kiszámítható a valós idő és a p paraméter közötti szinkronizáló skálázás értéke:

$$\delta_{pred} = \frac{p_5 - p_4}{T_5 - T_4} \quad (60)$$

A becsült pálya végső időfüggő (T) mozgásegyenlete ez alapján:

$$\begin{pmatrix} x_{pred}(T) \\ y_{pred}(T) \end{pmatrix} = \begin{pmatrix} x_{pred}(p_5 + \delta_{pred} * T) \\ y_{pred}(p_5 + \delta_{pred} * T) \end{pmatrix} \quad (61)$$

2.8. Dinamikus komponens

A dinamikus komponens szerepe a becsült jövőbeli információk alapján a mozgó akadályok elkerülése. Az alrendszer fő bemenete a pályabecsítő algoritmus eredményeként kapott becsült jövőbeli elmozdulás-idő függvények:

$$\mathbf{r}_{pred_j}(t) = \begin{pmatrix} x_{pred_j}(t) \\ y_{pred_j}(t) \end{pmatrix} \quad (62)$$

Az összesen detektált n_{din} mozgó akadály mindegyikéhez van rendelve egy ilyen becsült pálya $j \in N \mid j \leq n_{din}$ indexszel ellátva. A cél, hogy a mozgásirány megválasztásakor ezen becsült pályák környezetét elkerüljük. Ennek érdekében kiszámítunk n_{pred} darab diszkrét jövőbeli pozíciót, p indexszel ($p \in N \mid p \leq n_{pred}$), amelyek taszítóhatást fejtenek ki a robotra a pálya hosszában, melynek nagysága:

$$f_{din_j}(p) = \begin{cases} 0, & \text{ha } d_{pred_j} > d_{max} \\ g_{din} h_{din_j} e^{-h_{din_j} * d_{pred_j}^2(p)}, & \text{egyébként} \end{cases} \quad (63)$$

$$\left(d_{pred_j}(p), \gamma_j(p) \right) = \left(x_{pred_j}(t_p), y_{pred_j}(t_p) \right)^{D \rightarrow p} \quad (64)$$

Itt $t_p = p * T_{pred}/n_{pred}$ és a h_{din_j} együttható értéke a dinamikus akadály méretétől függ az alábbi összefüggés alapján:

$$h_{din_j} = \frac{-\ln\left(\frac{g_{cél}}{2g_{din}}\right)}{\left(r_r + r_{bizt} + \sigma_{din_j}\right)^2} \quad (65)$$

$$\sigma_{din_j} = \frac{1}{2} |\mathbf{G}_1(k_m) \mathbf{G}_2(k_m)|, \text{ ha } \gamma_j(p_0) \in \Lambda(\mathbf{g}_1(k_m), \mathbf{g}_2(k_m)) \exists k_m \leq n_{gr} \quad (66)$$

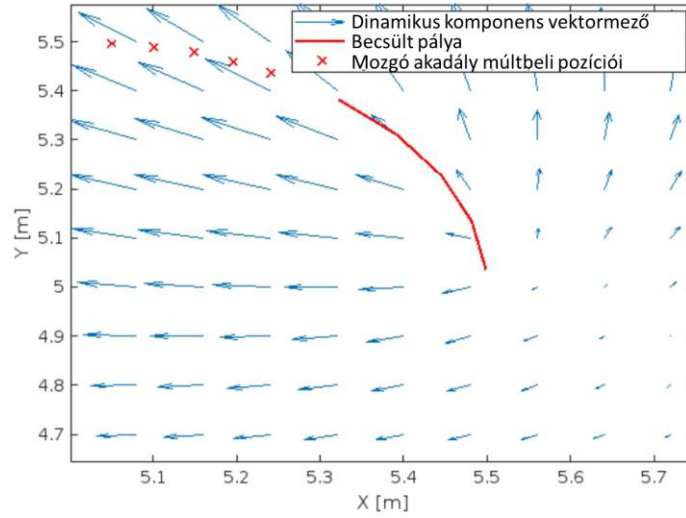
A σ_{din_j} paraméter a mozgó akadály köré írható kör sugarát becsüli meg a dinamikus objektum akadálycsoportjának mérete alapján. Minél nagyobb egy mozgó akadály annál nagyobb mértékű taszító hatást fejt ki a becsült pályája a robot mozgására. Ennek célja a pályabecslés pontatlanságaiból adódó esetleges hibák kiszűrése. A taszítóhatás iránya az alábbi módon számolható ki. A becsült pálya érintővektorát bármely pontban felírhatjuk a koordinátafüggvények deriváltjaként:

$$\mathbf{q}_j(t) = \frac{d}{dt} \mathbf{r}_{pred_j}(t) = \begin{pmatrix} \dot{x}_{pred_j}(t) \\ \dot{y}_{pred_j}(t) \end{pmatrix} \quad (67)$$

A taszítóhatás iránya az érintővektorral ellentétes irány és a pálya pontjaiból húzott merőleges irány eredőjeként áll össze:

$$\mathbf{e}_{din_j}(p) = - \begin{pmatrix} \cos(\gamma_j(p) + \varphi_r) \\ \sin(\gamma_j(p) + \varphi_r) \end{pmatrix} - \frac{\mathbf{q}_j(t_p)}{|\mathbf{q}_j(t_p)|} \quad (68)$$

Ez egy olyan vektorteret eredményez a becsült pálya körül, ami annak megkerülésére ösztönzi a robotot, ahogy azt a 7. ábra illusztrálja.



7. ábra: Mozgó akadály becsült pályája által keltett taszító hatás vektormezője

Egy adott mozgó akadály dinamikus komponense a becsült pályán felvett diszkrét pontok hatásának eredőjéből adódik:

$$\mathbf{F}_{pred_j} = \frac{5}{n_{pred}} \sum_{p=1}^{n_{pred}} f_{din_j}(p) \mathbf{e}_{din_j}(p) \quad (69)$$

A pályabecslés bizonytalanságainak kompenzálása céljából ehhez még hozzáadunk egy a mozgó akadály jelen pozíciója (p_0 paraméter mellett) által kifejtett taszító hatást is, melynek nagysága:

$$f_{jel_j} = g_{jel} * h_{din_j} * e^{-h_{din_j} * d_{pred_j}(p_0)^2} \quad (70)$$

Itt a g_{jel} együttható értéke a g_{din} együtthatóval egyezik meg, ha az akadály közeledik a robotoz és annak a fele, amennyiben távolodik tőle:

$$g_{jel} = \begin{cases} 0.5g_{din}, & \text{ha } (\mathbf{R}_{lid}(\gamma_j(p_0)) < d_{pred_j}(p) \forall p \leq n_{pred}) \\ g_{din}, & \text{egyébként} \end{cases} \quad (71)$$

Ezt követően az összes detektált dinamikus akadályra összegezve megkapjuk a teljes dinamikus komponens:

$$\mathbf{F}_{din} = \sum_{j=1}^{n_{din}} \mathbf{F}_{pred_j} - f_{jel_j} \begin{pmatrix} \cos(\gamma_j(p_0) + \varphi_r) \\ \sin(\gamma_j(p_0) + \varphi_r) \end{pmatrix} \quad (72)$$

2.9. Utazási irány és sebesség

A rendszer 3 komponensének összegzésével megkaphatjuk a pillanatnyi helyzetben választott utazási irányt ($\delta_{vál}$) az alábbi módon:

$$\mathbf{F}_{er} = \begin{pmatrix} F_{er_x} \\ F_{er_y} \end{pmatrix} = \mathbf{F}_{cél} + \mathbf{F}_{stat} + \mathbf{F}_{din} \quad (73)$$

$$\delta_{vál} = \tan^{-1} \left(\frac{F_{er_y}}{F_{er_x}} \right) \quad (74)$$

Itt $\mathbf{F}_{cél}$ a célkomponens súlyozott irányvektora az alábbiak szerint:

$$\mathbf{F}_{cél} = g_{cél} \begin{pmatrix} \cos(\delta_{cél} + \varphi_r) \\ \sin(\delta_{cél} + \varphi_r) \end{pmatrix} \quad (75)$$

A három komponens súlyozása a következő együtthatókon keresztül valósul meg: $g_{cél} = 60$; $g_{st} = 500$ és $g_{din} = 40$.

A kívánt utazási irány meghatározását követően meg kell adnunk a kívánt utazási sebességet is. Ehhez először kiszámoljuk a pillanatnyi helyzetben megengedhető maximális sebességet (v_{lim}) a robot orra előtti tartományban található akadályok távolsága és a robot maximális gyorsulása/lassulása (a_{max}) alapján:

$$v_{lim} = \sqrt{0.65 a_{max} (\min(\mathbf{R}_{lid}(\theta_{orr})) - r_r - r_{bzt})} \quad (76)$$

$$\theta_{orr} \in V(\delta_{seb} - \pi/6, \delta_{seb} + \pi/6) \quad (77)$$

Ahol $\delta_{seb} = \tan^{-1}(v_y/v_x) - \varphi_r$ a jelenlegi sebességvektor iránya. Statikus környezetben való navigáció során a fő szempontok, hogy a robot ne lépje túl ezt maximális sebességet, illetve hogy a korábbi sebességhez képest ne változzon sokat az új érték (ne tegyük ki indokolatlanul nagy gyorsulásoknak és ezáltal igénybevételnek a robotot). Ezek alapján a statikus helyzetekben kívánt sebesség az alábbiak szerint számolható:

$$u_{st} = (u_{seb} + \text{sgn}(v_{lim} - u_{seb}) * \Delta u) * \mu_{korm} \quad (78)$$

Itt $u_{seb} = \sqrt{v_x^2 + v_y^2}$ a jelenlegi sebességvektor nagysága, $\Delta u = 0.05 \text{ m/s}^2$ az egy iterációban megengedett sebességváltozás és μ_{korm} a kormányzási szög hatása:

$$\mu_{korm} = e^{-0.15 * \Delta \varphi (\delta_{vál}, \delta_{seb})^2} \quad (79)$$

Dinamikus helyzetekben a robot sebességét elsősorban a mozgó akadályok határozzák meg. A cél hogy közeledő akadályok esetén gyorsabban térjünk ki. Ennek érdekében a korábban kiszámolt dinamikus komponens alapján adjuk meg a dinamikus helyzetekben kívánt sebesség nagyságát:

$$u_{din} = \frac{v_{max} + v_{min}}{2} + \frac{v_{max} - v_{min}}{\pi} \tan^{-1} \left(\frac{\mathbf{F}_{din} \cdot \mathbf{F}_{er}}{g_{din} |\mathbf{F}_{er}|} \right) \quad (80)$$

Itt $v_{min} = 0.1 \text{ m/s}$ és $v_{max} = 0.8 \text{ m/s}$ az operátor által a működés előtt beállított abszolút maximum és minimum sebességkorlátok (a robot konstrukciójából, motorok teljesítményéből, működési környezettől függő határok, alapvetően a cél hogy a robot sebessége ebben a

tartományban maradjon működés során). Általános helyzetben a kívánt sebesség a korábban bemutatott statikus és dinamikus eset súlyozott átlagaként adható meg:

$$u_{val} = (1 - \tau_{din}) * u_{st} + \tau_{din} * u_{din} \quad (81)$$

Itt $\tau_{din} = 0.5$ a súlyozási paraméter, ami főleg annak függvényében adható meg mennyire zsúfolt mozgó akadályokban a munkatér.

2.10. Ütközésselkerülés megállással

Bizonyos helyzetek előnyösebb viselkedésforma a robot számára, hogyha megáll és engedi a mozgó akadályokat elhaladni, ahelyett hogy megkerülné őket a mozgásuk közben. Ezzel lerövidíthető a megtett út (helyzettől függően a menetidő), illetve javítható az ember-robot interakció is. Például egy étteremben vagy irodában az emberek sokkal pozitívabban viszonyulnak egy olyan önállóan mozgó robothoz, ami utat enged nekik a körül manőverezés helyett, mert sokkal kiszámíthatóbb és kisebb eséllyel okoz váratlan helyzetet. Ez növeli a rendszerbe vetett bizalmat és komfortérzetet is. Ezen viselkedésforma implementálására az általam fejlesztett navigációs algoritmusba az alábbi módszert dolgoztam ki.

Ez a megközelítés csak olyan mozgó akadályok esetén releváns, amelyek közvetlenül a robot előtt haladnak el. Továbbá el szeretnénk kerülni a robot mögött haladó mozgó akadályokkal ütközést is. Tehát a megállással történő akadályelkerülés csak akkor kerül szóba, ha az alábbi 2 feltétel teljesül:

$$\mathbf{F}_{er} - \mathbf{F}_{din} \in V(\gamma_{jel}(1), \gamma_{jel}(n_{pred})) \mid \exists j_{el} \leq n_{din} \quad (82)$$

$$d_{pred_j}(p) \geq 1 \mid \forall p \leq n_{pred}; \forall j \neq j_{el} \quad (83)$$

Más szóval akkor vizsgáljuk a megálló viselkedés meghívását, hogyha legalább egy mozgó akadály a robot előtt halad át, és minden egyéb mozgó akadály (robot mellett, mögött haladó) becsült jövőbeli pozíciói a robot 1 méteres környezetén kívül találhatóak. Emellett, ha a túl közel van az előtte haladó akadály becsült pályájához, akkor nem aktiválja a megálló viselkedést, hanem áthalad a dinamikus akadály előtt. Ez az eset az alábbi feltétellel írható le:

$$d_{fék} + r_r + r_{bizt} + \sigma_{din_{jel}} > d_{pred_{jel}}(p) \mid \forall p \leq n_{pred} \quad (84)$$

$$d_{fék} = \frac{|v_{lin}|^2}{2 * 0.67 a_{max}} \quad (85)$$

Ahol $v_{lin} = [v_x; v_y]^T$ a robot lineáris sebessége, $d_{fék}$ pedig a megálláshoz szükséges biztonságos féktávolság.

Mivel a mozgó akadályok kezelése teljes egészében a dinamikus komponens hatóköre, a hatásuk mértékét reprezentálhatjuk a dinamikus komponens többi komponenshez képest vett

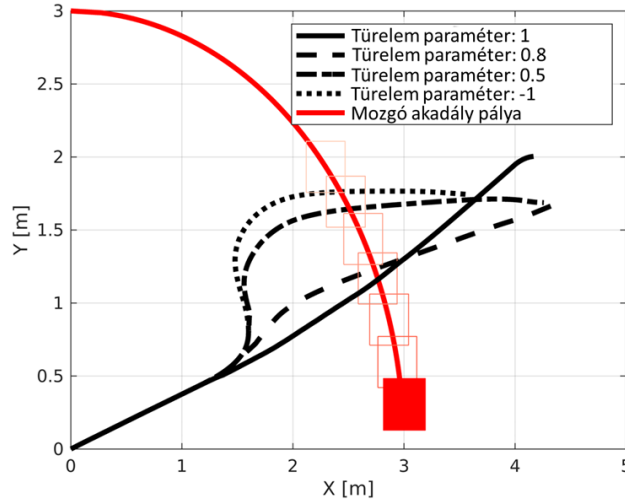
nagyságával. A dinamikus komponenst az eredő irányvektorból kivonva megadható, hogy a mozgó akadályok hatására milyen mértékben készül elkanyarodni a robot. A mozgó akadály hatására létrejött kívánt utazási irányvektor és a mozgó akadály nélküli utazási irányvektor által bezárt szöget dinamikus eltérülésnek (μ_{elt}) neveztem el, ami az alábbi módon számolható:

$$\cos(\mu_{elt}) = \frac{\mathbf{F}_{er} \cdot (\mathbf{F}_{er} - \mathbf{F}_{din})}{|\mathbf{F}_{er}| * |\mathbf{F}_{er} - \mathbf{F}_{din}|} \quad (86)$$

Ez a dinamikus eltérülés felhasználható annak eldöntésére, hogy mikor álljon meg a robot és engedje el a mozgó akadály maga előtt. A viselkedésmód testreszabhatósága érdekében bevezetünk egy „türelem” paramétert ($\psi_{tür}$), amivel befolyásolható milyen mértékben hajlandó a robot eltérni a pályájától a mozgó akadály elkerülése érdekében, ahelyett hogy megvárna amíg elhalad előtte. Ez alapján a megállás feltétele a következőképpen definiálható:

$$\cos(\mu_{elt}) < \psi_{tür} \quad (87)$$

A türelem paraméter értéke $\psi_{tür} \in [-1,1]$ tartományban mozoghat, a -1 érték esetén a megálló viselkedés inaktív és a robot mindig megpróbálja megkerülni a mozgó akadályokat, míg a 1 érték esetén a robot a legtürelemesebb és mindig megpróbálja elengedni a dinamikus objektumokat maga előtt. A különböző türelem értékek bejárt útra vonatkozó hatását a 8. ábra illusztrálja.



8. ábra: A türelem paraméter értékének hatása a robot pályájára [31]

3. Eredmények és azok értékelése

3.1. Robot modell

A szimulációs eredmények pontosságához szükséges a valóságot megfelelően közelítő modell alkalmazása a robot kiadott navigációs parancsok hatására történő mozgásának nyomonkövetésére. A modellezett rendszer egy egymással 120° -os szöget bezáróan elhelyezett

3 kerékkal rendelkező „omnidirectional” robot, így a szimulációban 4 összekapcsolt merev testként (a robottest valamint a 3 kerék) modelleztem. Minden kerék egy k_g áttételen keresztül egy permanens mágneses DC motorral van hajtva. A rendszer bemeneteként a 3 motorra kapcsolt feszültségek ($U_1; U_2; U_3$) szolgálnak, kimenetként pedig a robot pillanatnyi sebességvektorát tekintjük. A motorok szögsebessége ($\omega_{m_1}; \omega_{m_2}; \omega_{m_3}$) és a rajtuk átfolyó áramerősségek nagysága ($I_1; I_2; I_3$) a külső gerjesztésű DC motorok általános egyenletével adható meg:

$$\frac{d}{dt}I_i = \frac{1}{L_m} * U_i - \frac{R_\Omega}{L_m} * I_i - \frac{K_\tau}{L_m} * \omega_{m_i} \mid i = 1,2,3 \quad (88)$$

$$J_m * \frac{d}{dt}\omega_{m_i} = K_\tau * I_i - v_m * \omega_{m_i} - M_{t_i} \mid i = 1,2,3 \quad (89)$$

Ahol J_m a motor inerciája, v_m a viszkózus súrlódási együtthatója, K_τ a nyomaték konstans, R_Ω az armatúra ellenállás, L_m az induktivitás és M_{t_i} a terhelő nyomaték. A motorok és kerekek közötti áttétel miatt a kerekek nyomatéka (M_i) és szögsebessége (ω_i) az alábbi módon adható meg:

$$\omega_{m_i} = k_g * \omega_i \quad (90)$$

$$M_i = k_g * M_{t_i} \quad (91)$$

Behelyettesítve a (89)-es egyenletbe:

$$J_m * \frac{d}{dt}\omega_i = \frac{K_\tau}{k_g} * I_i - v_m * \omega_i - \frac{1}{k_g^2} * M_i \quad (92)$$

Tisztán gördülés esetén (a kerekek megcsúszása nélküli mozgás, amire törekszünk a navigáció során a lokalizáció pontossága kedvéért) teljesül az alábbi feltétel:

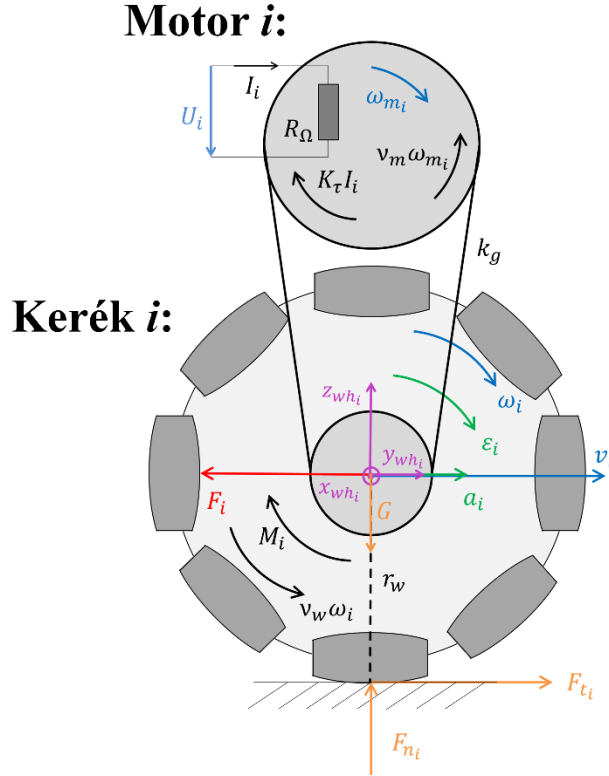
$$\frac{d}{dt}\omega_i = \frac{a_i}{r_k} \quad (93)$$

Ilyen esetben egy kerék mozgásegyenlete az alábbi módon írható fel ($i = 1,2,3$):

$$m_k * a_i = F_{t_i} - F_i \quad (94)$$

$$J_k * \frac{a_i}{r_k} = M_i - v_k * \omega_i - F_{t_i} * r_k \quad (95)$$

Ahol J_k a kerék inerciája, v_k a viszkózus súrlódási együttható, r_k a kerék sugara, m_k a kerék tömege, F_{t_i} a talppontban kifejtett tapadási erő és F_i a robottest csatlakozási pontjában kifejtett erő. Egy hajtott kerék modelljét a 9. ábra illusztrálja.



9. ábra: Egy hajtott kerék modellje [32]

Behelyettesítve a (92)-as egyenletbe a (93)-est és kifejezve a kerekre ható hajtónyomatékokat:

$$M_i = K_\tau k_g * I_i - v_m k_g^2 * \omega_i - \frac{J_m k_g^2}{r_k} * a_i \quad (96)$$

Behelyettesítve ezt, illetve (94)-öt a (95)-os egyenletbe és kifejezve a kerek és robottest csatlakozási pontjában ébredő erőt:

$$J_k * \frac{a_i}{r_k} = K_\tau k_g * I_i - v_m k_g^2 * \omega_i - \frac{J_m k_g^2}{r_k} * a_i - v_k * \omega_i - r_k * (m_k a_i + F_i) \quad (97)$$

$$F_i = \frac{K_\tau k_g}{r_k} * I_i - \frac{v_m k_g^2 + v_k}{r_k} * \omega_i - \left(\frac{J_m k_g^2 + J_k}{r_k^2} + m_k \right) * a_i \quad (98)$$

A kerek és robottest csatlakozási pontjaiban ható ($F_1; F_2; F_3$) erők befolyásolják a robottest mozgását. A robot testkoordinátarendszerben felírt mozgásegyenletei a kerek geometriai helyzetét ($\theta_1 = 0^\circ; \theta_2 = 120^\circ; \theta_3 = -120^\circ$) figyelembe véve az alábbi formában írhatóak fel:

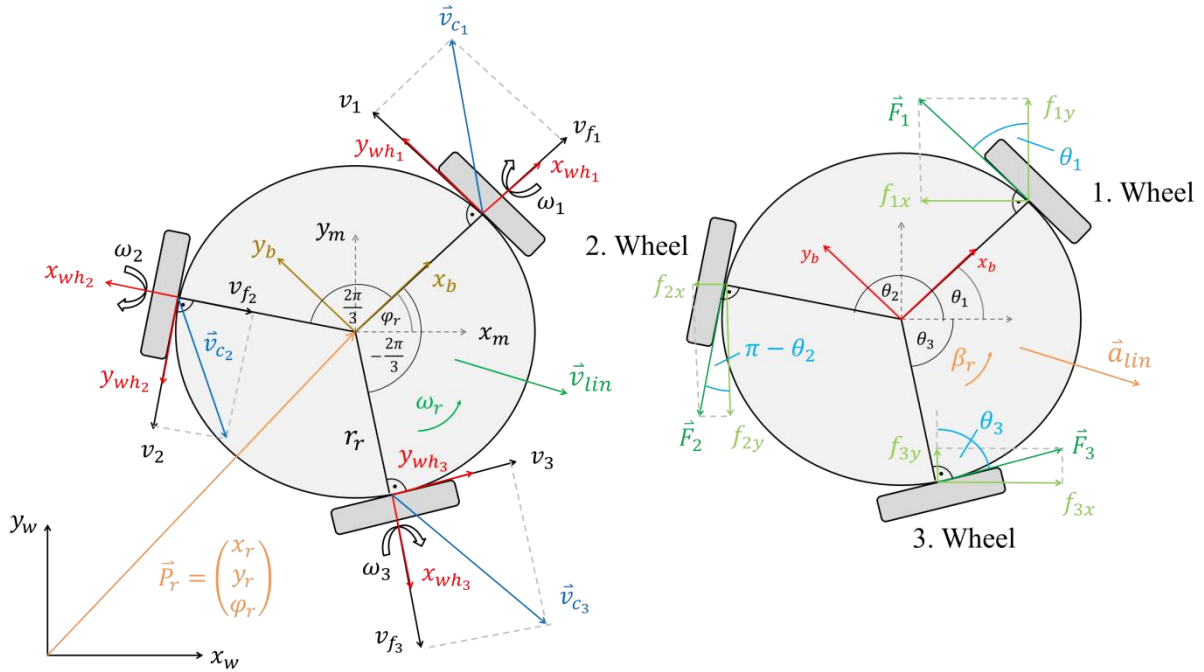
$$\begin{pmatrix} m_r * a_{t_x} \\ m_r * a_{t_y} \\ J_r * \beta_r \end{pmatrix} = \begin{bmatrix} -\sin(\theta_1) & -\sin(\theta_2) & -\sin(\theta_3) \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) \\ r_r & r_r & r_r \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \mathbf{D}_T \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \quad (99)$$

Ahol J_r a robottest inerciája, r_r a robot sugara, $\mathbf{D}_T$ a transzformációs mátrix és $[a_{t_x}; a_{t_y}]$ a testkoordinátarendszerbeli gyorsulásvektor. A transzformációs mátrix szintén felhasználható a

robottest gyorsulása és a kerekek csatlakozási pontjainak gyorsulása közötti összefüggés megadására az alábbi módon:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{bmatrix} -\sin(\theta_1) & \cos(\theta_1) & r_r \\ -\sin(\theta_2) & \cos(\theta_2) & r_r \\ -\sin(\theta_3) & \cos(\theta_3) & r_r \end{bmatrix} \times \begin{bmatrix} a_x \\ a_y \\ \beta_r \end{bmatrix} = \mathbf{D}_T^{-1} \times \begin{bmatrix} a_x \\ a_y \\ \beta_r \end{bmatrix} \quad (100)$$

A robottest és kerekek egymáshoz viszonyított pozícióját és a transzformációs mátrix levezetését a 10. ábra illusztrálja.



10. ábra: A modellezett rendszert alkotó merev testek egymáshoz viszonyított helyzete és kölcsönhatása [32]

A (88), (93) és (98)-(100) egyenletek alapján felírható a modellezett rendszer állapotter egyenlete az alábbi módon:

$$\frac{d}{dt} \mathbf{x}_{mod} = \mathbf{A}_{mod} \times \mathbf{x}_{mod} + \begin{bmatrix} \frac{1}{L_m} \mathbf{I}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \end{bmatrix} \times \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \quad (101)$$

$$\mathbf{A}_{mod} = \begin{bmatrix} -\frac{R_\Omega}{L_m} \mathbf{I}_{3 \times 3} & -\frac{K_\tau k_g}{L_m} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \frac{1}{r_k} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & -\mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \frac{K_\tau k_g}{r_k} \mathbf{D}_M & -\frac{v_m k_g^2 + v_k}{r_k} \mathbf{D}_M & \left(\frac{J_m k_g^2 + J_k}{r_k^2} + m_k \right) \mathbf{D}_M & -\mathbf{I}_{3 \times 3} \end{bmatrix} \quad (102)$$

$$\mathbf{x}_{mod} = [I_1 \quad I_2 \quad I_3 \quad \omega_1 \quad \omega_2 \quad \omega_3 \quad a_1 \quad a_2 \quad a_3 \quad a_{tx} \quad a_{ty} \quad \beta_r]^T \quad (103)$$

Ahol $\mathbf{A}_{mod}$ a rendszermátrix, $\mathbf{x}_{mod}$ az állapotváltozók vektora és $\mathbf{D}_M$ a inercia redukált transzformációs mátrix:

$$\mathbf{D}_M = \begin{bmatrix} -\frac{\sin(\theta_1)}{m_r} & -\frac{\sin(\theta_2)}{m_r} & -\frac{\sin(\theta_3)}{m_r} \\ \frac{\cos(\theta_1)}{m_r} & \frac{\cos(\theta_2)}{m_r} & \frac{\cos(\theta_3)}{m_r} \\ \frac{r_r}{J_r} & \frac{r_r}{J_r} & \frac{r_r}{J_r} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\sqrt{3}}{2m_r} & \frac{\sqrt{3}}{2m_r} \\ 1 & -1 & -1 \\ \frac{r_r}{J_r} & \frac{r_r}{J_r} & \frac{r_r}{J_r} \end{bmatrix} \quad (104)$$

A kimeneti egyenlet:

$$\begin{bmatrix} a_x \\ a_y \\ \beta_r \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos(\varphi_r) & \sin(\varphi_r) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sin(\varphi_r) & \cos(\varphi_r) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \mathbf{x}_{mod} \quad (105)$$

Az így megkapott világkoordinátarendszerbeli gyorsulásvektort integrálva megkapható a robot pillanatnyi sebessége és pozíciója, amivel a szimuláció már elvégezhető:

$$\mathbf{v}_r = \begin{pmatrix} v_x \\ v_y \\ \omega_r \end{pmatrix} = \int \mathbf{a}_r dt = \int \begin{pmatrix} a_x \\ a_y \\ \beta_r \end{pmatrix} dt \quad (106)$$

$$\mathbf{p}_r = \begin{pmatrix} p_x \\ p_y \\ \omega_r \end{pmatrix} = \iint \mathbf{a}_r dt = \iint \begin{pmatrix} a_x \\ a_y \\ \beta_r \end{pmatrix} dt \quad (107)$$

A navigációs algoritmus kimeneteként megkapott kívánt utazási irány és sebesség értékek az alábbi módon számolhatóak át a motorra kapcsolt feszültség beavatkozáj értékké (ami a robotmodell bemenete):

$$\boldsymbol{\omega}_{ref} = \begin{pmatrix} \omega_{ref1} \\ \omega_{ref2} \\ \omega_{ref3} \end{pmatrix} = \frac{1}{r_k} \mathbf{D}_T^{-1} \times \begin{pmatrix} u_{vál} * \cos(\delta_{vál} - \varphi_r) \\ u_{vál} * \sin(\delta_{vál} - \varphi_r) \\ 0.2 * \Delta\varphi(\delta_{vál}, \varphi_r) \end{pmatrix} \quad (108)$$

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = P_\omega * \left(\boldsymbol{\omega}_{ref} - \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \right) + I_\omega * \int \left(\boldsymbol{\omega}_{ref} - \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \right) dt \quad (109)$$

Ahol P_ω és I_ω az együtthatói a motorok PI szabályzójának és $\boldsymbol{\omega}_{ref}$ a kerekek referencia szögsebessége. A feszültség beavatkozáj értéke szaturálva van a $[-12; 12]V$ -os tartományban.

3.2. Teljesítménymutatók

A szakirodalomban ismertett navigációs algoritmusokkal való összehasonlításhoz szükség van egységes teljesítménymutatókra, amelyekkel jellemezni lehet az adott algoritmus hatékonyságát. A legkézenfekvőbb indikátoron – ami a navigáció sikeressége (a cél elérése

ütközés nélkül) – az alábbi táblázatban ismertetett mutatókat használom fel a továbbiakban [24], [35].

2. Táblázat: A vizsgálat során használt teljesítmény mutatók

Név	Jelölés	Kiszámítás	Mértékegység
Utazási idő	T_{nav}	A cél eléréséig eltelt idő	s
Megtett táv	$l_{út}$	$l_{út} = \int_0^{T_{nav}} \mathbf{v}_{lin}(t) dt$	m
Teljes elfordulás	φ_{tel}	$\varphi_{tel} = \int_0^{T_{nav}} \omega_r(t) dt$	rad
Átlagos pillanatnyi kormányzás	$AVSA$	$AVSA = \frac{1}{T_{nav}} * \int_0^{T_{nav}} \delta_{vál}(t) - \delta_{seb}(t) dt$	rad
Átlagos görbületi változás	CC	$CC = \int_0^{T_{nav}} \left \frac{1}{\dot{r}_{iv}(t)} \right dt$	$\frac{rad}{m}$
Forgásirány-váltás	ZC	$ZC = \sum_{k=1}^{fop * T_{nav}} \delta(\text{sgn}(\omega_r[k]) + \text{sgn}(\omega_r[k-1]))$	-
Laterális feszültség	LS	$LS = \int_0^{T_{nav}} \frac{ \mathbf{v}_{lin}(t) ^2}{r_{iv}(t)} dt$	Ns
Tangenciális feszültség	TS	$TS = \int_0^{T_{nav}} \mathbf{a}_{lin}(t) dt$	Ns
Kockázat	$RISK$	$RISK = \int_0^{T_{nav}} \frac{1}{\min(\mathbf{R}_{lid}(t))} dt$	$\frac{1}{m}$
Összesített rándulás	LJ	$LJ = \frac{1}{T_{nav}} * \int_0^{T_{nav}} \ddot{\mathbf{v}}_{lin}(t) ^2 dt$	$\left(\frac{m}{s^3}\right)^2$
Összesített rotációs rándulás	RJ	$RJ = \frac{1}{T_{nav}} * \int_0^{T_{nav}} \ddot{\omega}_r(t) ^2 dt$	$\left(\frac{rad}{s^3}\right)^2$
Futásidő	T_{ex}	Az algoritmus 1 iterációjának számítási ideje	[ms]

A táblázatban használt $\mathbf{a}_{lin} = [a_x; a_y]^T$ a robot lineáris gyorsulása, az $r_{iv}(t)$ paraméter pedig a robot pályájának pillanatnyi görbületi sugara, ami az alábbi módon számítható ki:

$$r_{curv}(t) = \frac{|\mathbf{v}_{lin}(t)|^2}{a_n(t)} \quad (110)$$

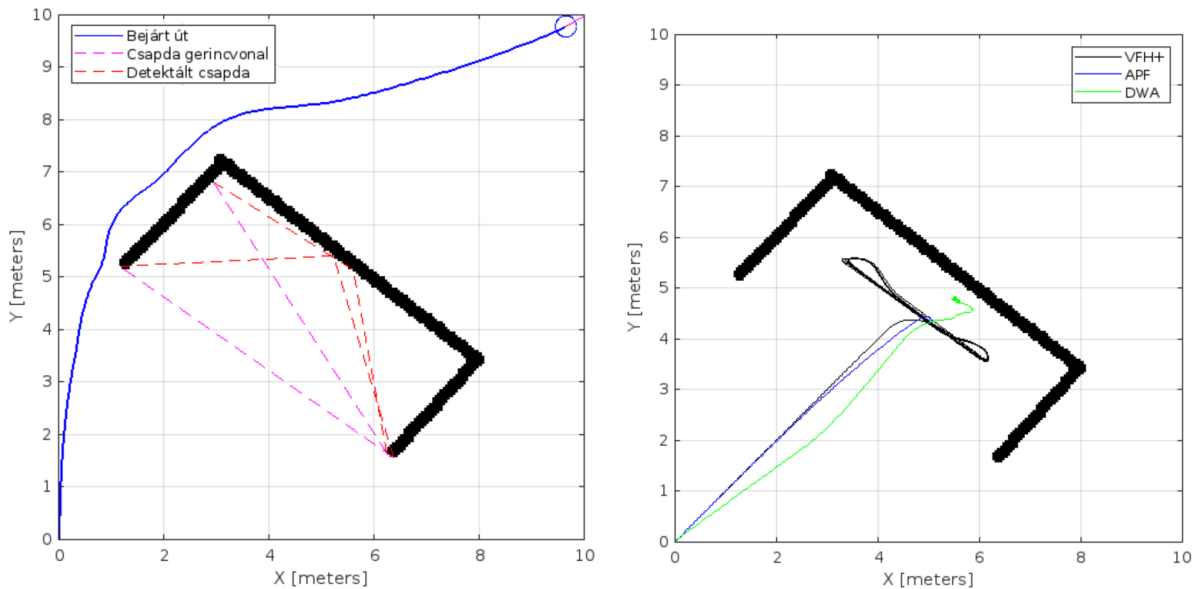
Ahol $a_n(t)$ a robot normál irányú gyorsulása, ami kiszámolható az alábbi módon:

$$a_n(t) = \sqrt{|\mathbf{a}_{lin}(t)|^2 - \left(\mathbf{a}_{lin}(t) \cdot \frac{\mathbf{v}_{lin}(t)}{|\mathbf{v}_{lin}(t)|} \right)^2} \quad (111)$$

3.3. Eredmények statikus környezetben

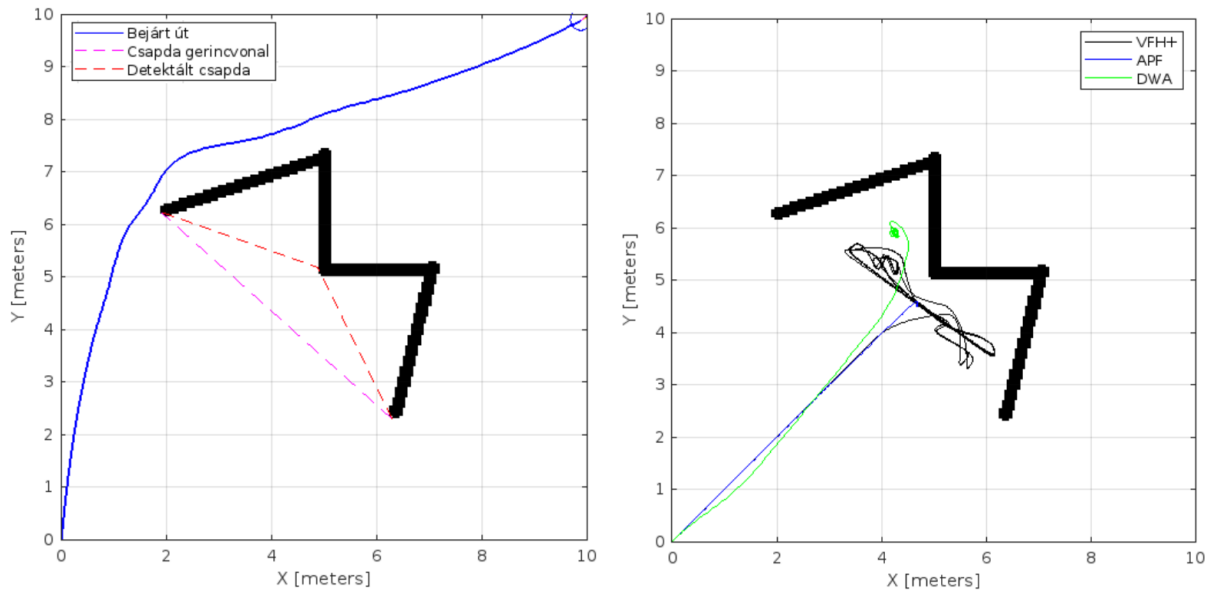
A statikus környezetben tesztelés elsősorban a tervezett algoritmus csapdahelyzetekkel szembeni ellenállóságát hivatott vizsgálni. Ehhez egyszerűbb típus-akadályokat tartalmazó 10x10m-es munkatérben, a [0;0]m pontból kiindulva a [10;10]m pontot célnak kitűzve futtatam a szimulációt. Az eredményeket bemutató ábrákon folytonos kék vonallal jelöltem a robot által bejárt utat és szaggatott vonalakkal a navigáció során detektált csapdahelyzeteket. A magenta színű szaggatott vonal jelöli a C_1C_2 gerincvonalat, ami egyben az adott csapdahelyzet nyitott oldala (ahol ki- és belépni lehet a csapdába). A piros színű szaggatott vonalak jelölik a C_1C_3 és C_2C_3 oldalait a csapdát jelképező háromszögnek, melyek nem átjárhatóak.

Az első statikus pálya a legegyszerűbb csapdahelyzetet, egy U-alakú akadályt tartalmaz. Itt a fejlesztett algoritmus mindössze egyetlen csapdahelyzet detektálása után képes elkerülni a lokális minimumot függetlenül az akadály méretétől. Az összehasonlításához használt tisztán lokális navigációs módszereknek azonban még az ilyen egyszerűbb csapdák is nehézséget okoznak, ahogy azt a 11. ábra illusztrálja. Itt megjegyzendő, hogy a VFH+ algoritmus általánosságban képes az U-alakú akadályok elkerülésére, viszont csak egy bizonyos méretig. Az itt bemutatott akadály mérete (~7m) azonban meghaladja ezt a korlátot, ezért esik csapdába ez a módszer is.



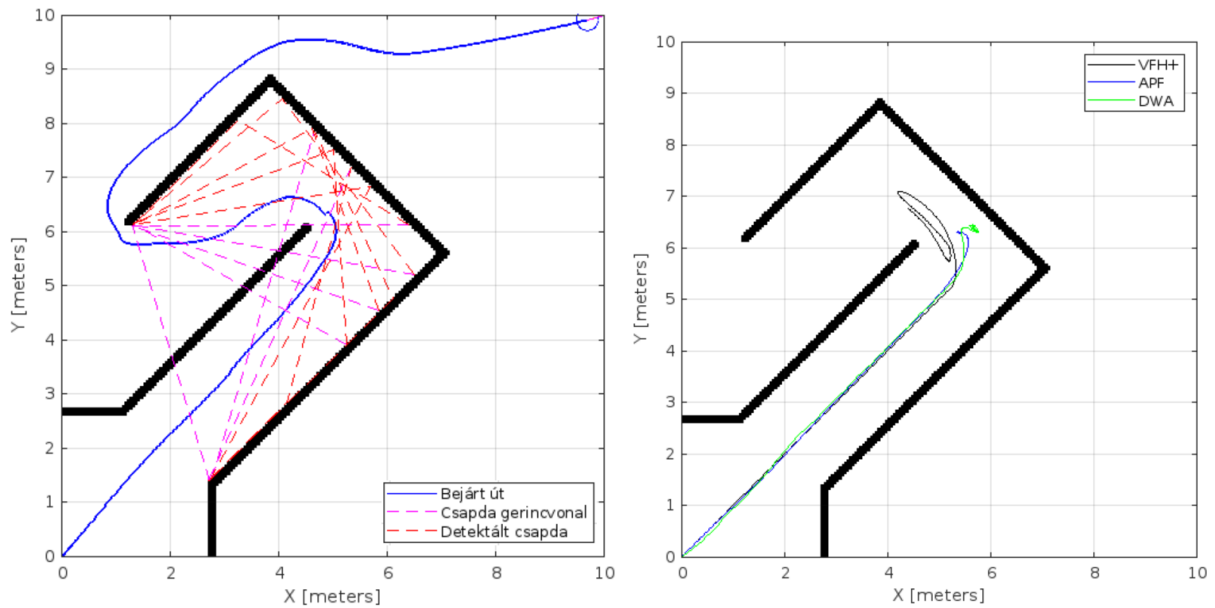
11. ábra: Statikus teszpálya U-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)

A második pálya egy W-alakú akadály, amely a látószög korlátok miatt egy komolyabb kihívást jelent. A fejlesztett algoritmus szintén egyetlen csapdahelyzet detektálása után gond nélkül elkerüli az akadályt, a többi módszer azonban a látószög korlátok miatt memóriefunkció hiányában képtelen kiszabadulni az oszcilláló mozgásból a W-alak két ága között, ahogy azt a 12. ábra mutatja.



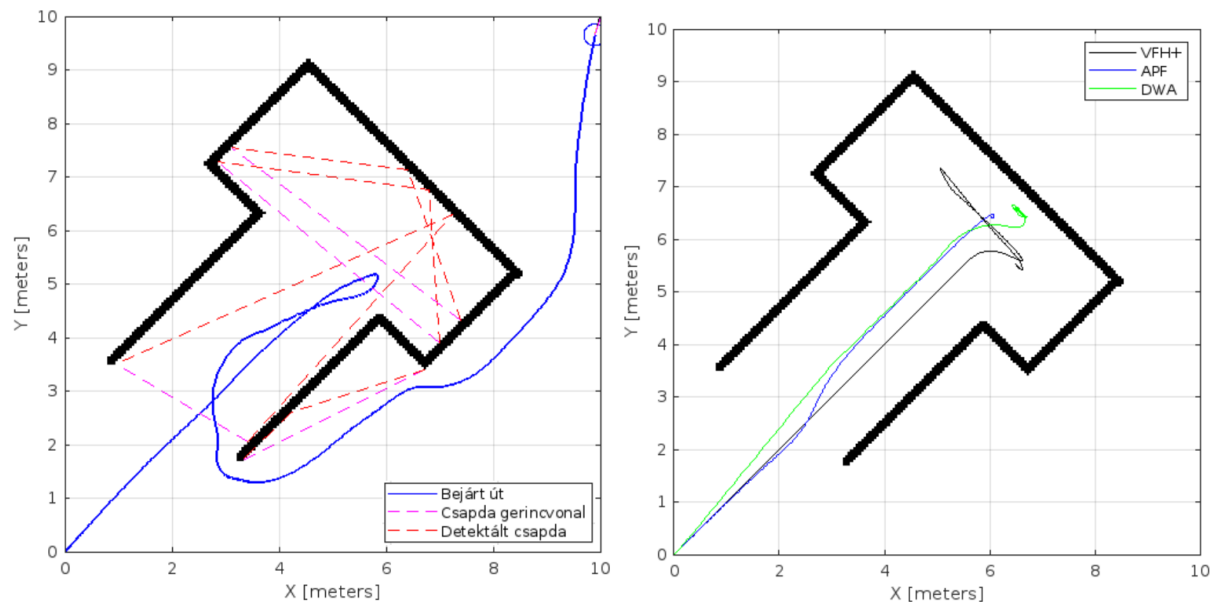
12. ábra: Statikus tesztpálya W-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)

A harmadik pályán egy L-alakú folyosó található, aminél a kihívás a fordulóban történő áthaladásban rejlik. Az akadály lényegében egy U-alak, ami a középső elválasztó fal miatt nem látható be ahogy a robot halad felé. Emiatt nem tudja az első tesztpályához hasonlóan elkerülni a csapdát, hanem kénytelen áthaladni rajta. A csapda csak a fordulóban detektálható, amikor a látószög korlátok megszűnnek és ekkor kell a robotnak megfelelő stratégiával megszökni a lokális minimumból. A fejlesztett algoritmus több csapdahelyzetet is detektál ahogy áthalad a fordulópontra, amelyek megakadályozzák, hogy rossz oldalon próbáljon meg távozni a kanyarból, ahogy azt a 13. ábra illusztrálja.



13. ábra: Statikus tesztpálya L-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)

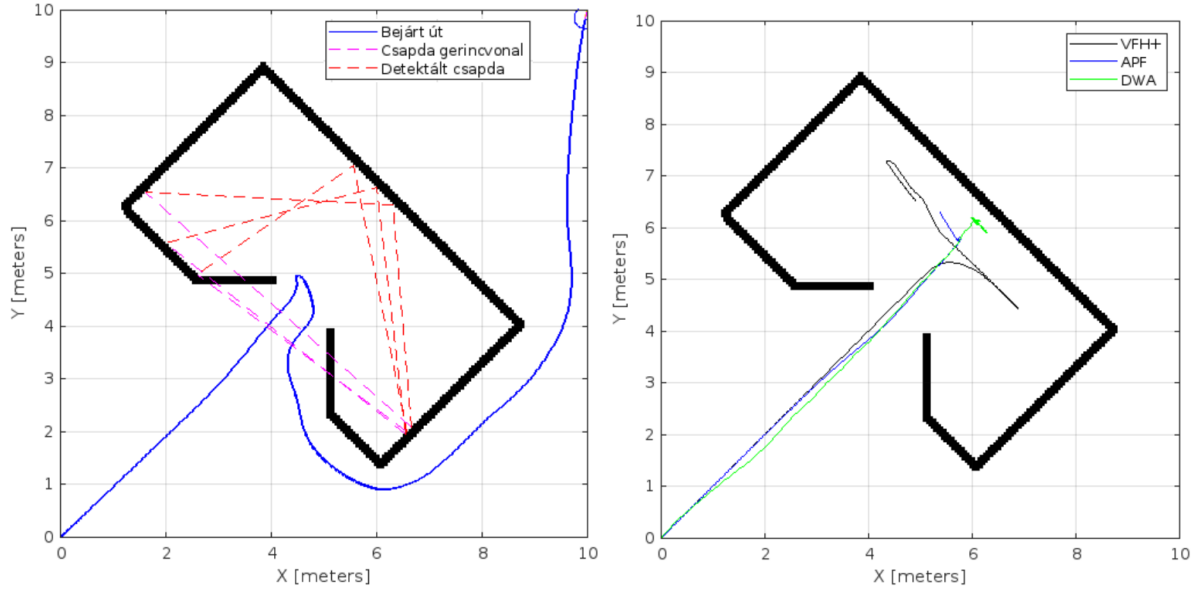
A negyedik pálya egy T-alakú akadály, melynek fő nehézsége hogy a csapdahelyzet csak azután ismerhető fel, hogy a robot már belépett az akadályba. Az akadály előtt állva a látószög korlátok miatt egy folyosó kereszteződésként jelenik meg az objektum, ahol mindkét oldalra el lehet haladni. Ennek következtében a navigációs algoritmusnak nem elegendő elkerülnie a csapdahelyzetet, hanem képesnek kell lennie elhagynia is azt.



14. ábra: Statikus tesztpálya T-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)

Az utolsó, ötödik statikus pályán egy üreges akadály található, melynek kihívásai a T-alakú akadályéhoz hasonlóak. A helyzet annyival nehezített, hogy az akadály bejárata jóval szűkebb,

mint a T-alakú akadályé volt, így a csapda felismerése után a megfordulást óvatosabban kell végezni.



15. ábra: Statikus tesztpálya W-alakú akadállyal: a fejlesztett algoritmus csapdaelkerülése (bal oldal) és az összehasonlított módszerek csapdába esése (jobb oldal)

A statikus tesztek során a fejlesztett navigációs algoritmus teljesítménymutatóit az alábbi táblázat tartalmazza:

3. Táblázat: A statikus tesztek teljesítménymutatói

Akadálytípus		U-alak	W-alak	L-alak	T-alak	Üreges
T_{nav}	[s]	36.9	35.4	98.1	73.3	79.8
$l_{út}$	[m]	16.07	15.85	23.53	25.99	22.58
φ_{tel}	[rad]	1.74	1.52	13.73	7.23	5.83
$AVSA$	[rad]	0.0703	0.0613	0.1161	0.1782	0.1143
CC	$\left[\frac{rad}{m}\right]$	15.77	12.07	402.4	94.95	164.77
LS	[Ns]	1.58	1.38	2.49	3.19	2.46
TS	[Ns]	2.05	1.89	4.25	5.23	3.66
$RISK$	$\left[\frac{1}{m}\right]$	27.97	24.98	153.97	96.16	111.89
ZC	[–]	7	7	18	9	117
LJ	$\left[\left(\frac{m}{s^3}\right)^2\right]$	0.04	0.03	0.17	0.22	0.07
RJ	$\left[\left(\frac{rad}{s^3}\right)^2\right]$	0.34	0.27	50.74	10.66	2.84

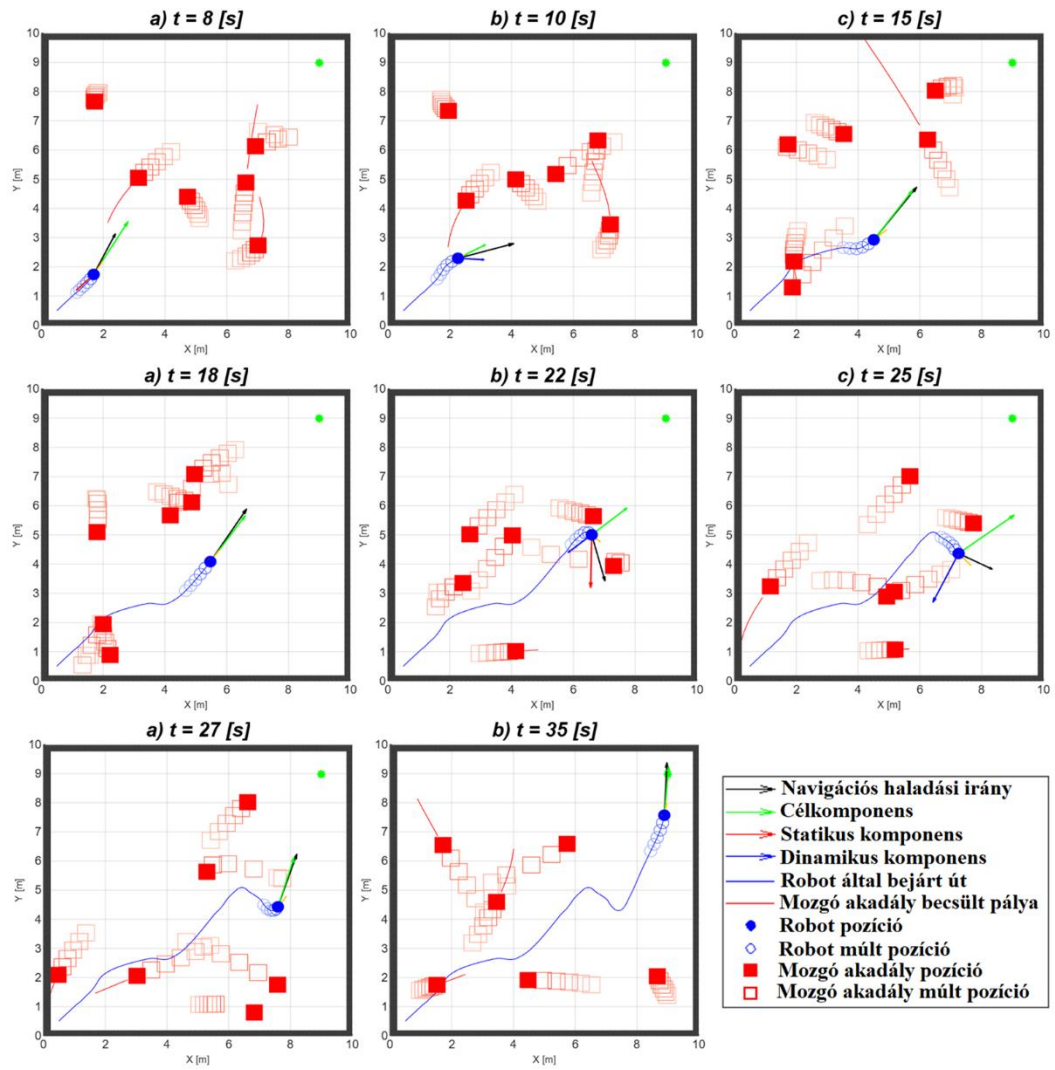
T_{ex}	[ms]	13.2	13.1	23.3	18.1	16.2
Státusz		SIK	SIK	SIK	SIK	SIK

A „Státusz” mutató értékei a következők lehetnek: SIK – amennyiben a robot sikeresen elérte a célt ütközések nélkül; CSAP – amennyiben a robot csapdába esett egy lokális minimumban a navigáció során és ÜTK – amennyiben a robot ütközött valamelyik akadállyal.

3.4. Eredmények dinamikus környezetben

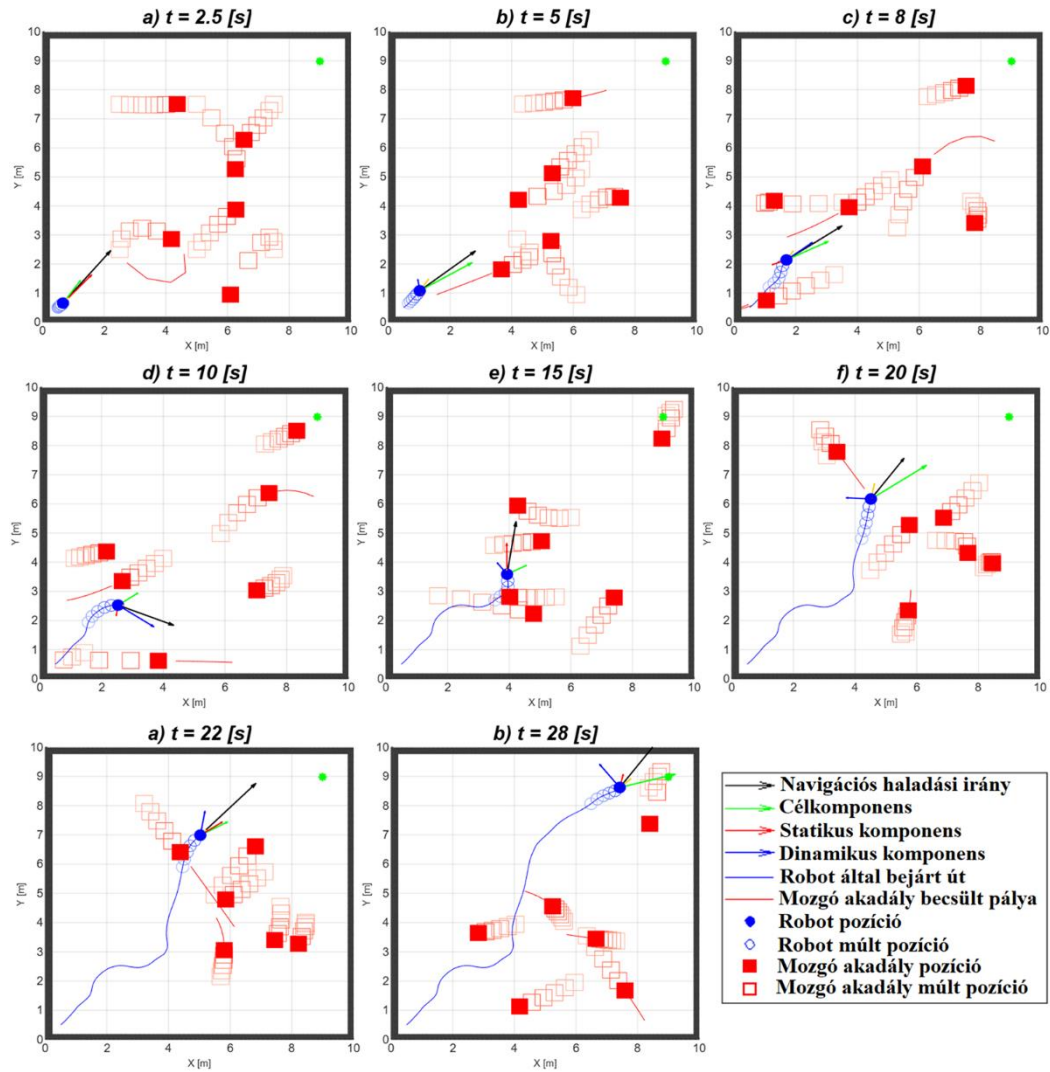
A dinamikus környezetben való tesztek célja az implementált prediktív mozgóakadályelkerülő módszer validációja. Ennek érdekében a 10x10m-es üres munkaterületbe 6 db véletlenszerűen generált pályán mozgó dinamikus akadályt helyeztem el, majd többször szimuláltam a robot viselkedését, ahogy megpróbál eljutni a [9;9]m pontba. Mivel ezekben a tesztekben nincsenek csapdahelyzetek, se egyéb statikus akadályok az összehasonlításhoz a tisztán lokális navigációs módszereket használtam (VFH+, APF, DWA).

Az első dinamikus teszt eredményét a 16. ábra mutatja. Mivel a mozgó akadályok pályái véletlenszerűen vannak generálva nem garantálható, hogy mindegyik kontaktba kerül a robottal, ahogy az végighalad a pályán. Viszont a pálya közepén található nagy sűrűségű mozgó tömegben való áthaladás így is igazolja, hogy a fejlesztett algoritmus képes egy bizonyos szinten kezelni a mozgó akadályok jelentette kihívásokat.



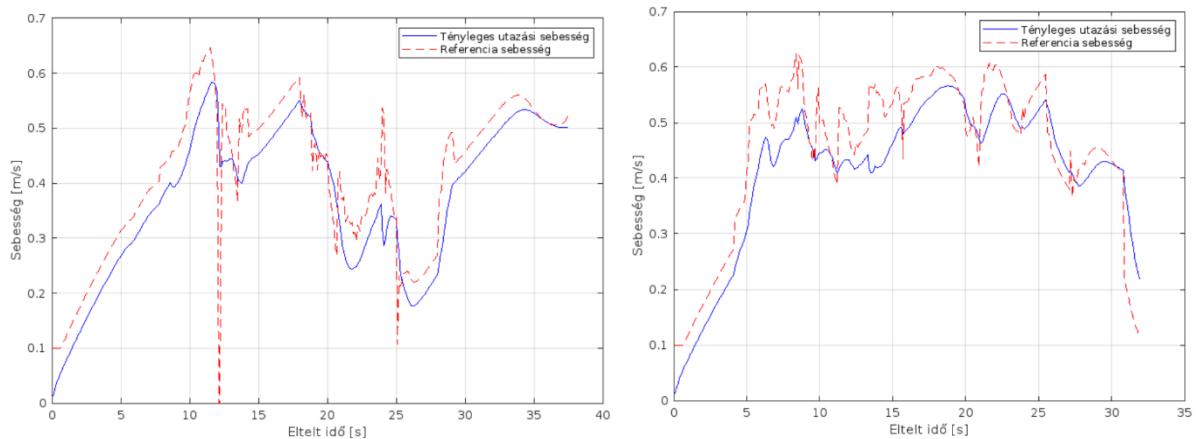
16. ábra: Első dinamikus teszt eredményei

A második dinamikus tesztet a 17. ábra illusztrálja. A környezet jellemzői megegyeznek az első tesztben használtakkal, csak az akadályok véletlenszerűen generált pályái okozzák az eltérést az eredményben.



17. ábra: A második dinamikus teszt eredményei

A két teszt során a robot sebesség-idő függvényét a 18. ábra mutatja. Látható, hogy a nagy számú mozgó akadály miatt sokszor gyors reakciók szükségesek az ütközések elkerüléséhez és az átlagos sebesség is viszonylag magas.



18. ábra: A robot sebesség-idő függvénye az első (balra) és a második (jobbra) dinamikus teszt során

Végül a két dinamikus környezetben futtatott teszt teljesítménymutatói az alábbi táblázatban láthatóak:

4. Táblázat: A dinamikus tesztek teljesítménymutatói és összehasonlítás a lokális navigációs módszerekkel

		1. Dinamikus környezet				2. Dinamikus környezet			
Algoritmus		Saját	VFH+	DWA	APF	Saját	VFH+	DWA	APF
T_{nav}	[s]	37.6	11.5	8.6	11.5	32	22.5	7.5	8.6
$l_{út}$	[m]	14.07	4.97	5.11	5.00	13.24	9.52	4.34	3.65
φ_{tel}	[rad]	3.03	0.15	0.14	0.18	2.04	0.62	0.17	0.15
AVSA	[rad]	0.178	0.069	0.098	0.048	0.178	0.103	0.141	0.082
CC	$\left[\frac{rad}{m}\right]$	91.35	9.13	8.30	5.26	111.0	29.09	9.19	5.83
LS	[Ns]	3.17	0.38	0.71	0.25	3.74	1.27	1.01	0.33
TS	[Ns]	4.29	0.95	1.29	0.83	4.59	2.13	1.55	0.91
RISK	$\left[\frac{1}{m}\right]$	44.2	12.87	8.99	12.65	47.69	35.42	8.80	9.92
ZC	[-]	11	24	13	2	14	27	14	4
LJ	$\left[\left(\frac{m}{s^3}\right)^2\right]$	1.15	0.84	5.39	0.37	0.93	0.58	3.20	0.55
RJ	$\left[\left(\frac{rad}{s^3}\right)^2\right]$	3.98	5.02	4.70	0.17	3.02	1.79	7.18	0.28
T_{ex}	[ms]	32.9	8.7	319.4	0.4	23.1	6.1	313.2	0.3
Státusz		SIK	ÜTK	ÜTK	ÜTK	SIK	ÜTK	ÜTK	ÜTK

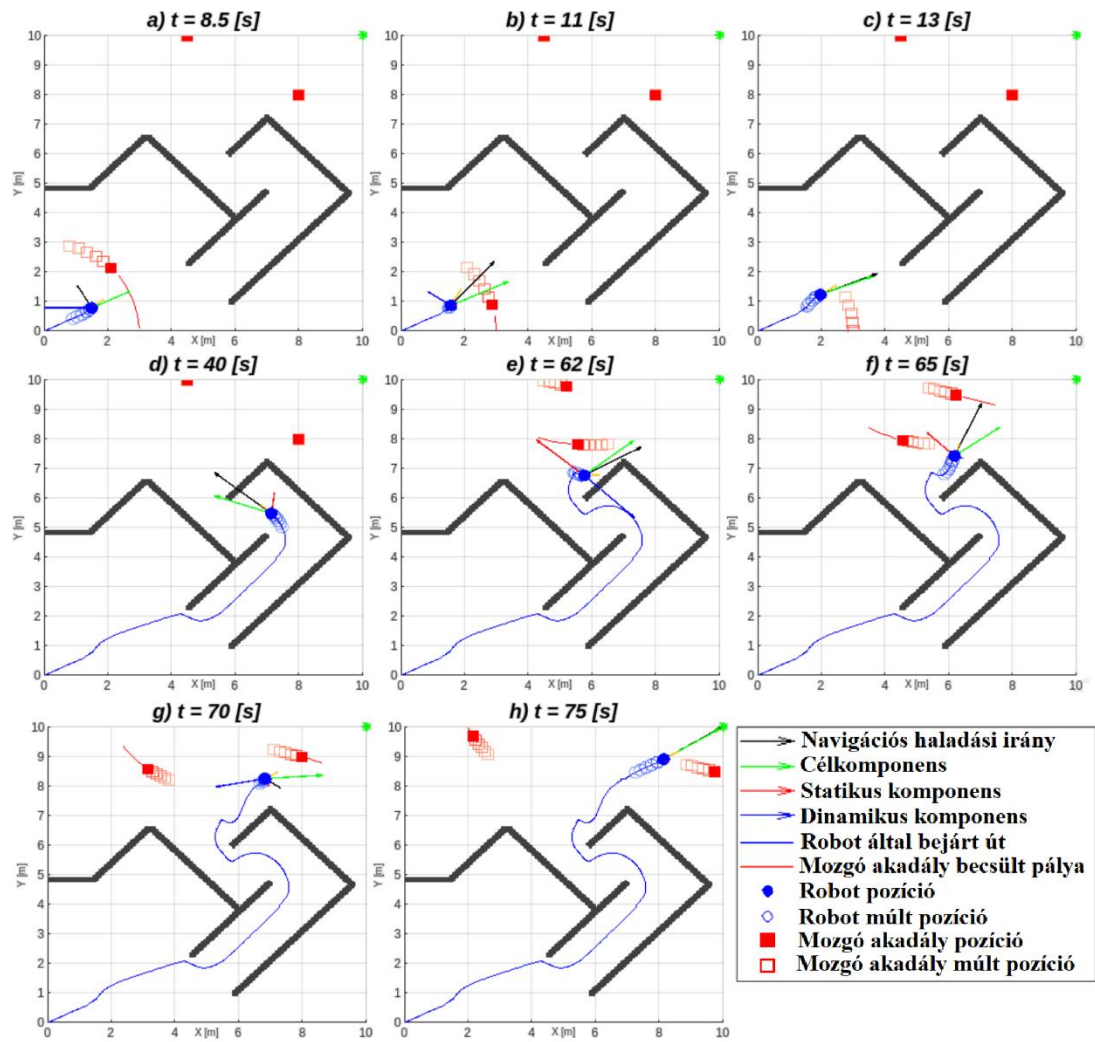
Ahogy az látható, a robot a saját algoritmust használva képes volt mindkét esetben ütközés nélkül elérni a célt, míg az összehasonlításhoz használt módszerek a prediktív funkciók hiányában mind ütköztek valamelyik mozgó akadállyal.

3.5. Eredmények komplex környezetben

A komplex környezet alatt jelen esetben az értendő, hogy egyaránt tartalmaz dinamikus és statikus akadályokat, valamint csapdahelyzeteket. Ez a készített navigációs algoritmus végső tesztelési környezete, ami a teljes funkcionalitást vizsgálja. Itt 3 db egyénileg készített komplex munkaterületen történő szimuláció eredményét mutatom be. A teszteken szereplő mozgó akadályok pályája itt már nem véletlenszerűen generált, hanem kézzel beállított annak érdekében, hogy garantálni lehessen a kontaktot a robot és az akadály között a futás során és meg lehessen vizsgálni a navigációs rendszer reakcióját.

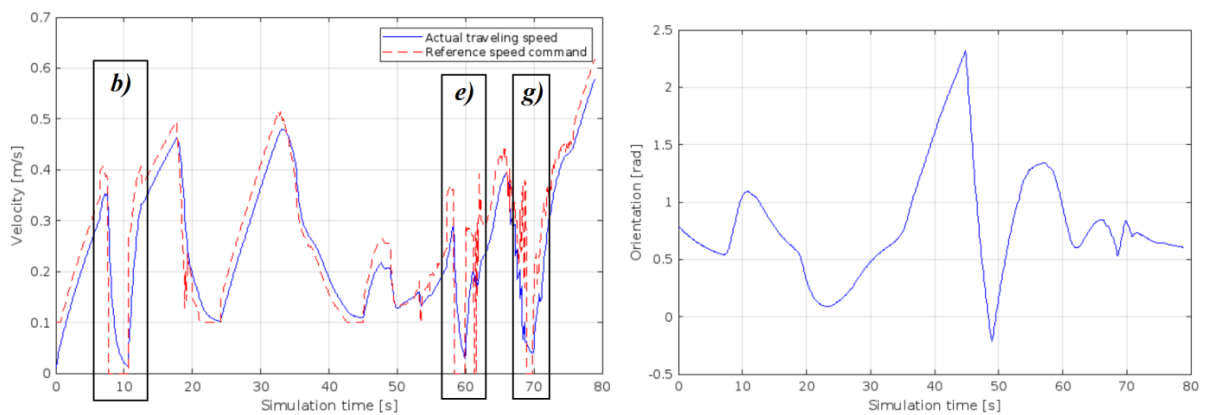
Az összehasonlításhoz a környezeteket DWA, VFH+ és hagyományos APF (kvadrátikus vonzó és exponenciális taszító erőfüggvényekkel) navigációs módszerekkel is teszteltem. Mivel az összehasonlításhoz használt módszerek az akadálypálya elején lokális minimumokba esnek, mindegyikük esetében egy második tesztet is elvégeztem kétszintű navigációs módszerrel, hogy a fejlesztett algoritmussal összehasonlítható értékelési mérőszámokat kapjak. Az összehasonlításhoz használt kétszintű algoritmusok: VFH-RRT*, APF-RRT és DWA-PRM. A szimulációs eredmény bemutatásának formátuma a környezetről készült képek sorozata az egyes forgatókönyvek esetében a dinamikus akadályok mozgásának jelzésére. Minden képet ábécérendben számoztam a megfelelő szimulációs idővel, amelyet t -vel jelöltem.

Az első tesztpályán a kezdeti mozgó akadályt megállás segítségével kerüli el a robot, ahogy az a 19. ábra b) részén látható. Ezt követően egy „U” alakú csapdát kerül el, mivel felismeri azt zsákutcaként (19. ábra c) részén a célkomponens iránya mutatja). Ezután egy „L” alakú folyosón halad át (19. ábra e) blokk) anélkül, hogy elakadna a fordulópontban a 19. ábra d) részén. Az utolsó szakaszban két dinamikus akadály egyidejű kikerülése történik a nyílt terepen (19. ábra f-h) részei) a pályabecslő algoritmus és a dinamikus komponens segítségével. A megállási mód mindkét esetben aktiválódik, de félbemarad mielőtt a robot teljesen lelassult volna, mivel az akadályok eltávolodtak.



19. ábra: A tervezett navigációs algoritmus az első komplex tesztkörnyezetben

Az első tesztpályán a robot sebesség-idő valamint orientáció-idő függvényét a 20. ábra mutatja. A sebesség-idő grafikonon kiemelt részeinek betűzése megegyezik a 19. ábra hozzájuk tartozó blokkjával.



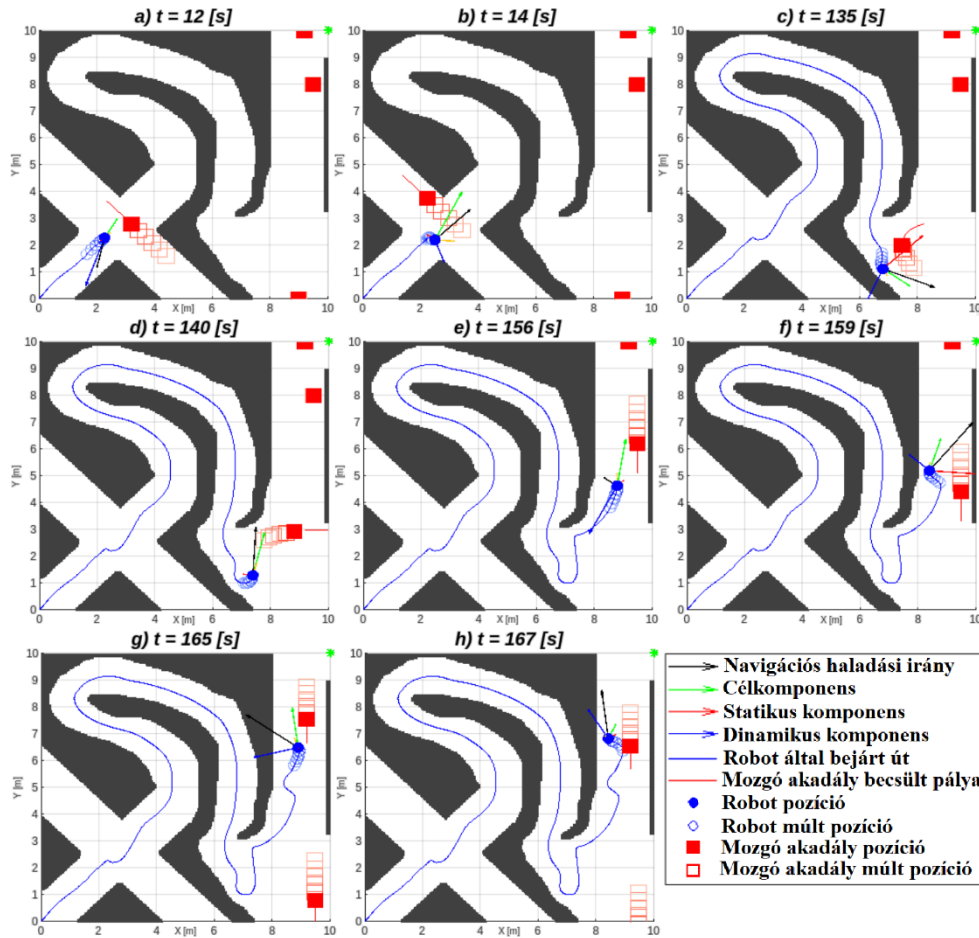
20. ábra: Sebesség-idő (balra) és orientáció-idő (jobbra) karakterisztikák az első komplex tesztpályán

Más módszerekkel összehasonlítva: a tisztán lokális módszerek mindegyike csapdába esett az „U” alakú akadályban. A teszt során a használt módszerek teljesítménymutatóit az alábbi táblázat tartalmazza:

5. Táblázat: Az első komplex teszt teljesítménymutatói és összehasonlítás kétszintű navigációs módszerekkel

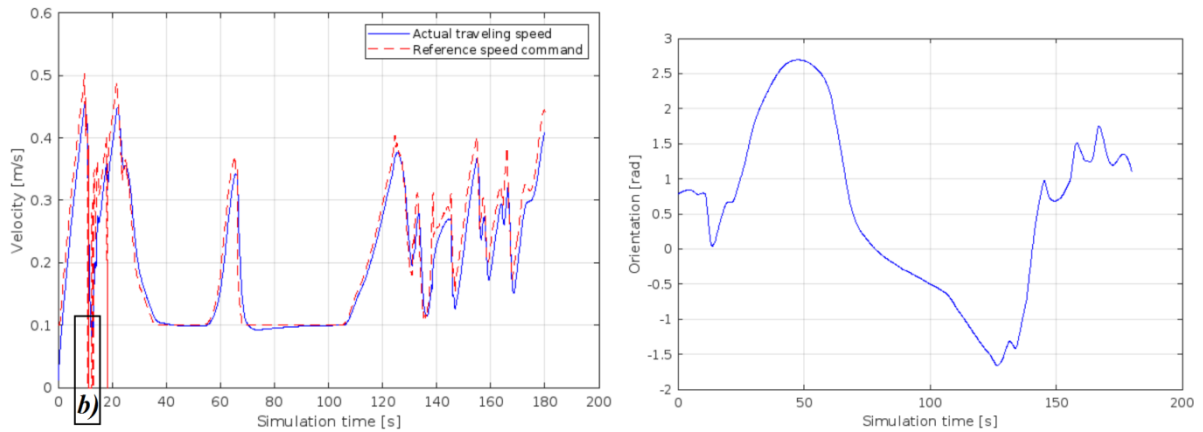
Algoritmus		Saját	VFH+	VFH-RRT*	APF	APF-RRT	DWA	DWA-PRM
T_{nav}	[s]	79	31.4	58.1	22.2	60.1	20.2	67.2
$l_{út}$	[m]	19.27	8.87	18.63	5.66	19.57	8.73	20.11
φ_{tel}	[rad]	7.52	4.31	7.06	1.92	7.70	2.64	7.53
AVSA	[rad]	0.223	0.229	0.207	0.237	0.169	0.210	0.246
CC	$\left[\frac{rad}{m}\right]$	1192	7220	3.25e4	1.33e5	1.27e4	8212	3003
LS	[Ns]	3.43	1.94	4.46	2.11	3.93	3.89	5.25
TS	[Ns]	6.47	3.50	6.03	4.09	5.15	7.78	8.18
RISK	$\left[\frac{1}{m}\right]$	116.0	27.1	102.1	13.85	130.5	34.95	116.7
ZC	[–]	18	19	8	35	6	20	23
LJ	$\left[\left(\frac{m}{s^3}\right)^2\right]$	0.88	0.93	0.55	8.02	0.30	21.29	1.39
RJ	$\left[\left(\frac{rad}{s^3}\right)^2\right]$	37.16	14.53	6.50	379.2	2.71	114.9	12.89
T_{ex}	[ms]	25.2	6.5	1237	0.2	1175	256	1611
Státusz		SIK	CSAP	SIK	CSAP	SIK	CSAP	SIK

A második tesztpálya elején egy keresztfolyosón áthaladó dinamikus akadályt a pályabecslő algoritmusnak köszönhetően elkerüli a robot (21. ábra a-b) része). Itt ismét a megállási üzemmódba lépünk, azonban az akadály viszonylag gyors mozgása ($\approx 0,68[m/s]$ sebességgel halad) lehetővé teszi, hogy a robot folytassa az útját, mielőtt teljesen meg tudna állni. Ezután egy szűk, íves folyosón halad át anélkül, hogy lokális minimumokban csapdába esne. A folyosó végére érve egy dinamikus akadályt úgy kerül ki, hogy átengedi maga előtt (21. ábra c-d) blokk). Ezután két következő, szembejövő mozgó akadályt (az első a 21. ábra e-f) részén a második a 21. ábra g-h) részén) egy egyenes folyosón kerül el. Bár mindkét akadály viszonylag gyors ($0,6[m/s]$ és $\approx 0,5[m/s]$), a robot a pályaelőrejelzés miatt rendelkezésre álló hosszabb reakcióidőnek köszönhetően mindkettőt el tudja kerülni a folyosó oldalára való elmozdulással.



21. ábra: A tervezett navigációs algoritmus a második komplex tesztkörnyezetben

A második komplex tesztkörnyezet sebesség-idő és orientáció-idő függvényeit a 22. ábra illusztrálja. A sebesség-idő grafikon b) része a megálló viselkedést mutatja be.



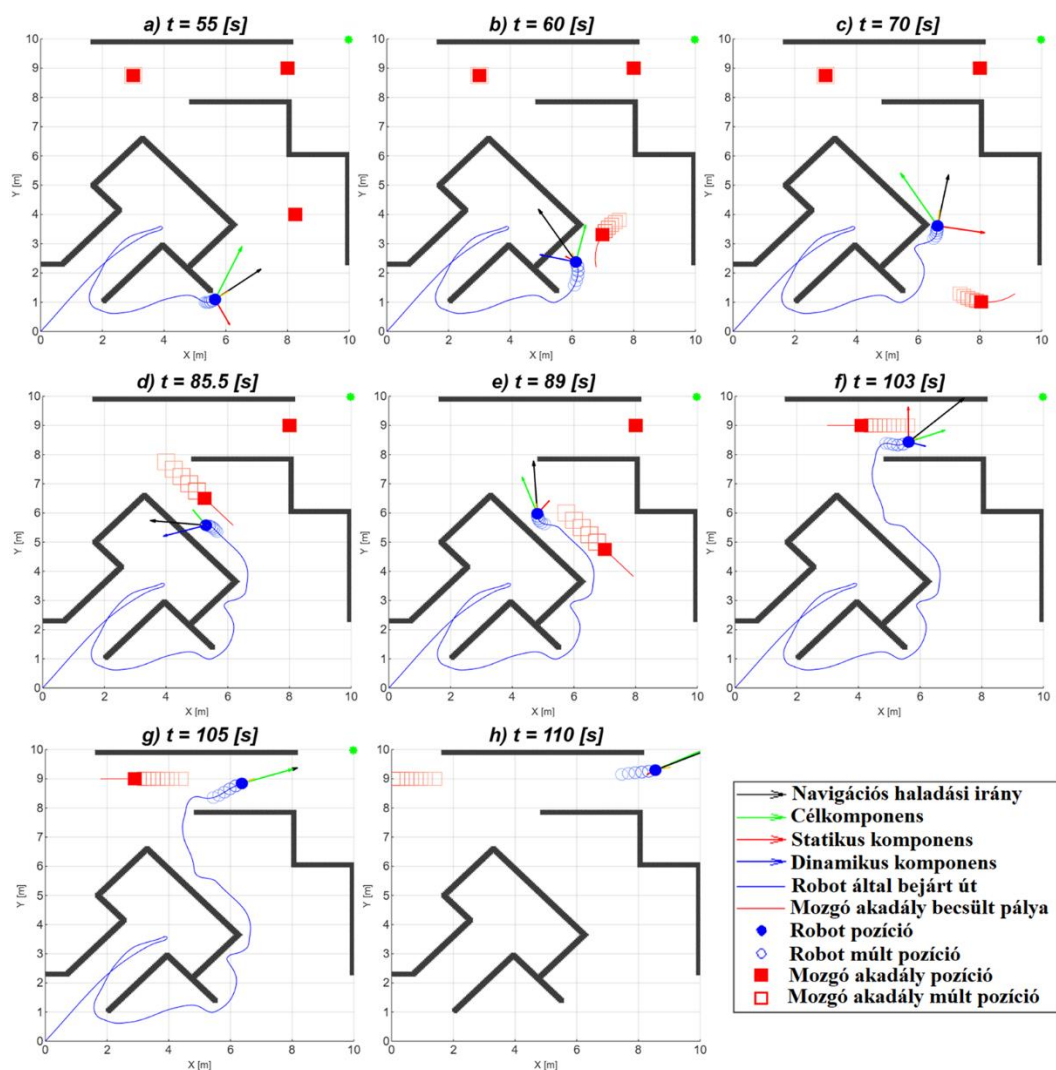
22. ábra: Sebesség-idő (balra) és orientáció-idő (jobbra) karakterisztikák a második komplex teszpályán

Ezen a pályán mindhárom összehasonlított tisztán lokális módszer a kanyargós folyosó különböző szakaszaiban reked. A teszt során a használt módszerek teljesítménymutatóit az alábbi táblázat tartalmazza:

6. Táblázat: A második komplex teszt teljesítménymutatói és összehasonlítás kétszintű navigációs módszerekkel

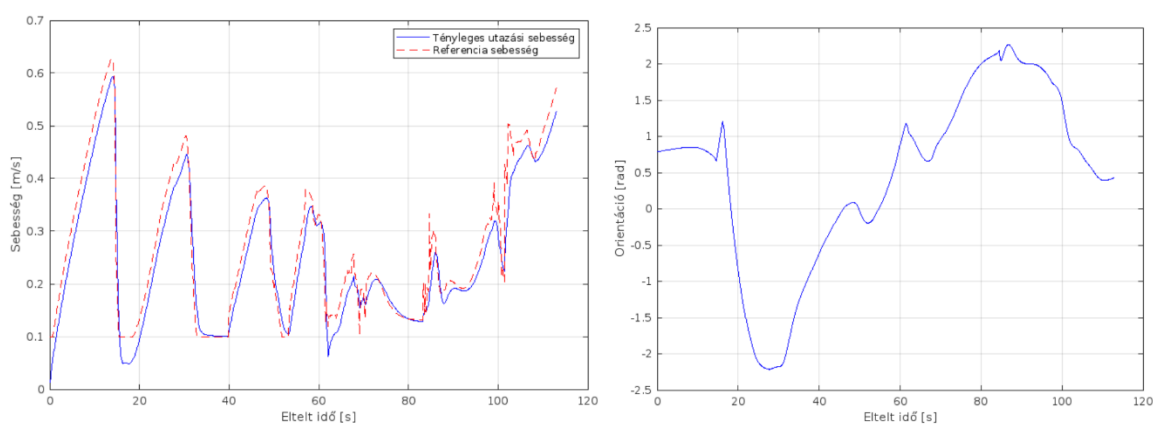
Algoritmus		Saját	VFH+	VFH-RRT*	APF	APF-RRT	DWA	DWA-PRM
T_{nav}	[s]	180	135.9	110.1	53.6	104.1	24.8	152.2
$l_{út}$	[m]	35.98	33.55	29.42	8.98	28.27	10.24	31.34
φ_{tel}	[rad]	13.02	15.64	10.29	5.16	9.72	2.54	17.96
$AVSA$	[rad]	0.149	0.085	0.126	0.220	0.104	0.201	0.217
CC	$\left[\frac{rad}{m}\right]$	422.6	3990	2566	5.86e5	112.6	5.06e4	5.59e4
LS	[Ns]	4.39	3.35	4.19	5.22	4.06	3.56	9.81
TS	[Ns]	7.73	5.42	5.43	9.25	4.70	9.55	15.54
$RISK$	$\left[\frac{1}{m}\right]$	314.7	281.3	266.4	86.7	245.7	60.7	392.2
ZC	[-]	22	40	10	77	8	29	77
LJ	$\left[\left(\frac{m}{s^3}\right)^2\right]$	0.17	0.26	0.23	8.79	0.12	11.56	0.92
RJ	$\left[\left(\frac{rad}{s^3}\right)^2\right]$	3.31	5.61	4.14	503.4	1.63	30.86	28.05
T_{ex}	[ms]	44.8	6.5	1516	0.2	1517	256	1741
Státusz		SIK	CSAP	SIK	CSAP	SIK	CSAP	SIK

A harmadik tesztpályán először egy „T” alakú csapda kerülhető el a memória hatásának köszönhetően. Mivel a csapda az akadály elejéről nem észlelhető (a látószög miatt folyosó keresztmetszetként jelenik meg, amelynek mindkét végén a rá merőleges szakaszon átjáró van), a robot először belép, de amikor a látószög korlát a megszűnik, észleli a csapdát majd kivezeti a robotot belőle. Ezután egy dinamikus tárgyat úgy kerül el, hogy letér a megjósolt pályáról és hagyja elhaladni (23. ábra b-c) blokk). A következő W alakú akadályt a csapdaként való előre felismerés miatt kerül ki (a 23. ábra d) részén látható). Ezzel egyidejűleg egy másik dinamikus akadályt is kikerül azáltal, hogy ideiglenesen letér a megjósolt pályától (23. ábra d-e) részén). Az utolsó dinamikus akadályt is így kerüli meg a navigációs célhoz vezető folyosón (23. ábra f-g) blokk).



23. ábra: A tervezett navigációs algoritmus a harmadik komplex tesztkörnyezetben

A második komplex tesztkörnyezet sebesség-idő és orientáció-idő függvényeit a 24. ábra mutatja.



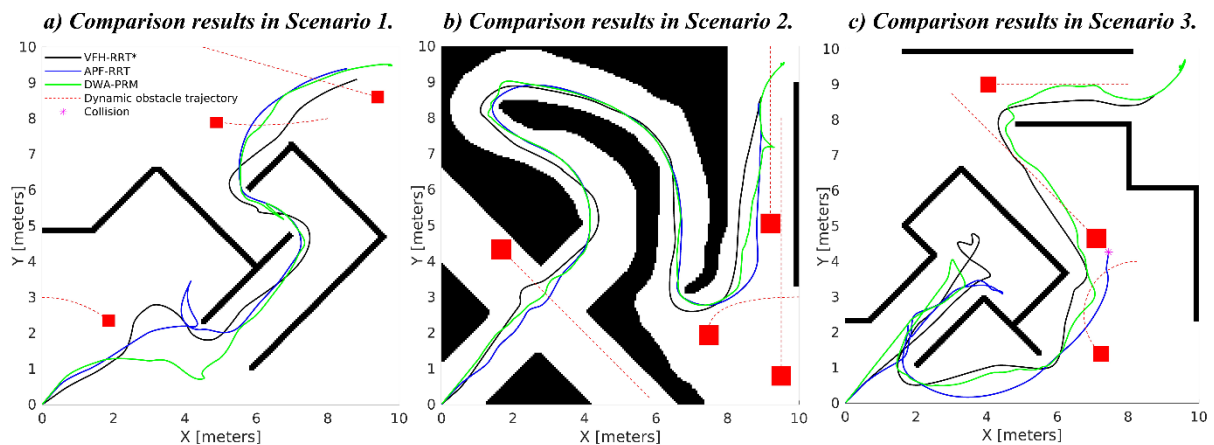
24. ábra: Sebesség-idő (balra) és orientáció-idő (jobbra) karakterisztikák a harmadik komplex testpályán

Ebben a forgatókönyvben az APF-RRT módszer ütközött a viszonylag gyors (0,5[m/s]) második dinamikus akadállyal, mivel a pálya előrejelzése nélkül nem volt ideje a kikerülő manőver végrehajtására. Továbbá a globális tervezők segítségével nélkül az összes összehasonlított tisztán lokális módszer csapdába esett a „T” alakú akadályon belül.

7. Táblázat: A harmadik komplex teszt teljesítménymutatói és összehasonlítás kétszintű navigációs módszerekkel

Algoritmus		Saját	VFH+	VFH-RRT*	APF	APF-RRT	DWA	DWA-PRM
T_{nav}	[s]	113.1	34.7	96.1	39.8	61.3	11.2	103.4
$l_{út}$	[m]	27.96	8.77	29.55	7.17	19.43	6.82	29.15
φ_{tel}	[rad]	12.49	3.85	12.32	3.78	9.71	0.34	14.13
$AVSA$	[rad]	0.177	0.119	0.177	0.245	0.195	0.126	0.242
CC	$\left[\frac{rad}{m}\right]$	306.1	2651	375.6	4.59e5	775.6	8.50	1572
LS	[Ns]	4.96	1.09	6.26	3.72	4.49	0.95	9.99
TS	[Ns]	7.23	2.45	8.05	7.14	6.26	1.72	16.31
$RISK$	$\left[\frac{1}{m}\right]$	216.5	44.2	165.0	57.8	116.7	13.2	187.75
ZC	[-]	13	47	11	64	11	6	37
LJ	$\left[\left(\frac{m}{s^3}\right)^2\right]$	0.16	0.57	0.42	7.70	0.47	1.92	1.02
RJ	$\left[\left(\frac{rad}{s^3}\right)^2\right]$	18.78	13.63	5.32	344.3	21.90	17.98	9.69
T_{ex}	[ms]	33.3	6.5	1349	0.2	1282	256	1514
Státusz		SIK	CSAP	SIK	CSAP	ÜTK	CSAP	SIK

A három komplex tesztpályán az összehasonlításul használt módszerek által bejárt pályáját a 25. ábra mutatja. Mivel az egyszerű lokális algoritmusok (APF, VFH+, DWA) mindegyike a pálya elején csapdába esik, itt csak a kétszintes navigációs módszerek, saját algoritmussal összevethető eredményeit ábrázolom.



25. ábra: Az összehasonlításul használt kétszintű navigációs algoritmusok által bejárt pálya a komplex tesztkörnyezetekben

4. Következtetések és javaslatok

A bemutatott eredmények alapján kijelenthető, hogy a fejlesztett navigációs algoritmus képes széles körű csapdahelyzetek elkerülésére, dinamikus környezetben való navigációra és olyan előre nem ismert komplex környezetben működésre, ahol mind statikus mind mozgó akadályok megtalálhatóak. Az algoritmus az összes teszt pályán sikeresen elérte a célt, míg az összehasonlításához használt módszerek különböző nehézségekbe ütköztek. A tisztán statikus algoritmusok szinte minden esetben csapdába kerültek, valamint a prediktív viselkedés hiányában időnként mind a lokális mind a kétszintű algoritmusok ütköztek mozgó akadályokkal. A teljesítménymutatókat megvizsgálva észrevehetőek bizonyos tendenciák: a nagyobb sebességgel mozgás (alacsonyabb navigációs idő) magasabb igénybevételnek teszi ki a robotot (LS , TS , LJ és RJ mutatók), valamint az agresszívabb kormányzás ($AVSA$ mutató) és nagyobb oszcilláció (elsősorban a ZC indikátor mutatja) pedig a rotációs igénybevételt növeli (TS , RJ). Ezen észrevételek mellett kijelenthető, hogy az összehasonlításához használt kétszintű navigációs algoritmusok sok esetben hasonló teljesítményt értek el a saját fejlesztésű algoritmussal, azonban ezt egy jelentősen nagyobb számítási kapacitás (lassabb döntéshozás) árán érték el (ugyanazon feltételek mellett). Az általam készített algoritmus egy iterációjának futásideje átlagosan 2.80%-a az összehasonlításban használt módszerek futásidejének, amivel lényegesen alkalmasabb a valós idejű döntéshozásra. Ennek oka, hogy a kétszintű algoritmusok feltérképezés komponense és a globális tervező egy jelentős számítási többletet okoz, amit az általam készített algoritmus optimalizált információfeldolgozása kiküszöböl. További előny, hogy a kétszintű algoritmusokkal ellentétben figyelembe veszi a mozgó akadályok becsült

közeljövőbeli pályáját a döntéshozáskor ezzel képes elkerülni olyan ütközéseket, melyet azok nem.

Megjegyzés

A dokumentumban bemutatott kutatási téma az EKÖP-1 pályázatához kapcsolódik, mely eredményei az alábbi [31] nemzetközi (C1, IF: 8.6) folyóiratban szintén megjelenésre felülvizsgálat alatt vannak (2024.11.11.).

„A KULTURÁLIS ÉS INNOVÁCIÓS MINISZTERIUM EKÖP-24-1 KÓDSZÁMÚ EGYETEMI KUTATÓI ÖSZTÖNDÍJ PROGRAMJÁNAK A NEMZETI KUTATÁSI, FEJLESZTÉSI ÉS INNOVÁCIÓS ALAPBÓL FINANSZÍROZOTT SZAKMAI TÁMOGATÁSÁVAL KÉSZÜLT.”

Irodalomjegyzék

- [1] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, ‘A Systematic Literature Review of A* Pathfinding’, *Procedia Comput. Sci.*, vol. 179, pp. 507–514, Jan. 2021, doi: 10.1016/j.procs.2021.01.034.
- [2] S. LaValle, ‘Rapidly-exploring random trees : a new tool for path planning’, *Annu. Res. Rep.*, 1998, Accessed: Mar. 01, 2024. [Online]. Available: <https://www.semanticscholar.org/paper/Rapidly-exploring-random-trees-%3A-a-new-tool-for-LaValle/d967d9550f831a8b3f5cb00f8835a4c866da60ad>
- [3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, ‘Probabilistic roadmaps for path planning in high-dimensional configuration spaces’, *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, Art. no. 4, 1996, doi: 10.1109/70.508439.
- [4] G. Wang, C. Jiang, G. Tao, and C. Ye, ‘Dynamic path planning based on the fusion of improved RRT and DWA algorithms’, in *2023 4th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM)*, 2023, pp. 534–538. doi: 10.1109/ICMTIM58873.2023.10246738.
- [5] O. Khatib, ‘Real-time obstacle avoidance for manipulators and mobile robots’, in *1985 IEEE International Conference on Robotics and Automation Proceedings*, Mar. 1985, pp. 500–505. doi: 10.1109/ROBOT.1985.1087247.
- [6] Y. Koren and J. Borenstein, ‘Potential field methods and their inherent limitations for mobile robot navigation’, in *1991 IEEE International Conference on Robotics and Automation Proceedings*, Apr. 1991, pp. 1398–1404 vol.2. doi: 10.1109/ROBOT.1991.131810.
- [7] Z. Wu, J. Dai, B. Jiang, and H. R. Karimi, ‘Robot path planning based on artificial potential field with deterministic annealing’, *ISA Trans.*, vol. 138, pp. 74–87, Jul. 2023, doi: 10.1016/j.isatra.2023.02.018.
- [8] J. Amiryan and M. Jamzad, ‘Adaptive Motion Planning with Artificial Potential Fields Using a Prior Path’, arXiv.org. Accessed: Mar. 01, 2024. [Online]. Available: <https://arxiv.org/abs/2005.04191v1>

- [9] X. Fan, Y. Guo, H. Liu, B. Wei, and W. Lyu, ‘Improved Artificial Potential Field Method Applied for AUV Path Planning’, *Math. Probl. Eng.*, vol. 2020, p. e6523158, Apr. 2020, doi: 10.1155/2020/6523158.
- [10] Y. H. Kim, W. S. Son, J. B. Park, and T. S. Yoon, ‘Smooth Path Planning by Fusion of Artificial Potential Field Method and Collision Cone Approach: 1st International Conference on Measurement Instrumentation and Electronics, ICMIE 2016’, *MATEC Web Conf.*, vol. 75, no. 05004, Art. no. 05004, Sep. 2016, doi: 10.1051/mateconf/20167505004.
- [11] B. Kovács, G. Szayer, F. Tajti, M. Burdelis, and P. Korondi, ‘A novel potential field method for path planning of mobile robots by adapting animal motion attributes’, *Robot. Auton. Syst.*, vol. 82, pp. 24–34, Aug. 2016, doi: 10.1016/j.robot.2016.04.007.
- [12] J. Borenstein and Y. Koren, ‘The vector field histogram-fast obstacle avoidance for mobile robots’, *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, Art. no. 3, Jun. 1991, doi: 10.1109/70.88137.
- [13] I. Ulrich and J. Borenstein, ‘VFH+: reliable obstacle avoidance for fast mobile robots’, in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, 1998, pp. 1572–1577 vol.2. doi: 10.1109/ROBOT.1998.677362.
- [14] D. Díaz and L. Marín, ‘VFH+D: An Improvement on the VFH+ Algorithm for Dynamic Obstacle Avoidance and Local Planning*’, *IFAC-Pap.*, vol. 53, no. 2, Art. no. 2, Jan. 2020, doi: 10.1016/j.ifacol.2020.12.2450.
- [15] I. Ulrich and J. Borenstein, ‘VFH/sup */: local obstacle avoidance with look-ahead verification’, in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Apr. 2000, pp. 2505–2511 vol.3. doi: 10.1109/ROBOT.2000.846405.
- [16] D. Fox, W. Burgard, and S. Thrun, ‘The dynamic window approach to collision avoidance’, *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, Art. no. 1, Mar. 1997, doi: 10.1109/100.580977.
- [17] W. Yang *et al.*, ‘Improved Artificial Potential Field and Dynamic Window Method for Amphibious Robot Fish Path Planning’, *Appl. Sci.*, vol. 11, no. 5, Art. no. 5, Jan. 2021, doi: 10.3390/app11052114.
- [18] Y. Cao and N. Mohamad Nor, ‘An improved dynamic window approach algorithm for dynamic obstacle avoidance in mobile robot formation’, *Decis. Anal. J.*, vol. 11, p. 100471, Jun. 2024, doi: 10.1016/j.dajour.2024.100471.
- [19] Z. Lin and R. Taguchi, ‘Faster Implementation of The Dynamic Window Approach Based on Non-Discrete Path Representation’, *Mathematics*, vol. 11, no. 21, Art. no. 21, Jan. 2023, doi: 10.3390/math11214424.
- [20] J. Minguez and L. Montano, ‘Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios’, *IEEE Trans. Robot. Autom.*, vol. 20, no. 1, Art. no. 1, 2004, doi: 10.1109/TRA.2003.820849.
- [21] J. W. Durham and F. Bullo, ‘Smooth Nearness-Diagram Navigation’, *2008 IEEE RSJ Int. Conf. Intell. Robots Syst.*, pp. 690–695, Sep. 2008, doi: 10.1109/IROS.2008.4651071.
- [22] V. Sezer and M. Gokasan, ‘A novel obstacle avoidance algorithm: “Follow the Gap Method”’, *Robot. Auton. Syst.*, vol. 60, no. 9, Art. no. 9, Sep. 2012, doi: 10.1016/j.robot.2012.05.021.
- [23] M. Mujahed, D. Fischer, and B. Mertsching, ‘Tangential Gap Flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments’, *Robot. Auton. Syst.*, vol. 84, pp. 15–30, Oct. 2016, doi: 10.1016/j.robot.2016.07.001.
- [24] M. Mujahed, D. Fischer, and B. Mertsching, ‘Admissible gap navigation: A new collision avoidance approach’, *Robot. Auton. Syst.*, vol. 103, pp. 93–110, May 2018, doi: 10.1016/j.robot.2018.02.008.

- [25] H. Houshyari and V. Sezer, ‘A new gap-based obstacle avoidance approach: follow the obstacle circle method’, *Robotica*, vol. 40, pp. 1–24, Nov. 2021, doi: 10.1017/S0263574721001624.
- [26] K. Yan and B. Ma, ‘Mapless Navigation Based on 2D LIDAR in Complex Unknown Environments’, *Sensors*, vol. 20, no. 20, Art. no. 20, Oct. 2020, doi: 10.3390/s20205802.
- [27] T. Elmokadem and A. V. Savkin, ‘A Hybrid Approach for Autonomous Collision-Free UAV Navigation in 3D Partially Unknown Dynamic Environments’, *Drones*, vol. 5, no. 3, Art. no. 3, Sep. 2021, doi: 10.3390/drones5030057.
- [28] J. Fan, X. Chen, Y. Wang, and X. Chen, ‘UAV trajectory planning in cluttered environments based on PF-RRT* algorithm with goal-biased strategy’, *Eng. Appl. Artif. Intell.*, vol. 114, p. 105182, Sep. 2022, doi: 10.1016/j.engappai.2022.105182.
- [29] Y. He, T. Hou, and M. Wang, ‘A new method for unmanned aerial vehicle path planning in complex environments’, *Sci. Rep.*, vol. 14, no. 1, p. 9257, Apr. 2024, doi: 10.1038/s41598-024-60051-4.
- [30] A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, ‘HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots’, *IEEE Access*, vol. 8, pp. 221743–221766, 2020, doi: 10.1109/ACCESS.2020.3043333.
- [31] N. Husam A., E. Donát, and P. Ya-Jun, ‘GMP: Gap-Following Memory and Prediction Based Local Navigation Method for Obstacle Avoidance in Unknown Complex Environments’, *IEEE Trans. Syst. Man Cybern. Syst.*, under revision.
- [32] H. A. Neamah, E. Donát, and P. Korondi, ‘Optimizing autonomous navigation in unknown environments: A novel trap avoiding vector field histogram algorithm VFH+T’, *Results Eng.*, vol. 23, p. 102625, Sep. 2024, doi: 10.1016/j.rineng.2024.102625.
- [33] N. Husam A., E. Donát, and K. Péter, ‘Educational Approach to Trajectory Prediction for Dynamic Obstacle Avoidance Using Curve Fitting Methods’, *ICELIE Conf. 3-6 Novemb. 2024 Chic. Ill. USA*.
- [34] D. Anikó, ‘Másodrendű görbék a projektív síkon. Matematika BSc Szakdolgozat - PDF Free Download’, adoc.pub. Accessed: May 15, 2024. [Online]. Available: <https://adoc.pub/masodrend-grbek-a-projektiv-sikon-matematika-bsc-szakdolgoza.html>
- [35] D. Calisi and D. Nardi, ‘Performance evaluation of pure-motion tasks for mobile robots with respect to world models’, *Auton. Robots*, vol. 27, p. 465, Nov. 2009, doi: 10.1007/s10514-009-9150-y.