

TDK-dolgozat

Szabó István

Pannon Egyetem
Műszaki Informatikai Kar
Villamosmérnöki és Információs Rendszerek Tanszék

Szimultán lokalizáció és térképezés Java alapon

Készítette: Szabó István
4. évfolyam

Témavezető: Dr. Magyar Attila
egyetemi docens

Tartalmi összefoglaló

A 20. század vége felé és a 21. század kezdetén egyre különfélebb és egyre bonyolultabb feladatokat kapnak az úgynevezett mobil robotok. Ezek hely- és helyzetváltoztatáson kívül a környezetükkel való interakcióra, kölcsönhatásra is képesek. Jelen dolgozat célja *szimultán lokalizáció és térképezési* feladat megvalósítása egy LEGO Mindstorms NXT roboton.

A LEGO Mindstorms NXT csomag egy egyszerű ám sokoldalú, több platformon is programozható robotépítő készlet, melyet világszerte használnak az oktatásban. A megépített mobil robot programozása Java nyelven (leJOS) történt. A robot két hajtott kerékkel rendelkező háromkerekű jármű, a környezetből származó információt pedig egy ultrahang szenzor, egy giroszkóp, valamint egy érintés érzékelő mérései szolgáltatják.

Mind a térképkészítés, mind pedig a lokalizáció során kétdimenziós eset feldolgozása volt a cél. A megvalósított robot a térképet az úgynevezett foglaltsági háló alapú térképezési eljárással építi fel, ami egy valószínűségi alapú eljárás. Előnye, hogy a zajos, pontatlan mérésekhez valószínűségi sűrűségfüggvényeket rendel, és ezek alapján állítja elő a térképet, ami szintén valószínűségi jellegű.

A robot mozgása egy költségmátrix alapján, A* algoritmus segítségével történik, mely során az előzőleg felépített valószínűségi térkép, valamint egy súlyozás segítségével valósítjuk meg a mozgást.

A lokalizáció esetében a bővített kiterjesztett Kalman szűrő (Augmented Extended Kalman Filter) lokalizációs eljárás került implementálásra, mely az odometrikus adatok és a zajos mérések alapján kiszámítja a robot becsült helyzetét.

A robot vezérlőegységének véges erőforrásai miatt a roboton, és a vele kommunikáló PC-n is fut egy - egy program, köztük a kommunikáció Bluetooth kapcsolaton keresztül folyik. Az ezzel kapcsolatban felmerülő problémák megoldása is szerves része az elvégzett munkának.

Kulcsszavak: *mobil robotika, SLAM, Kalman szűrés, mesterséges intelligencia, foglaltsági háló, költségmátrix.*

Tartalomjegyzék

Tartalomjegyzék	I
Ábrák jegyzéke	II
Táblázatok jegyzéke	IV
1. Bevezető	1
1.1. Előszó	1
1.1.1. A dolgozat szerkezete	1
1.2. A robot definíciója	2
1.3. Történelem	3
1.4. Mobil robotok	4
1.5. Robot generációk	6
1.5.1. Első generációs robotok	6
1.5.2. Második generációs robotok	6
1.5.3. Harmadik generációs robotok	7
1.6. Irodalmi áttekintés	8
1.7. LEGO Mindstorms NXT	9
1.7.1. A készlet bemutatása	9
1.7.2. Az NXT programozása	10
1.7.2.1. A leJOS programozási nyelv	10
1.8. Problémakitűzés	11
1.9. Jelölések	13
2. SLAM	15
2.1. Bevezetés a SLAM világába	15
2.2. Történelme	15
2.3. A SLAM probléma megfogalmazása és struktúrája	16
2.4. A valószínűségi SLAM és struktúrája	18
3. Foglaltsági háló alapú térképezés	21
3.1. Bevezetés	21
3.2. Az ultrahang szenzor	22
3.3. A térkép készítése	22
3.3.1. Megvalósítása	22
3.3.2. A mérési modell	23
3.3.3. Felmerülő problémák és megoldásuk	26

4. Navigáció	29
4.1. Odometrián alapuló navigáció	29
4.2. Költségmátrixon alapuló navigáció	30
4.2.1. A mátrix megvalósítása	30
4.2.2. Felmerülő problémák és megoldásuk	31
4.3. Útkeresés	31
4.3.1. A* algoritmus	31
5. Lokalizáció	35
5.1. Bevezetés	35
5.2. Bővített kiterjesztett Kalman szűrő	35
5.2.1. Kalman szűrő	35
5.2.2. Kiterjesztett Kalman szűrő	37
5.2.3. Rendszermodell	38
5.2.4. Bővített kiterjesztett Kalman szűrő	40
6. Esettanulmány	43
6.1. Grafikus felhasználói felület	43
6.2. A térkép rajzolása	44
6.3. Tesztek	44
6.3.1. Egyszerű <i>L</i> -alakú labirintus	44
6.3.2. Egyszerű <i>négyszet</i> -alakú labirintus	46
7. Összegzés	49
7.1. Konklúzió	49
7.2. Lehetséges további munka	50
Irodalomjegyzék	50
Melléklet	53
A. Egyebek	55

Ábrák jegyzéke

1.1. Első generációs robot modell [2]	6
1.2. Második generációs robot modell [2]	7
1.3. Harmadik generációs robot modell [2]	7
2.1. Az alapvető SLAM probléma. [27]	17
2.2. Rugó hálózat analógia. [27]	20
3.1. Inverz szenzor modell. [40]	23
3.2. Inverz szenzor modell, a szükséges adatok feltüntetésével. [41]	24
4.1. Az A* algoritmus indulási állapota.	32
4.2. Az A* algoritmus folytatása.	33
5.1. Differenciál-hajtású mobil robot kinematikája.	39
5.2. A bővített kiterjesztett Kalman szűrő működése.	42
A.1. A grafikus felhasználói felület.	55
A.2. Az ultrahang szenzor teszt pályája.	56
A.3. Az L-alakú labirintus a kikerülendő akadállyal.	57
A.4. A négyzet-alakú labirintus, közepén egy oszloppal.	57
A.5. L labirintus, valószínűségi térkép, 1. lépés.	58
A.6. L labirintus, valószínűségi térkép, 15. lépés.	58
A.7. L labirintus, valószínűségi térkép, 28. lépés.	58
A.8. L labirintus, valószínűségi térkép, 2. lépés.	58
A.9. L labirintus, valószínűségi térkép, 26. lépés.	58
A.10.L labirintus, valószínűségi térkép, 40. lépés.	58
A.11.L labirintus, valószínűségi térkép, 61. lépés.	59
A.12.L labirintus, költségmátrix, 1. lépés.	59
A.13.L labirintus, költségmátrix, 30. lépés.	59
A.14.L labirintus, valószínűségi térkép, 80. lépés.	59
A.15.L labirintus, költségmátrix, 11. lépés.	59
A.16.L labirintus, költségmátrix, 35. lépés.	59
A.17.L labirintus, költségmátrix, 81. lépés.	60
A.18.L labirintus, talált legrövidebb út, 15. lépés.	60
A.19.L labirintus, költségmátrix, 95. lépés.	60
A.20.L labirintus, talált legrövidebb út, 83. lépés.	60
A.21.Négyzet labirintus, valószínűségi térkép, 1. lépés.	61
A.22.Négyzet labirintus, valószínűségi térkép, 30. lépés.	61
A.23.Négyzet labirintus, valószínűségi térkép, 103. lépés.	61
A.24.Négyzet labirintus, valószínűségi térkép, 10. lépés.	61

A.25.Négyzet labirintus, valószínűségi térkép, 70. lépés.	61
A.26.Négyzet labirintus, valószínűségi térkép, 109. lépés.	61
A.27.Négyzet labirintus, valószínűségi térkép, 119. lépés.	62
A.28.Négyzet labirintus, költségmátrix, 2. lépés.	62
A.29.Négyzet labirintus, költségmátrix, 106. lépés.	62
A.30.Négyzet labirintus, valószínűségi térkép, 149. lépés.	62
A.31.Négyzet labirintus, költségmátrix, 64. lépés.	62
A.32.Négyzet labirintus, költségmátrix, 128. lépés.	62
A.33.Négyzet labirintus, költségmátrix, 132. lépés.	63
A.34.Négyzet labirintus, talált legrövidebb út, 65. lépés.	63
A.35.Négyzet labirintus, költségmátrix, 141. lépés.	63
A.36.Négyzet labirintus, talált legrövidebb út, 143. lépés.	63

Táblázatok jegyzéke

1.1. Jelölés jegyzék	13
3.1. Az ultrahang szenzor méréseinek eredménye	22

1. fejezet

Bevezető

A tudomány, a technológia - ezt világosan és erősen akarom mondani - nem old meg minden problémát. De tudomány és technológia nélkül semmiféle problémát nem lehet megoldani.

/Teller Ede/

1.1. Előszó

A robotika napjaink egyik leggyorsabban fejlődő területe, ahol napról napra újabbnál újabb megoldások születnek az egyes problémákra, melyek rengeteg kihívást jelentenek a kutatók számára. Ilyen kihívás a szimultán lokalizáció és térképezés problémájának gyakorlati megoldása, melyet jelen dolgozat is célul tűzött ki.

1.1.1. A dolgozat szerkezete

Dolgozatomban mindenek előtt a *Bevezető* fejezetben bemutatom a robot, mint munkavégző eszköz definícióját, történelmét, a napjainkban egyre népszerűbb autonóm mobil robotok leírását, az egyes robot generációk bemutatását, magát a készletet, melyből a saját robotomat építettem, az általam használt programozási nyelvet, valamint összegzem a dolgozatomban bemutatásra kerülő jelöléseket.

Ezek után bemutatásra kerül a *SLAM* fejezetben a megoldott alap probléma történelme, megfogalmazása és struktúrája, mely elengedhetetlen ahhoz, hogy átfogó képet kapjunk a kitűzött feladat terjedelméről és bonyolultságáról.

Az ezután következő fejezetek mutatják be az általam megoldott részproblémákat, valamint az elért eredményeimet. A *Foglaltsági háló alapú térképezés* c. fejezetben bemutatom az ultrahang szenzorral elvégzett tesztek, melyek közelebb vezettek a pontosabb távolságadatok vételéhez és a pontosabb térkép megalkotásához. Bemutatom a használt térképezési algoritmust és mérési modellt, melyet implementáltam, valamint részletezem a használt egyenleteket, és ezen egyenletek kiütköző hibáit, melyek javítása elengedhetetlen feltétele volt pontos térképezési eljárás megvalósításához.

A *Navigáció* c. fejezetben prezentálom az általam megvalósított költségmátrixon és A* algoritmuson alapuló navigációs eljárást, melynél a probléma tökéletes megoldásához a költségmátrix esetében az alapegyenlet saját magam által eszközölt átalakítása is elengedhetetlen feltétele volt. Ezen eljárás biztosítja a térképezési

algoritmus számára, hogy az újabbnál újabb ultrahang mérések különböző helyről származzanak, így lehetővé tegyék új információ integrálását a készülő térképbe, ami nélkülözhetetlen a hiba nélküli működéshez. Navigáció nélkül a robot ugyanarról a helyről szerezne be az információkat, ami az első mérési sorozat után nem szolgálna új információval, ezért van szükség a robot léptetésére, mint említettem korábban. Az általam, erre a problémára optimalizált A* keresés pedig szükséges és elengedhetetlen feltétele a robot számára, hogy két pont között mindig a legrövidebb utat találjuk meg az akadályokkal való ütközés elkerülésével.

A következő implementált részprobléma a *Lokalizáció*, melynél bemutatom a bővített kiterjesztett Kalman szűrő nevezetű lokalizációs eljárást lépésről lépésre felépítve, mely elengedhetetlen az előzőleg említett térképezési és navigációs problémához egyaránt. Eszerint a robotnak minden pillanatban tudnia kell a pontos tartózkodási helyét és irányát a labirintuson belül, hogy a beérkező távolságadatokat tökéletesen dolgozza fel és integrálja az addigi térképbe. Ezen kívül fontos hogy a robot mozgása során pontosan tudjuk, hogy jelenleg hol tartunk a két pont közötti megtalált legrövidebb útban, és mely lesz a soron következő cella, melybe lépni fog. A lokalizáció esetében fontos volt az algoritmust jelen problémára és robotra optimalizálni. Ez is a feladatom részét képezte.

Végezetül prezentálom az *Összegzés* című fejezetben, munkám eredményének sikeres működését két különböző felépítésű teszt pályán, ezeken bemutattva az általam implementált algoritmusok erősségét, valamint az ezekből levont konklúzió és lehetséges további munkák is bemutatásra kerülnek.

1.2. A robot definíciója

A "robot" szó a cseh robota¹ szóból származik.

Jelenleg nincs egységes meghatározás a "robot" szóra. Az International Federation of Robotics (IFR) - melynek célja, hogy előmozdítsa és megerősítse a robotika ipart világszerte - definiálta a robotot [15]:

"Automatikusan vezérelt, újraprogramozható, többféle funkcióra felhasználható manipulatív gép, három vagy több újraprogramozható tengellyel, akár rögzített, akár szabadon mozgó változatban, ipari automatizálási célokra való felhasználásra."

Ebből adódóan a robotok főbb jellemzői:

- Rugalmas újraprogramozhatóság
- Több szabadságfokú mozgáslehetőség
- Anyagok, berendezések, szerszámok mozgatása vagy különböző feladatok megvalósítása

Tehát a robot egy elektromechanikai szerkezet, amely egy adott előzetes programozás után képessé válik különböző feladatok, műveletek végrehajtására. A robot állhat emberi irányítás alatt, vagy akár önállóan is végezhet munkát egy számítógép segítségével.

¹jelentése: szolgaság

A Hollywoodi filmekben feltűnő robotok általában:

- Gyorsabbak, erősebbek, és ügyesebbek, mint az emberek.
- Extra intelligensek.
- Természetes nyelvet beszélnek, és minden gond nélkül megértik az embereket.
- Határozott személyiségűek.
- Általában érzelmeik kimutatására is képesek
- Gyakran világhuralmi törekvéssel érkeznek közénk.

Ezzel szemben a valóságos robotok - egyelőre - nem ezt a képet mutatják. Lehetnek:

- Ipari robotok
- Katonai robotok
- Űrkutató robotok
- Háztartási robotok
- Szórakoztató robotok
- Oktatási, kutató robotok

1.3. Történelem

A robotika szót Isaac Asimov használta először egy 1942-es történetében. Első robot amelyről Asimov írt, egy robot terapeuta volt. Majd 1950-ben kiadja az általa megfogalmazott robotika három törvényét. Az eredeti három törvényben logikai buktatókat fedeznek fel, ezért megjelenik a negyedik, törvénytípus szerint a nulladik törvény. A négy törvény:

0. A robot nem árthat az emberiségnek, és nem nézheti tétlenül, ha az emberiséget veszély fenyegeti.
1. A robotnak nem szabad kárt okoznia emberi lényben, vagy tétlenül tétlenie, hogy emberi lény bármilyen kárt szenvedjen, kivéve, ha ez összeütközésbe kerül a nulladik törvénnyel.
2. A robot engedelmeskedni tartozik az emberi lények utasításainak, kivéve, ha ezek az utasítások a nulladik vagy első törvény előírásaiba ütköznének.
3. A robot tartozik saját védelméről gondoskodni, amennyiben ez nem ütközik a nulladik, az első és második törvény előírásaiba.

1962-ben Amerikában megalakul az első ipari robotgyártó cég, mely az Unimation nevet viselte. Az első általuk gyártott robot a cég nevét kapta, és a General Motors autózúzában teljesített szolgálatot. Az emberek ráeszméltek, hogy a robotmunkások sokkal jobban tűrik a monoton, ám nagy pontosságot igénylő tevékenységeket, ilyen pl. az autógyártásban a szerelősorok mellett dolgozó robotok munkája. E mellett a veszélyes feladatokat, mint például a nukleáris hulladék megsemmisítése, valamint az emberek számára túl nehéz munkákat. Ez utóbbi pl. egy több tonnás szállítmány le- illetve felrakódása 1-2 milliméter pontossággal. Egy robotnak nem

okozna nagy gondot, ha 24 órában kellene dolgoznia, vagy az emberek számára lehetetlen feladatokat kellene megoldania, mint például egy tenger mélyén futó kábelköteg megjavítása. Mindemellett a munkaköltségek csökkenése is a robotok mellett szól. A fenti tények ellenére is elég ellenségesen fogadták a robotmunkásokat, mivel számos munkahely került veszélybe. Ezután Amerikában csak egyetlen robotgyártó cég maradt, a többi főleg japán, valamint európai.

1965-ben Herbert Simon megjósolja, hogy 1985-ben a gépek képesek lesznek megcsinálni mindazt, amit az emberek tudnak. Ugyanebben az évben megalakul a Carnegie Mellon Egyetemen a Robotikai Intézet, melyet Raj Reddy vezet.

1969-ben a Stanford Research Institute-on létrehozzák Shakey-t, aki egyszerű angol szavakkal vezérelhető mozgó robot volt. A robot már rendelkezett kamerával, valamint távolság és érintés érzékelőkkel.

1973-ban jelenik meg a Fuzzy Logikát leíró tanulmány, amely az eddigi egyértelműen behatárolható és besorolható logikával szemben (igen-nem, 0-1) abban jelent újdonságot, hogy a besorolást nem veszi ennyire sarkalatosnak, hanem finomít rajta, lágygá teszi, azaz egy elem nem csak beletartozhat egy halmazba vagy nem, hanem beletartozhat kevésbé, kissé, jobban, nagyon vagy akár egyáltalán nem. Ez a változás nagyon fontos szerepet tölt be a szakértő rendszerek korábbi Boole-féle logikára épülő szabályaiban. A valós életet a maga pontatlanságával pontosabban leírhatóvá teszi ez a módszer (például egy pedál nem csak benyomott, illetve kiengedett állapotban lehet, hanem közbülső fokozatokkal is rendelkezhet).

1979-ben az Amerikai Robot Intézet definiálja a robot fogalmát:

"Újraprogramozható, többfunkciós manipulátor, amelyet azért terveztek, hogy mozgasson alapanyagot, alkatrészeket, eszközöket vagy specializált eszközöket programozott mozdulatokkal, különböző célok teljesítése érdekében."

Az 1990-as években Rod Brooks és csapata a MIT-nél jelentős eredményeket ért el a humanoid robotok építése terén.

2000-ben pedig megjelenik a Sony legújabb vívmánya, AIBO a robot kutya, melynek a gyártását 2006-ban fejezték be. A Sony jelezvén, hogy továbbra is az intelligens robotok mellett marad, 2003-ban bejelenti következő termékét, QRIO-t az emberszabású robotot, melynél a cég már megoldja a két lábon járás problémáit, valamint az emberi arcok feismersét, és érzelmek kifejezését [1].

1.4. Mobil robotok

Napjainkban egyre több figyelmet kapnak a mobil robotok. A fejlesztések irányát meghatározza az a tény, hogy az intelligenciával nem rendelkező hagyományos ipari robotok mellett a hétköznapi használat szintjén is egyre több területen használnak komplexebb, autonómabb, rutinmunkák helyett összetettebb feladatokat végrehajtó mobil robotokat. Ilyen területek lehetnek pl. a háztartások, az egészségügy, stb. A robotok osztályozását és definícióját az alábbi három tényező határozza meg [2]:

- Teljes egészében ember által készített szerkezetek.
- Mozogni tudnak, a mozgásban több szabadságfokkal rendelkeznek.
- Tevékenységüket részben, vagy teljesen önállóan irányítják.

Ez az elemzés elsősorban emberi feladatokat végrehajtó ágensekkel foglalkozik. Képességeiket, valamint a jelenlegi és későbbi alkalmazásaikat e három szempont közül leginkább a mobilitás és autonómia határozzák meg.

Robotkarokat - más néven manipulátorokat - viszonylag könnyű létrehozni és megfelelően programozni. Ezek feladataikat korlátozott térben végzik el, azaz fizikailag a munkavégzés helyszínéhez kötöttek, nem mozdulnak el. Egy adott munkatérben csuklók kinematikai láncán keresztül szabadon programozhatóak, bármely pontot képesek elérni. A gyakori ipari alkalmazásokat, mint például hegesztés, összeszerelés, festékszórás nagyon könnyen lehet szemléltetni. Azonban, ha egy robotot ki akarunk küldeni a nagyvilágba, a fejlesztőknek rengeteg felmerülő problémát kell megoldaniuk. Ezek közül a legfontosabb az adott terepre optimalizált, működő helyváltoztató rendszer. Ezen robotoknak rendelkezniük kell valamilyen fokú intelligenciával és autonómiával, hogy a felmerülő problémákat - ami lehet például a folyamatosan változó környezet - önállóan, időben és pontosan meg tudják oldani. Ahhoz, hogy ezek a környezetben tapasztalt változások ne zavarják össze őket, hibatűrőknek kell lenniük.

A mobil robotokat csoportosíthatjuk a következő módon:

- Kerekeken guruló robotok
- Lépegető robotok
- Repülő robotok
- Egyéb (úszó, csúszó, stb.)

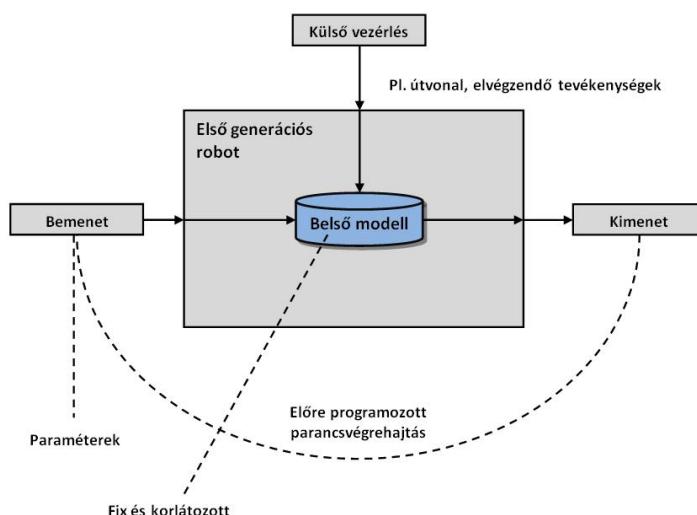
A mobil robotok akkor váltják be a velük szemben támasztott elvárásokat, ha a feladataikat nem csak laboratóriumi környezetben, hanem a folyamatosan, akár percről percre módosuló külvilágban is képesek valós időben elvégezni. Beépített, önállóan kibővített modellel kell rendelkezniük környezetükről, melyet érzékelniük kell saját érzékelőikkel, majd feldolgozniuk, elemezniük, megtalálni benne saját pontos pozíciójukat, végül meg kell tervezniük mozgásukat, amit lehetőleg minél pontosabban végre kell hajtaniuk. Ha eleget tesznek ezeknek a feltételeknek, akkor tudják csak megkönnyíteni az emberek mindennapi életét, mint például takarítás, szórakoztatás, bevásárlóközpontokban, kórházakban vagy múzeumokban - azaz belső, struktúrált környezetben - való munka, erdőkben, bányákban, tenger fenekén vagy netán katasztrófa sújtotta területeken, világűrben - azaz külső, struktúrátlan környezetben - való munka.

A mai világban megtalálható mobil robotok megbízhatóságával kapcsolatban számos kivetnivalót találhatunk, s ennek következtében nagyobb szerephez kell jutniuk a speciális megbízhatóság-mérő módszereknek, melyekkel előre jelezhető a robot meghibásodása, meghibásodási rátája, feladatának sikeres végrehajtási valószínűsége. Ezen kívül pedig a fejlődés ütemét a technológiai korlátok mellett gazdasági szempontok is késleltetik. Már az első robotok tervezésekor azt jósolták előre, hogy csak tíz-tizenöt évet kell várni az intelligens gépekig. Nem tudni pontosan ezek a befektetések mikor térülnek meg, mivel a mobil robotok gyártásához mindig az éppen aktuális csúcstechnológiára van szükség, így ez egy rendkívül költséges ágazat.

1.5. Robot generációk

1.5.1. Első generációs robotok

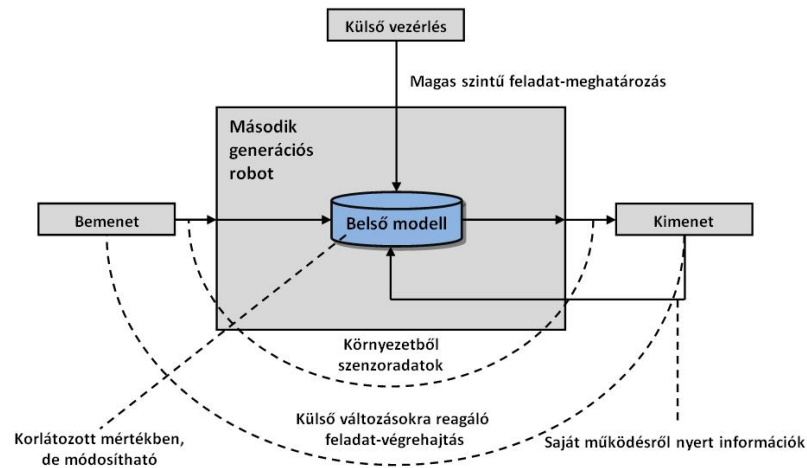
A 60-as években járunk. Ezek a robotok még csak egyszerű feladatokat látnak el, mint például különböző tárgyak felemelése - lerakása. Kizárólag vezérléssel működ-tethetőek, egy számítógép adja meg a mozgásuk pontos útvonalát és az elvégzendő műveletet, így saját "akarattal" nem rendelkeztek. Nem rendelkeznek a környezet változásait figyelő külső érzékelőkkel, szenzorokkal, így ezekre a változásokra nem is reagálnak. Az 1.1. ábrán ezen generációs robotok modellje látható.



1.1. ábra. Első generációs robot modell [2]

1.5.2. Második generációs robotok

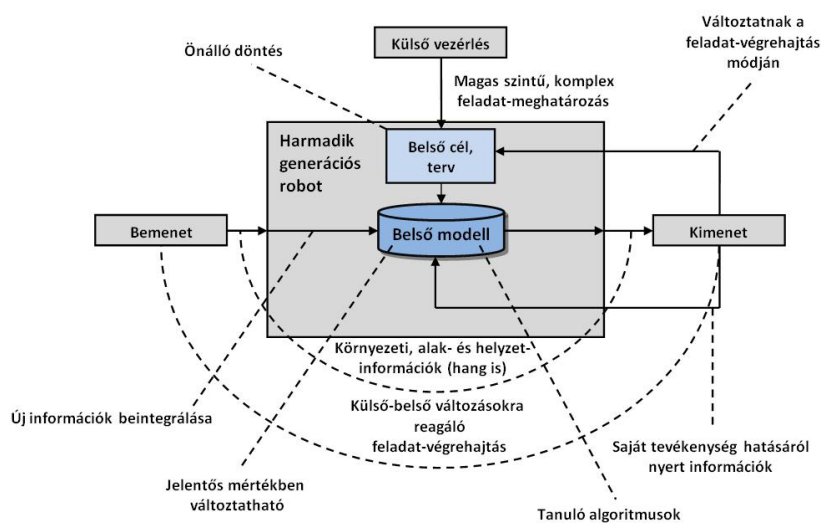
A 70-es években a környezetüket már érzékelők segítségével vizsgálták. Tevékenységüket immár módosítani is tudták, ha azt az adott szituáció megkövetelte tőlük, például ha egy hirtelen felbukkanó akadályt kellett kikerülniük. Feladataikat magas szintű programozási nyelven lehet megadni. 1972-ben az USA-ban már 12 olyan vállalat volt, amelyek ipari robotokat gyártottak. Megjelennek az első teljesen számítógép-irányítású robotok. Valamint ezen években kezdődnek meg az európai autógyárakban a saját robotfejlesztések. Az alábbi ábrán látható ezen robotok modellje (1.2. ábra).



1.2. ábra. Második generációs robot modell [2]

1.5.3. Harmadik generációs robotok

Itt már tökéletesen megfigyelhető a mesterséges intelligencia térhódítása (1.3. ábra). Az érzékelőkből származó információt képesek feldolgozni, eltárolni és a magukról és környezetükről tárolt modellt önállóan módosítani. Alakzatokat és helyzeteket ismernek fel. Hanggal is vezérelhetők, amire képesek ugyancsak hanggal válaszolni. Képesek információ kiválasztására és kombinálására. Rendelkeznak tanuló algoritmusokkal, amik segítségével felhasználhatják korábbi tapasztalataikat. Alkalmazási körük bővül, jelenleg már laboratóriumi, mélytengeri, űrkutatási és mezőgazdasági területeken is hasznosítják őket.



1.3. ábra. Harmadik generációs robot modell [2]

A hagyományos ipari robotok, mint például a robotkarok, még csak az első és második generációba tartoznak, míg az autonóm mobil robotok már a második és harmadik generációba. További fejlesztések újabb generációkat hozhatnak létre. Ezekbe a generációkba tartozó robotoknak a racionális cselekedetekhez a következő elvárásoknak kell megfelelniük:

- feldolgozási képesség (tárolás, számítás),
- kommunikációs képesség (információkeresés és továbbítás),
- társítási képesség (információk összekapcsolása),
- reprezentációs képesség (belső modell),
- következtetési képesség (rejtett információk kinyerése),
- tervezési képesség (célvezéreltség),
- újrafelhasználási képesség (tanulás),
- önrendelkezési képesség (autonómia).

Valamint az információkezelés öt pontjának:

- információfelvevő képesség,
- információfeldolgozó képesség,
- tudáskezelő képesség,
- tanulási (adaptálódás) képesség,
- mozgási (helyváltoztatási, reagálási) képesség.

1.6. Irodalmi áttekintés

Rengeteg elméleti és gyakorlati megoldás született a SLAM problémára az elmúlt közel húsz évben. Többek között Agostino Martinelli^a, Viet Nguyen^b, Nicola Tomatis^c és Roland Siegwart^b munkája [33], mely leírja a SLAM egy megoldását sztochasztikus térkép környezetben, relatív térképre alapulva, vagy például K. M. Wurm, C. Stachniss és G. Grisetti munkája [34], mely szintén megvalósítja a SLAM-et egy olyan megközelítésben, melyben több különböző térkép reprezentációt használ szimultán. A legtöbb ilyen jellegű munka az EKF² algoritmuson alapul, mint például A. Chatterjee, O. Ray, A. Chatterjee, A. Rakshit munkája [35], melynél egy EKF-en alapuló SLAM implementációt valósítottak meg egy mobil roboton beltéri környezetben, felhasználva két webkamerát egy sztereó látási mechanizmushoz. Más munkák, például Luis Morenó, Santiago Garrido, Dolores Blanco és M. Luisa Muñoz munkája [36] egy egészen más módszert alkalmaz a SLAM probléma megoldására. Ez pedig az úgynevezett ELF³, melynél sztochasztikus kereséseket végzünk egy állapot tér mentén, hogy megtaláljuk a robot legjobb pozícióbecslését. A Távol-Keleten is születtek megoldások, nevezetesen C. Haoyao, S. Dong, Y. Jie munkája [37], mely globális lokalizációt alkalmaz többrobotos formációk esetén. Mások, nevezetesen M. Begum, G. K.I. Mann, R. G. Gosine munkája [38], mely fuzzy logikát és genetikus algoritmust alkalmaz a SLAM megoldásának megtalálására. Végül pedig C. Cadena, J. Neira munkája [39], melyben bemutatják az úgynevezett Combined

²Extended Kalman Filter - kiterjesztett Kalman szűrő

³Evolutionary Localization Filter - fejlődéses lokalizációs szűrő

Kalman-Information Filter-t, mely tulajdonképpen egy igen megfontolt kombinációja a kiterjesztett Kalman szűrőnek és a kiterjesztett információs szűrőnek, melynél hatékonyan lehet megvalósítani a SLAM-et kiterjedt környezetekben.

1.7. LEGO Mindstorms NXT

1.7.1. A készlet bemutatása

A LEGO Mindstorms NXT egy viszonylag fejlett robotépítő készlet, melyet a dán⁴ LEGO cég dobott a piacra [3]. Az első széria 1988-ban látta meg a napvilágot, azóta is remek kikapcsolódást nyújt a szórakozni vágyóknak és komoly kihívást a vállalkozó kedvű fejlesztőknek egyaránt. Rengeteg formájú és funkciójú robot megalkotható ebből a készletből, amit viszonylag olcsón megkaphatunk a komolyabb szinten programozható robot készletekhez képest. Egyetemi keretek között az oktatók is előszeretettel választják ezt a nem túl drága megoldást arra, hogy bemutassák tanítványaiknak a "mérnöki tudományok" eme ágát.

A központi egység az intelligens NXT *tégla*⁵, melyben megtalálható [16]:

- egy 32-bit ARM7 mikroprocesszor,
- 256 Kbyte FLASH, 64 Kbyte RAM memória,
- 8-bit mikroprocesszor,
- 4 Kbyte FLASH, 512 Byte RAM memória,
- 4 bemeneti port a szenzorok adatainak vételéhez,
- 3 kimeneti port a motorokhoz való csatlakozáshoz,
- Bluetooth vezeték nélküli kommunikációs eszközt,
- 100x60 pixeles monokróm LCD kijelző,
- USB 2.0 port,
- energiaforrás⁶,
- hangszóró⁷,
- 4 darab gomb a navigációhoz

A szenzorok és motorok összekapcsolása a *téglával* RJ12 kábelek segítségével történik. Az alapsomag tartalmaz öt szenzort a környezetből érkező információk érzékelésére, melyek:

- Érintésérzékelő (2 darab)
- Fényérzékelő
- Hangérzékelő (mikrofon)
- Ultrahang érzékelő

Ezekon kívül számos kiegészítő szenzor vásárolható meg, melyek:

⁴A Magyarországon kapható LEGO elemek nagy részét Nyíregyházán gyártják.

⁵Téglára emlékeztető alakja miatt nevezik így.

⁶A fogyasztói változata 6 darab, egyenként 1,5V-os elemet, míg az oktatási célokra szánt készlet egy újratölthető Li-ion akkumulátort és egy hozzá való töltőt tartalmaz.

⁷Hangfájlok lejátszásához a mintavételi arány maximum 8kHz.

- Iránytű
- Színérzékelő
- Gyorsulásmérő
- Giroszkóp
- Stb.

Viszont ez utóbbiak az alap csomaghoz képest is elég drágák.

1.7.2. Az NXT programozása

Az NXT programozásának kétféle módja létezik. Az egyik, amikor egy PC-n megírt programot töltünk rá az NXT-re, ami önállóan, PC-s irányítás nélkül fut a *téglán*. Ennek a módszernek egy nagyon nagy hátránya az, hogy a kis memóriakapacitású eszközre csak igen kis méretű programokat tölthetünk rá, így ezek a programok csak korlátozott tevékenységre utasíthatják a robotunkat. Előnye, hogy nagyfokú autonómia érhető el vele. A másik módszer nagyon nagy előnye, hogy a nagy számítási kapacitásokat igénylő feladatok végrehajtása a PC-re hárul, míg a robotunk tulajdonképpen csak az érzékelőkből származó adatokat, információkat küldi el a számítógépünknek, így nagy komplexitású feladatokat is végrehajthatunk. Ennek egy igen nagy hátránya viszont, hogy a kapcsolat a két eszköz között USB porton vagy Bluetooth-on keresztül jöhet létre, és a feladat végrehajtásáig ennek a kapcsolatnak megfelelőnek is kell maradnia. Az NXT egyik nagy előnye, hogy sokféle platform alatt programozható:

- Matlab: Hatékony numerikus algoritmusok programozhatók le vele.
- NBC: Next Byte Code, egy alacsony szintű assemblyhez hasonló nyelv.
- NXC: Not Exactly C, egy C alapú programozási nyelv.
- RobotC, szintén C alapú nyelv, melyet a Carnegie Mellon Robotics Academy fejlesztett ki.
- Java (leJOS): [1.7.2.1.](#) fejezetben erről bővebben.
- LabVIEW: grafikus nyelv, mely az utasítások és függvény hívások helyett az adatfolyamra helyezi a hangsúlyt.
- NXT-G: A LEGO saját grafikus nyelve, mellyel akár gyerekek is tudnak vizuálisan programozni egész könnyedén. Hasonlít a LabVIEW-hoz, mivel a National Instruments fejlesztette ki a LEGO számára.

1.7.2.1. A leJOS programozási nyelv

A leJOS⁸ [17] egy kis JAVA virtuális gép, melyet 2006-ban portáltak az intelligens NXT *téglára*. Feltelepítve az NXT-re lehetővé teszi az NXT-vel kapcsolatos lehetőségekkel kibővített Java nyelvű programok futtatását a robotunkon. A leJOS NXJ tartalmazza az összes osztályt az NXJ API-ból, valamint az ezen osztályok segítségével megírt kódok egyszerűen feltölthetők az NXT *téglára*. Ezen programozási nyelv a következőket kínálja:

- Objektum-orientált nyelv (Java).

⁸LEGO Java Operating System, ejtsd, mint a spanyol *lejos* jelentése: *jóval* szót.

- Szálkezelés.
- Tömbök, beleértve a több-dimenziós tömböket.
- Rekurzió.
- Szinkronizáció.
- Kivételek (exceptions) kezelése.
- JAVA típusokat, beleértve: float, long, String.
- A legtöbb java.lang, java.util és java.io osztályok.
- Jól dokumentált robotika API.

Az általam preferált nyelv a leJOS volt, mely teljes mértékben megfelelt elvárásaimnak.

1.8. Problémakitűzés

Dolgozatom írása során feladatomból volt:

- Egy két hajtott kerékkel rendelkező differenciál-hajtású mobil robot építése.
- A megépített robot programozása JAVA leJOS nyelven.
- Az érdemi munka megkezdése előtti irodalomkutatás, mely tartalmazza az alapprobléma megismerését és a megoldás alternatíváinak felkutatását.
- A részproblémák⁹ megoldásának kiválasztása.
- Ezen megépített robot segítségével a szimultán lokalizáció és térképezés problémáját megoldani az alábbi részproblémák implementálásával:
 - Létrehozni egy foglaltsági háló alapú térképezési eljárást.
 - Egy költségmátrixon és A* kereséssel alapuló navigációs algoritmust.
 - Valamint egy bővített kiterjesztett Kalman szűrőt használó lokalizációs eljárást.
- Az egyes részproblémák elméletének dokumentálása.
- A SLAM-et, és ezzel együtt ennek részproblémáit megoldó szoftver implementálása.
- A felépített mobil robot memóriakapacitási korlátai miatt egy megfelelő szoftverarchitektúra kiépítése, mely a következő részekből áll:
 - Robot oldali program, mely a szenzorok és motorok kezelésének feladatát látja el.
 - PC oldali főprogram, mely a robottól érkező információk feldolgozásáért, valamint a PC-re megírt osztályok kezeléséért felel.
- A szoftverarchitektúrából adódó kommunikáció kiépítése, mely Bluetooth kapcsolaton keresztül történik. Az ebből adódó esetleges hibák kiküszöbölése.
- Egy saját, erőforráskímélő térképrajzoló algoritmus megalkotása.
- Egy felhasználóbarát grafikus felhasználói felület elkészítése, mely minden lépésben prezentálja az addig elkészült térképet.

⁹Térképezés, navigáció és lokalizáció.

- Két különböző felépítésű és tulajdonságú labirintus megépítése a rendelkezésre álló alapanyagok segítségével.
- Az elkészült program és robot tesztelése a két felépített tesztpályán.
- Eredményeim összegzése, kiértékelése és dokumentálása.

Az egyes részproblémák közötti kapcsolatok az alábbi felsorolásban tekinthetők meg:

- A megvalósított térképezési eljárás elengedhetetlen a SLAM probléma tökéletes megvalósításához, mivel csak ezáltal tudunk egy pontos térképet építeni a környezetünkből érkező szenzoradatok segítségével.
- A térképezéshez egy megfelelő navigációs algoritmus választása nélkülözhetetlen, hogy egyrészt a robot az újabb szenzorméréseket is sikerrel integrálhassa a készülő térképbe, másrészt pedig fontos, hogy kifogástalanul tudjon mozogni az adott környezetben a saját épülő térképe segítségével.
- Az előzőleg említett eljárások viszont használhatatlanok lennének egy tökéletesen működő lokalizációs algoritmus elkészítése nélkül, ugyanis a robot nélküle nem tudná, hogy a jelenlegi szenzoradatok, melyeket vételezett, a környezet mely részéből származnak, ő maga pontosan hol helyezkedik el, mi a pozíciója, és egyáltalán merre induljon el feltérképezetlen területek után kutatva.

A következőkben az elért eredményeimet fogom bemutatni.

1.9. Jelölések

A dolgozatban használt jelöléseket a 1.1. táblázatban foglaltam össze.

Szimbólum	Mennyiség vagy objektum	SI Mértékegység
$\mathbf{x}$	állapotvektor	-
x, y	pozíció	-
θ	orientáció	rad
$\mathbf{z}$	megfigyelések vektora	-
Pr	valószínűség	-
v	lineáris sebesség	m/s
ω	szögsebesség	rad/s
γ	keréksugár	m
T	mintavételi idő	s
$\mathbf{u}$	bemenetek vektora	-
k	időpillanat	-
m	tájékozódási pont	-
Occ	foglaltsági határ	-
d	kerekek távolsága	cm
w	a folyamat normál eloszlású fehér zaja	-
v	a mérés normál eloszlású fehér zaja	-
$\mathbf{A}$	állapot egyenlet Jakobi mátrixa	-
$\mathbf{P}^-$	a priori kovariancia	-
$\mathbf{P}$	a posteriori kovariancia	-
$\hat{\mathbf{x}}^-$	a priori állapot vektor	-
$\hat{\mathbf{x}}$	a posteriori állapot vektor	-
$\mathbf{W}$	folyamat hiba	-
$\mathbf{Q}$	folyamat hiba kovariancia	-
$\mathbf{K}$	Kalman erősítés	-
$\mathbf{H}$	mérési egyenlet Jakobi mátrixa	-
$\mathbf{V}$	mérési hiba	-
$\mathbf{R}$	mérési hiba kovariancia	-
$\mathbf{I}$	egység mátrix	-

1.1. táblázat. Jelölés jegyzék

2. fejezet

SLAM

2.1. Bevezetés a SLAM világába

A *szimultán lokalizáció és térképészet*, azaz SLAM¹ problémájáról beszélünk, ha egy mobil robotot elhelyezünk egy ismeretlen környezetben és ismeretlen pozícióban, majd ez a robot fokozatosan felépít egy térképet a környezetéről vont következtetései segítségével, miközben szimultán meghatározza a pozícióját a felépített térképen. A SLAM megoldását, mint "Szent Grált" keresi a robotika közösség, hogy létrehozzanak egy robotot, ami teljes mértékben autonóm. A SLAM megvalósítására számos elméleti megoldás látott napvilágot különböző formában, valamint számos implementáció született rá az elmúlt évtizedben. Többek között Csorba munkája a konvergencia alapvető elméletéről [18, 19], valamint számos csoport munkája a térképészeti és lokalizációs problémáról, nevezetesen a Massachusetts Institute of Technology [20], Zaragoza [21, 22], az ACFR Sydneyben [23, 24] és mások [25, 26] beltéri, kültéri és tenger alatti környezetben. Ugyanakkor jelentős kérdések maradnak megválaszolatlanul a SLAM gyakorlati megvalósításával kapcsolatban.

A térképezés az egy sor érzékelő által összegyűjtött információ egy közös modellben való integrációjának a problémája. Ezt egy jellemző kérdéssel lehet leírni: *Hogy néz ki a világ?* Központi szempont a térképezésben a környezet reprezentációja és a szenzor adatok értelmezése.

Ezzel szemben a lokalizáció a robot pozíciójának és orientációjának a térképhez viszonyított becslésének a problémája. Tehát a robotnak meg kell tudnia válaszolnia a következő kérdést: *Hol vagyok?* Tipikus megoldás, amikor a robot kezdeti helyzete ismert és ezután próbáljuk meg követni a robotot, míg egy másik a globális lokalizáció, ahol nincs, vagy csak igen kevés előzetes ismeretünk van a környezetről a kiinduló pozícióban. A gyakorlatban ezt a két kérdést nem lehet egymástól függetlenül kezelni.

2.2. Történelme

A SLAM problémája 1986-ban fogalmazódott meg, a San Francisco-ban megrendezésre kerülő IEEE Robotics and Automation Conference nevű konferencián. Ekkoriban a valószínűségi metódusok még csak kezdtek bemutatkozni a robotikában és a mesterséges intelligenciában. Néhány kutató, többek között Peter Cheeseman, Jim

¹Simultaneous Localization and Mapping

Crowley, és Hugh Durrant-Whyte kereste a becslélmélet alkalmazási lehetőségét a térképezési és lokalizációs problémákban. Ezen a konferencián hosszasan tárgyaltak a következetes térképezésről, melynek eredménye lett egy felismerés, miszerint a valószínűségi térképezés egy alapvető probléma a robotikán belül. Az elkövetkező néhány évben rengeteg kulcsfontosságú munka született, így például Smith és Cheesman [11], valamint Durrant-Whyte [12] statisztikai alapon határozta meg a tereptárgyak és a geometriai bizonytalanságok közötti kapcsolatot. E munka legfontosabb része megmutatta, hogy létezik egy magas fokú korreláció egy adott térképen megtalálható különböző tereptárgyak pozíciójának becsléseiben. Ezen kívül rávilágított, hogy ha egy mobil robot átmegy egy területen, miközben megfigyeléseket végez az őt körülvevő tereptárgyakról, akkor ezen tereptárgyak pozícióbecslései mind szükségszerűen korreláltak lesznek egymással a közös hiba miatt, ami a becsült jármű-pozícióból adódik. Tehát a korreláció a megfigyelések számával növekedik. Ennek hatására rájöttek, hogy a kombinált lokalizációs és térképészeti probléma teljes megoldásához szükség lenne arra, hogy valamennyi tereptárgy megfigyelése után frissítsük a jármű és az összes addigi tereptárgy pozícióját. Ehhez viszont elengedhetetlenné válna, hogy karban tartsunk egy olyan nagy méretű állapotvektort, mely sorrendben tartalmazná az összes megtalált tereptárgyat. Ennek köszönhetően viszont nagyban megnőne a szükséges számítási kapacitás, ami hosszú távon szintén nem megfelelő megoldás.

Döntő fontosságú, hogy ez a munka nem figyelt a térkép konvergencia tulajdonságaira, miszerint a térkép becsült hibái nem konvergálnak, hanem egy hibahatáron belül, mely hibahatár határtalanul növekszik, végtelen mozgásra hasonlít a viselkedésük. A térképezési probléma számítási kapacitása és a térkép konvergencia tulajdonságainak a pontos ismerete nélkül, a kutatók a tereptárgyak közötti korrelációt minimálisnak, vagy teljesen megszüntnek feltételezték. Ezen okok miatt megpróbálták vagy csak a térképezésre, vagy csak a lokalizációra szétválasztani a problémát.

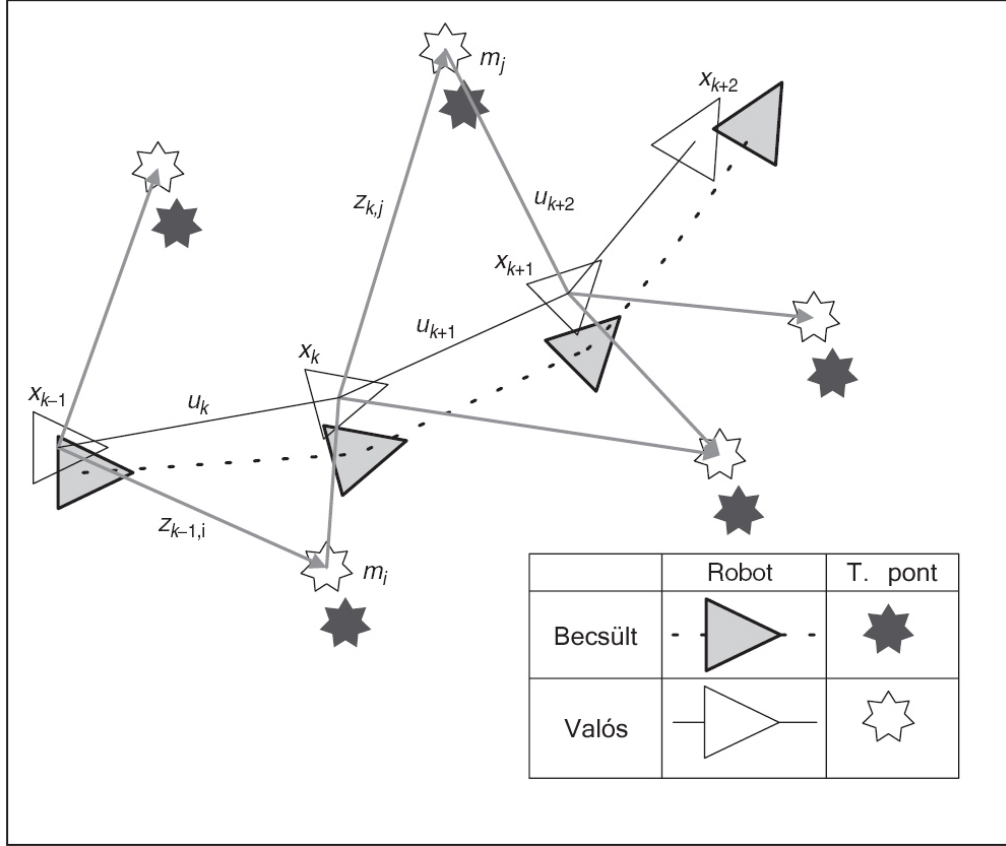
Az áttörés 1995-ben történt meg, mikor rájöttek, hogy a kombinált lokalizációs és térképészeti probléma - amit megfogalmaztak egyetlen becslési problémaként - valójában konvergens. A legfontosabb felfedezés, hogy a korreláció a tájékozódási pontok között, amit a legtöbb kutató minimalizálni próbált, valójában a legkritikusabb része a problémának, sőt mi több, minél nagyobb a korreláció, annál jobb a megoldás. A SLAM probléma struktúráját, valamint a konvergencia eredményeket 1995-ben az International Symposium on Robotics Research nevű konferencián mutatták be egy mobil robotikával foglalkozó munkában. [4]

Ezután a munkák a számítási hatékonyság növelésére koncentráltak leginkább. Az 1999-es International Symposium on Robotics Research volt az egyik legfontosabb történelmi pont, ahol megtartották az első SLAM ülést, valamint ahol bizonyos fokú konvergenciát ért el Thrun [5] a Kalman szűrő alapú SLAM és a valószínűségi lokalizációs és térképező metódusok között.

2.3. A SLAM probléma megfogalmazása és struktúrája

A 2.1. ábra a SLAM alapproblémát mutatja be. Jelen helyzetben a robot egy ismeretlen területen halad keresztül, mely során relatív megfigyeléseket végez az ismeret-

len számú tájékozódási pontokról szenzora segítségével, mely megtalálható magán a roboton. Egyidejűleg szükséges becsülni a robot és a tájékozódási pontok elhelyezkedését. A valós elhelyezkedéseket sosem ismerjük, vagy sosem tudjuk pontosan mérni közvetlenül. A megfigyelések a robot és a tájékozódási pont valós helyzete között helyezkednek el. A k az aktuális időpillanat jele az alábbi felsorolásban.



2.1. ábra. Az alapvető SLAM probléma. [27]

- $\mathbf{x}_k$: Az állapotvektor leírja a pozícióját és orientációját.
- $\mathbf{u}_k$: A bemeneti vektor, mely a $k - 1$ -edik időpillanatban, az $\mathbf{x}_k$. pozícióba történő mozgást adja meg.
- $\mathbf{m}_i$: Az i -edik tájékozódási pont helyzetét leíró vektor. A pont valós helyzete időben állandó.
- $\mathbf{z}_{ik}$: A k -adik időpillanatban, az i -edik tájékozódási pontra vonatkozó megfigyelés. Ha többszörös tájékozódási pont észlelés történt egy adott időpillanatban, vagy egy speciális tájékozódási pont nem releváns, akkor a megfigyelést egyszerűen $\mathbf{z}_k$ -nak is írhatjuk.
- $X_{0:k} = \{x_0, x_1, \dots, x_k\} = \{X_{0:k-1}, x_k\}$: A jármű állapotának a története, az aktuális és korábbi állapotok halmaza.
- $U_{0:k} = \{u_0, u_1, \dots, u_k\} = \{U_{0:k-1}, u_k\}$: A bemenetek története, az aktuális és korábbi bemenetek halmaza.

- $Z_{0:k} = \{z_0, z_1, \dots, z_k\} = \{Z_{0:k-1}, z_k\}$: Az összes tájékozódási pontra vonatkozó megfigyelések halmaza.
- $m = \{m_1, m_2, \dots, m_n\}$: Az összes tájékozódási pont.

2.4. A valószínűségi SLAM és struktúrája

A valószínűségi formában a szimultán lokalizációs és térkép építési probléma szükségessé teszi a robot és a tájékozódási pontok helyzetének ismeretét minden k -adik időpillanatban.

$$Pr(\mathbf{x}_k, m | Z_{0:k}, U_{0:k}, \mathbf{x}_0) \quad (2.1)$$

Általában a SLAM problémának a rekurzív megoldása hatékonyabb. Kezdve a robot és a tájékozódási pontok helyzetének együttes eloszlásról történő becslésével a $k-1$ -edik időpillanatban,

$$Pr(\mathbf{x}_{k-1}, m | Z_{0:k-1}, U_{0:k-1}) \quad (2.2)$$

valamint az $\mathbf{u}_k$ bemeneti vektor és $\mathbf{z}_k$ megfigyelés segítségével kiszámolható az együttes becslés a *Bayes-tétel* alkalmazásával. Ehhez szükségünk van egy állapot átmenet és egy megfigyelési modellre, amely leírja a bemenet hatását, illetőleg a megfigyeléseket.

A megfigyelési modell megadja a $\mathbf{z}_k$ megfigyelés valószínűségét, a robot és a tájékozódási pontok helyzetének ismeretében. Mindez leírható a 2.3. valószínűség formájában

$$Pr(\mathbf{z}_k | \mathbf{x}_k, m). \quad (2.3)$$

Logikus a feltételezés, miszerint ha a robot helyzete és a térkép definiálva van, akkor a megfigyelések feltételesen függenek a térképtől és a robot jelenlegi helyzetétől.

A robot mozgási modellje leírható valószínűségi eloszlás formájában a 2.4. kifejezés szerint

$$Pr(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \quad (2.4)$$

Vagyis az állapot átmenet feltételezhetően egy *Markov folyamat*, amelyben az $\mathbf{x}_k$ következő állapot csak az ezt közvetlenül megelőző $\mathbf{x}_{k-1}$ állapottól és az $\mathbf{u}_k$ bemenettől függ, viszont független a megfigyelésektől és a térképtől.

A SLAM algoritmus megvalósítható szabványos kétlépcsős rekurzív jóslat (time-update) korrekció (measurement-update) formában, ahogy azt a 2.5. és 2.6. egyenletek mutatják:

Time Update

$$Pr(\mathbf{x}_k, m | Z_{0:k-1}, U_{0:k}, \mathbf{x}_0) = \int Pr(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) Pr(\mathbf{x}_{k-1}, m | Z_{0:k-1}, U_{0:k-1}, \mathbf{x}_0) d\mathbf{x}_{k-1} \quad (2.5)$$

Measurement Update

$$Pr(\mathbf{x}_k, m | Z_{0:k}, U_{0:k}, \mathbf{x}_0) = \frac{Pr(\mathbf{z}_k | \mathbf{x}_k, m) Pr(\mathbf{x}_k, m | Z_{0:k-1}, U_{0:k}, \mathbf{x}_0)}{Pr(\mathbf{z}_k | Z_{0:k-1}, U_{0:k})} \quad (2.6)$$

A fenti két egyenlet egy rekurzív eljárást biztosít a $Pr(\mathbf{x}_k, m | Z_{0:k}, U_{0:k}, \mathbf{x}_0)$ valószínűség kiszámításához, melynél a robot $\mathbf{x}_k$ állapota és az m térkép egy k -adik időpillanatban az összes $Z_{0:k}$ megfigyelésen, az összes $U_{0:k}$ vezérlő bemeneten alapszik

egészen a k -adik időpillanatig. A rekurzió a robot modell és a megfigyelési modell függvénye.

Érdemes megjegyezni, hogy a térkép építés problémája megfogalmazható feltételes sűrűség számításként $Pr(m|X_{0:k}, Z_{0:k}, U_{0:k})$. Ez azon a feltételezésen alapul, hogy a robot $\mathbf{x}_k$ helyzete ismert (vagy legalábbis determinisztikus) minden esetben, feltéve, ha ismerjük a kiindulási elhelyezkedést. Az m térkép ezután megalkotható a különböző pontokban tett megfigyelések fúziójaként. Ezzel szemben a lokalizációs probléma valószínűségi eloszlás számításként fogalmazható meg $Pr(\mathbf{x}_k|Z_{0:k}, U_{0:k}, m)$. Ez feltételezi, hogy a tájékozódási pontok helye ismert, és a cél az, hogy kiszámítsuk a robot becsült helyzetét ezen tereptárgyak tekintetében.

A megfigyelési modell $Pr(\mathbf{z}_k|\mathbf{x}_k, m)$ kifejezetten függ a robot és a tájékozódási pontok helyzetétől. Ebből következik, hogy az együttes eloszlást nem lehet felosztani ilyen nyilvánvaló módon, mint ahogy a 2.7. egyenlőtlenség mutatja

$$Pr(\mathbf{x}_k, m|\mathbf{z}_k) \neq Pr(\mathbf{x}_k|\mathbf{z}_k)Pr(m|\mathbf{z}_k), \quad (2.7)$$

és valóban a jól ismert korai munkák esetén a konzisztens térképezésnél, a felosztás ellentmondásos becsléshez vezetne. Mindazonáltal a SLAM problémának több struktúrája van, mint az nyilvánvaló a fenti egyenletekből.

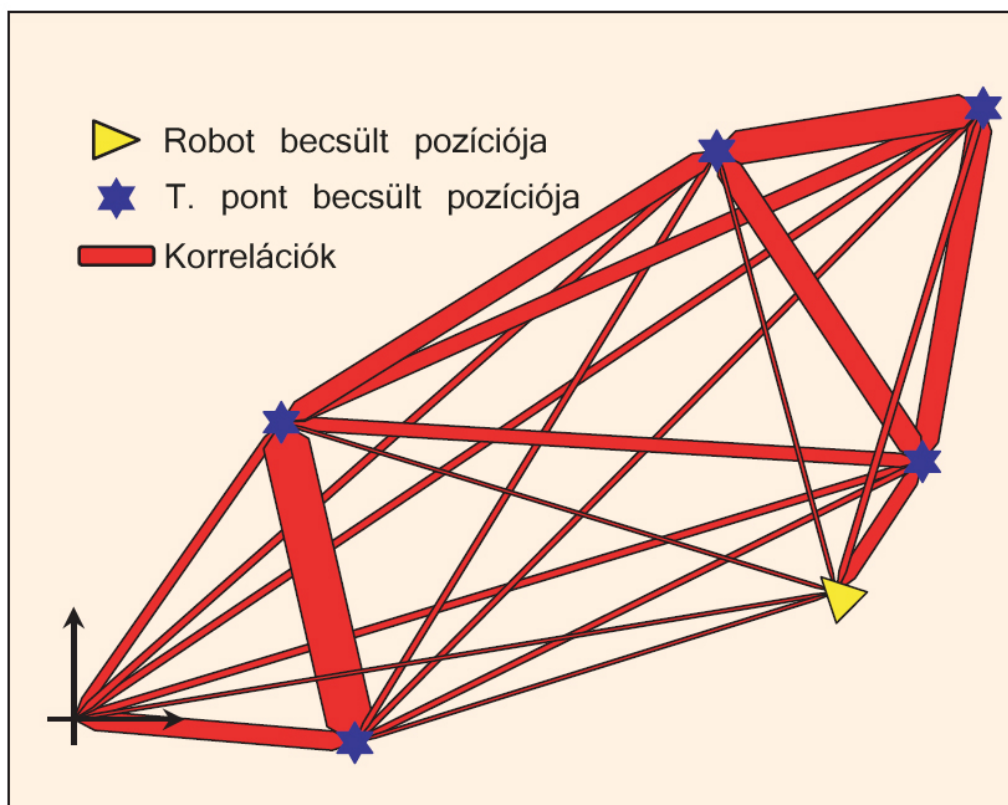
Tekintsük újra az 2.1. ábrát, ahol megfigyelhető, hogy a tájékozódási pontok becsült és valós helyzete közötti különbség minden egyes pontnál körülbelül azonos. Ennek oka az, hogy a tájékozódási pontokra tett megfigyelések ideje alatt a robot helyzete is hibákkal terhelt ismert. Ez viszont azt jelenti, hogy a hibák a tájékozódási pontok helyzetének becslésekor erősen korreláltak. Gyakorlatilag két tájékozódási pont relatív helyzete, azaz a két pont közötti távolság ($m_i - m_j$) nagy pontossággal ismert, még akkor is, ha az egyes pontok abszolút helyzete meglehetősen bizonytalan.

A legfontosabb felfedezés az volt, hogy a tájékozódási pontok közötti becslés korrelációja monoton nő minden újabb és újabb megfigyelés után. Gyakorlatilag ez azt jelenti, hogy a tájékozódási pontok relatív helyzetének az ismerete fokozatosan javul és sosem romlik a robot mozgásától függetlenül. Ezek az eredmények viszont csak a lineáris Gauss esetre bizonyított, az általánosabb valószínűségi eset még nyitott probléma.

A konvergencia oka, hogy a robot által tett megfigyelések a tájékozódási pontok relatív helyzetének közel független méréseinek tekinthetők. Tekintsük meg újra az 2.1. ábrát, ahol láthatjuk, hogy a robot az $\mathbf{x}_k$ pozícióban észleli az m_i és m_j tájékozódási pontokat. A megfigyelt tájékozódási pontok relatív helyzete egyértelműen független a robottól. Egymást követő megfigyelések erről a fix pontról további független méréseket eredményezne a tájékozódási pontok közötti relatív kapcsolatról. Ha a robot továbbmegy az $\mathbf{x}_{k+1}$ helyzetre, itt ismét észleli az m_j tájékozódási pontot. Ez megengedi a robot és a tájékozódási pont becsült helyzetének frissítését az előző $\mathbf{x}_k$ ponthoz képest. Ez viszont visszafele terjedve frissíti az m_i tájékozódási pontot, még abban az esetben is, ha ez a pont nem látható az új helyzetből. Ez amiatt fordul elő, hogy ez a két pont erősen korrelál, azaz a relatív helyük ismert, a korábbi mérésekből, továbbá még korreláltabbá válnak, mivel ugyanazokat a mérési adatokat használják a helyzetük frissítéséhez. A kifejezés közel független mérést indokol, mivel a megfigyelési hibák korrelálnak az egymást követő robot mozgásokon át. Szintén érdemes megjegyezni, hogy az ábrán a robot az $\mathbf{x}_{k+1}$ helyzetben két újabb tájékozódási pontot észlel m_j -hez relatívan. Ezek a pontok így azonnal

kapcsolódnak, vagy korrelálnak a térkép többi részével. Későbbi frissítésük hatására frissíteni fogják m_j -t és ezen keresztül m_i -t és így tovább. Végül minden tájékozódási pont egy hálózattá áll össze, ahol az egyes kapcsolatok a pontok relatív helyzetei, a háló pontossága pedig mindegy egyes új megfigyelés hatására növekedni fog.

Ezt a folyamatot mutatja be a 2.2. ábra, ahol mint rugók hálózata kötődik össze az összes tájékozódási pontot egymással. Egy-egy megfigyelés a szomszédokra olyan, mintha a rugós rendszert elmozdítanánk, melynek hatása a szomszédos pontokra nagy, míg a távolsággal arányosan egyre csökken a távolabbi pontoknál. Ez persze függ a lokális merevség, azaz korreláció tulajdonságaitól. Ahogy a robot keresztülmegy ezen a környezeten és megfigyeléseket végez a tereptárgyakról, a rugók egyre merevebbek lesznek, végül pedig a tájékozódási pontok egy igen merev térképét kapjuk, avagy a környezet pontos relatív térképét. Ahogy a térkép felépült, a robot helyzetének pontosságát csak a térkép és a szenzor méréseinek pontossága korlátozza. Elméletben a robot relatív helyzetének pontossága egyenlő egy adott térképen elérhető lokalizáció pontosságával.



2.2. ábra. Rugó hálózat analógia. [27]

A SLAM probléma megoldására leggyakrabban a kiterjesztett Kalman szűrőn alapuló módszerek, valamint a FastSLAM [27, 28] használatosak.

3. fejezet

Foglaltsági háló alapú térképezés

3.1. Bevezetés

A térképezés az egyik legalapvetőbb probléma a mobil robotikán belül. Egy térkép segítségével egy robot eredményesen elvégezheti feladatát, lokalizálhatja saját magát, útvonalakat tervezhet, melyeket sikeresen végigjárhat. A sikeres robot rendszerek ezekre a térképekre támaszkodnak lokalizáció, útvonaltervezés, cselekvéstervezés során. A legáltalánosabb probléma a térképezésben, hogy környezetünk pontosan hogyan is nézhet ki. Formálisan magában foglalja a szenzor adatok alapján történő legvalószínűbb térkép építését.

Valójában a SLAM-ben ez a probléma hasonlít a csirke és tojás esetéhez, ahol nem lehet eldönteni, hogy melyik volt előbb, mivel térképet építeni az alapján tudunk, hogy ismerjük a robot pontos pozícióját, míg magát a robotot lokalizálni egy már felépített térkép alapján lehetséges.

Térképezési problémák:

- *Szenzor interpretáció*: Hogyan nyerhetünk ki lényeges információkat a nyers szenzor adatokból, valamint hogyan ábrázolhatjuk és egyesíthetjük ezeket az információkat az idő előrehaladtával.
- *Robot helyének becslése*: Hogyan tudjuk felismerni, hogy már egy előzőleg meglátogatott helyen vagyunk. Ezt a problémát nevezzük az úgynevezett adat egyesítési problémának.

Az érzékelők általában zajosak és az általuk szolgáltatott adatok gyakran nem egyértelműek vagy félrevezetőek. Egy folyosón végiggördülő robot valószínűleg bizonytalannak érzi a közte és a fal között lévő távolságot mindkét oldalon, a robot csak találgatni tud, hogy mikor következik egy forduló, vagy az is előfordulhat, hogy emberek állják el az útját. Kitalálni, hogy mikor és hogyan oldjuk meg ezeket a bizonytalanságokat egy nagyon fontos része a robotikának és az irányítás elméletnek. Tekintettel arra, hogy a valószínűség számítást és a statisztikát gyakran használjuk arra, hogy leírjuk egy robot ismereteit az őt körülvevő világról, nem túl meglepő dolog, hogy a matematika eme területei kiemelkedő szerepet játszanak a robotikán belül.

Dolgozatomban egy *foglaltsági háló alapú* térképészeti eljárást valósítottam meg, melynél a 2 dimenziós térképet egy ultrahang szenzor méréseiből megalkotott inverz szenzor modell alapján készítettem el.

3.2. Az ultrahang szenzor

Mielőtt el tudnánk kezdeni a térkép építését, szükségünk van az ultrahang szenzor mérési hibáinak és korlátainak a kiderítésére. Ennek megoldásaképp végeztem el egy kísérletet, egy mérési sorozatot, az A.2. ábrán látható teszt pályán.

A mérés során minden egyes távolságban és fokban 100 mérés átlagát vettem. A mérési eredmények az alábbi táblázatban lettek összefoglalva:

		Szög (fok)								
		-40°	-30°	-20°	-10°	0°	10°	20°	30°	40°
Távolság (mm)	50	70	60	60	60	50	60	60	60	60
	60	70	70	60	60	60	60	60	70	80
	70	80	80	70	70	70	70	70	80	90
	80	110	90	80	80	80	80	80	100	110
	90	150	110	100	90	90	90	100	110	150
	100	190	130	110	100	100	100	110	140	190
	110	190	140	130	120	110	120	130	160	190
	120	190	160	140	130	130	140	140	170	190
	130	200	180	150	140	140	150	160	190	190
	140	190	180	160	150	150	160	170	190	200
	150	190	180	170	160	160	160	170	190	190
	160	190	180	180	170	170	170	180	190	200
	170	190	190	180	180	180	180	180	190	200
	180	190	190	190	180	180	190	190	190	200
	190	200	190	190	190	190	190	190	200	200
200	210	200	200	190	190	200	200	200	210	
210	220	200	200	200	200	200	200	210	220	

3.1. táblázat. Az ultrahang szenzor méréseinek eredménye

3.1. táblázatból is látszik, hogy egy ± 15 fokos tartomány az, ahol a szenzor még érdemben használható adatokat ad.

3.3. A térkép készítése

3.3.1. Megvalósítása

A foglaltsági háló alapú térkép készítésnél a világot egy rácsra képezzük le, amelynek minden pontjához egy valószínűségi értéket párosítunk ([6]). Ez a $[0,1]$ intervallumból vett érték azt adja meg, hogy mekkora valószínűséggel foglalt az adott cella, vagy rácspont. Minél közelebb van az adott érték a 0-hoz, annál valószínűbb, hogy az adott cella üres, illetve minél közelebb van az 1-hez, annál valószínűbb, hogy az a cella foglalt. A cél, hogy megbecsüljük a térkép valószínűségeit a mért távolság-adatok alapján:

$$Pr(m|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}), \quad (3.1)$$

ahol m a térkép, $\mathbf{z}_{1:t}$ a mért távolságadatok az 1. időpillanattól a t -edik időpillanatig, valamint $\mathbf{x}_{1:t}$ a robot pozíciói az 1. időpillanattól a t -edik időpillanatig. A navigáció

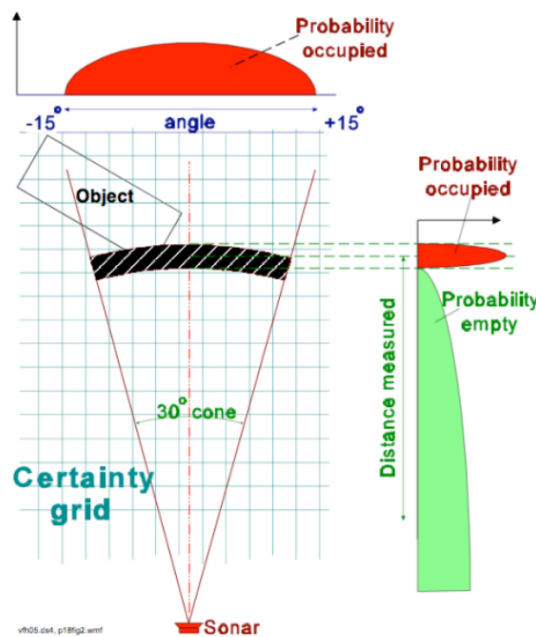
és az odometriai adatok nem játszanak szerepet a térkép készítésénél. A leggyakoribb típus a 2 dimenziós, de található 3 dimenziós térkép reprezentáció is.

A megépített NXT roboton elhelyezett ultrahang szenzor segítségével szereztem be a távolságadatokat. Tapasztalataim, valamint az elvégzett kísérletek szerint ennek szenzorkúpja körülbelül 30° -os, ami azt jelenti, hogy ezen tartományon belül kaphatunk használható, valósághoz közeli, valamint zajjal kevésbé terhelt méréseket, így összesen 12 mérést végez a robot egy teljes (360° -os) szenzor-fordulat alatt.

3.3.2. A mérési modell

Minden teljes fordulatot a szenzoradatok feldolgozása követ. Ahhoz, hogy ebből a 12 távolságadattól térképet lehessen építeni, az úgynevezett inverz szenzor modellt alkalmaztam, mely a 3.1. ábrán látható.

A modell szerint az adott távolsági adatnál közelebb levő cellák alacsonyabb valószínűséget, míg azok a cellák, melyeknek robottól vett távolsága közel megegyezik a kapott távolsági adattal, magasabb valószínűséget kapnak. Ezen kívül a szenzorkúp tengelyéhez közelebb lévő cellák szintén magasabb, míg ettől távolodva arányosan alacsonyabb valószínűségi értéket kapnak.

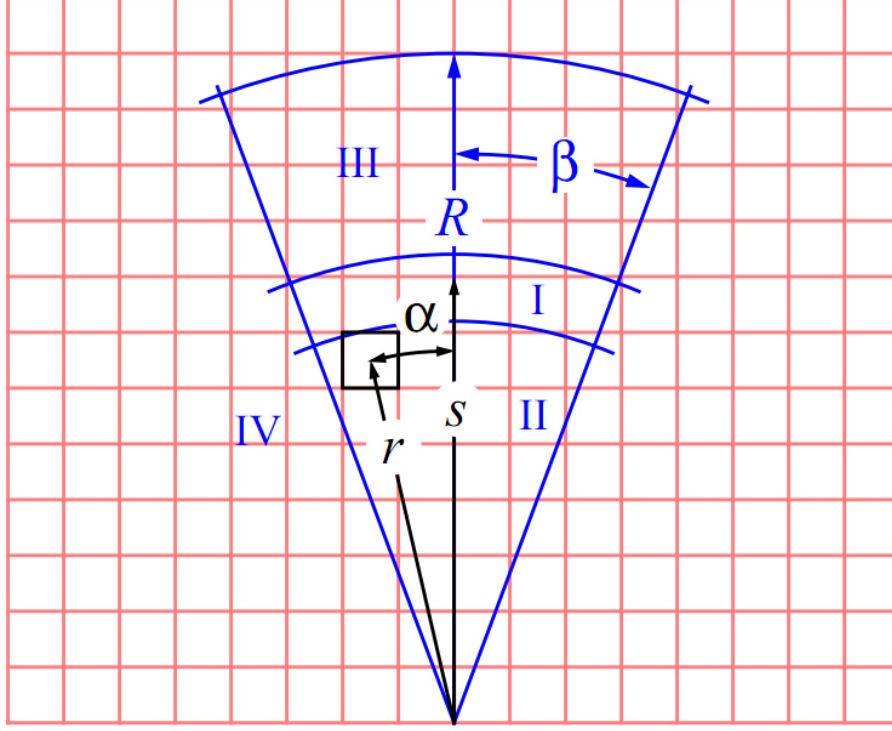


3.1. ábra. Inverz szenzor modell. [40]

Szemléletesebb azonban a 3.2. ábra, ahol

- I: azon cellák összessége, amik magas valószínűséget kapnak.
- II: azon cellákat foglalja magában, amik alacsony valószínűséget tartalmaznak.
- III: azokat a cellákat tartalmazza, amik a megtalált akadály mögött helyezkednek el.
- IV: ezen terület kívül esik a szenzor mérési tartományán.

- β : a szenzorkúp fele.
- R : az ultrahang szenzor érzékelési határa, ami 255 cm.
- S : a kapott távolság adat.
- α : az adott cellának, a szenzorkúp tengelyéhez viszonyított szöge.
- r : az adott cella robothoz viszonyított távolsága.



3.2. ábra. Inverz szenzor modell, a szükséges adatok feltüntetésével. [41]

A teljes térkép elkészítéséhez egy adott cellára a folyamat során több mérést is végzünk, melyek esetében különböző a robot helye, iránya, valamint a zaj mértéke is változó, így egy adott cellára több valószínűségi értéket kellene megadni. Ezekből az egymástól független mérésekből adódik a cellák végleges valószínűsége a folyamat végére. Az egyes értékek kiszámításához a Bayes-tételt alkalmaztam, valamint azt a feltételt, miszerint az egyes mérések egymástól függetlenek. Ezeket az egyes értékeket kell összesíteni a folyamat során az adott cellában már megtalálható értékkel egy végleges valószínűségi értéké.

Az implementáció során minden egyes mérés alkalmával az összes rácsponton végig kell haladni és meg kell vizsgálni, hogy az adott cella mely területbe esik bele, ettől függően eltérő módszerrel számítható ki az adott cella valószínűsége. Ha egy cella az I. területbe esik bele, akkor az alábbi, 3.2. egyenleteket kell alkalmaznunk:

$$Pr(s|Occ_{x,y}) = \frac{\left(\frac{R-r}{R}\right) + \left(\frac{\beta-\alpha}{\beta}\right)}{2} M \quad (3.2)$$

$$Pr(s|\overline{Occ_{x,y}}) = 1 - Pr(Occ_{x,y}),$$

ahol az $Occ_{x,y}$ jelöli azt az eseményt, hogy az (x,y) koordinátájú cella foglalt, míg az $\overline{Occ_{x,y}}$ azt, hogy ez a cella szabad. Ezen kívül $Pr(Occ_{x,y})$ azt a valószínűséget mutatja meg, hogy az adott (x,y) cella mekkora valószínűséggel foglalt, míg a $Pr(\overline{Occ_{x,y}})$ az előbbiekhez hasonlóan azt, hogy mekkora valószínűséggel szabad. Ezen utóbbi valószínűségeket pontosabban a $Pr(s|Occ_{x,y})$ feltételes valószínűséggel kell leírni, ami megadja a szenzor mérés valószínűségét, feltéve, ha az adott (x,y) cella foglalt. Az itt megjelenő M egy konstans, melynek általam beállított értéke 0,98. Ha azonban a soron következő cella a II. területbe esne bele, a 3.3. egyenletek lépnek életbe:

$$\begin{aligned} Pr(s|Occ_{x,y}) &= 1 - Pr(\overline{Occ_{x,y}}) \\ Pr(s|\overline{Occ_{x,y}}) &= \frac{(\frac{R-r}{R}) + (\frac{\beta-\alpha}{\beta})}{2} \end{aligned} \quad (3.3)$$

Sajnos jelenleg csak $Pr(s|Occ_{x,y})$ -t és $Pr(Occ_{x,y})$ -t ismerjük. Ahhoz, hogy megkapjuk azt amire szükségünk van, alkalmaznunk kell a Bayes-szabályt, mely szerint:

$$Pr(H|d) = \frac{Pr(d|H)Pr(H)}{Pr(d)} = \frac{Pr(d|H)Pr(H)}{Pr(d|H)Pr(H) + Pr(d|\neg H)Pr(\neg H)}, \quad (3.4)$$

ahol H egy hipotézist jelöl, d pedig egy adatot vagy bizonyítékot. A 3.4. kifejezésnek egy foglaltsági hálóra alapozott alkalmazása a következő:

$$Pr(Occ_{x,y}|s) = \frac{Pr(s|Occ_{x,y})Pr(Occ_{x,y})}{Pr(s|Occ_{x,y})Pr(Occ_{x,y}) + Pr(s|\overline{Occ_{x,y}})Pr(\overline{Occ_{x,y}})}, \quad (3.5)$$

így megkapjuk $Pr(Occ_{x,y}|s)$ -t, azaz annak a valószínűségét, hogy egy adott (x,y) koordinátájú cella foglalt, feltéve, hogy az adott s szenzor mérést végeztük.

Mivel minden cella végleges valószínűségi értéke egy sor független mérésből adódik, így minden mérés alkalmával, ha megkaptuk a fenti egyenletből a cella valószínűségi értékét a jelenlegi mérésből adódóan, akkor azt integrálni kell a cella már meglévő valószínűségi értékével, ami az előző mérésekből következett. Ezt megtehetjük az alábbi (3.6.) egyenlet segítségével:

$$1 - \left(1 + \frac{Pr(Occ_{x,y}|s_1)}{1 - Pr(Occ_{x,y}|s_1)} \prod_{t=2}^T \left(\frac{Pr(Occ_{x,y}|s_t)}{1 - Pr(Occ_{x,y}|s_t)} \frac{1 - Pr(Occ_{x,y})}{Pr(Occ_{x,y})} \right) \right)^{-1} \quad (3.6)$$

Ezzel az egyenlettel minden egyes új, az előzőektől független mérés az előzőekhez illeszthető iteratív módon. Ily módon a folyamat végére megkapjuk az adott cella végleges valószínűségi értékét. Ennek a módszernek egy igen nagy hátránya, hogy a rossz visszaverődésekből származó távolsági adatok is összegződnek az addigi pontos mérésekből származó adatokkal.

Ezen egyenlet implementálása során két változót alkalmaztam:

- $Pr_{(x,y)}$: az aktuális iterációs lépésben, az adott (x,y) cella valószínűségét tartalmazza.
- $f_{(x,y)}$: segédváltozó, mely a soron következő iterációs lépésben segít az adott (x,y) cella valószínűségét kiszámolni.

A 3.6. egyenletből látható, hogy a térképezés során a legelső iterációs lépésben ezen egyenletnek csak egy részét használhatjuk a valószínűség kiszámítására. Ebben a lépésben a két változó értéke a következő módon adódik:

$$\begin{aligned} Pr_{(x,y),1} &= 1 - \left(1 + \frac{Pr(Occ_{x,y}|s_1)}{1-Pr(Occ_{x,y}|s_1)} \right)^{-1} \\ f_{(x,y),1} &= \frac{Pr(Occ_{x,y}|s_1)}{1-Pr(Occ_{x,y}|s_1)} \end{aligned} \quad (3.7)$$

Ezután minden soron következő iterációs lépésben a változók értékei az alábbi módon számolhatóak:

$$\begin{aligned} Pr_{(x,y),i} &= 1 - \left(1 + f_{(x,y),i-1} \left(\frac{Pr(Occ_{x,y}|s_t)}{1-Pr(Occ_{x,y}|s_t)} \frac{1-Pr(Occ_{x,y})}{Pr(Occ_{x,y})} \right) \right)^{-1}, \\ f_{(x,y),i} &= \left(\frac{Pr(Occ_{x,y}|s_t)}{1-Pr(Occ_{x,y}|s_t)} \frac{1-Pr(Occ_{x,y})}{Pr(Occ_{x,y})} \right) f_{(x,y),i-1} \end{aligned} \quad (3.8)$$

ahol i az i -edik iterációs lépést jelöli.

3.3.3. Felmerülő problémák és megoldásuk

Problémák adódtak a térkép építése során a PC és a robot közötti szinkronizációs gondokból. Ezt úgy oldottam meg, hogy a robot csak akkor küldhet adatot a PC-nek, ha utóbbi azt egy szinkronizációs jellel engedi, valamint az adatküldés végét is egy bizonyos szinkronizációs jel jelzi a roboton futó programnak.

Egy pontos térkép készítéséhez elengedhetetlen, hogy a robot egy adott pozícióból egynél több mérést vegyen a környezetéről, jelen esetben körülbelül 12 mérés már kielégítő. Az alap Mindstorms készlet viszont csak egyetlen egy ultrahang szenzort tartalmaz, tehát a robot felépítésénél kellett megoldani a szenzor forgathatóságát. Ebből a forgatásból adódóan pedig minden forgatási lépésben enyhe rázkódás tapasztalható a szenzornál, így bekerült egy kódrészlet, ami késlelteti egy másodperccel az ultrahang mérést a pontosabb vétel végett. Ezen kívül pedig mind a 12 mérés folyamán szinkronizálni kellett a szenzorkúp tengelyét a robot mindenkor irányához, hogy az összes mérés a megfelelő irányban jelenjen meg a térképen és minden esetben a megfelelő cellák valószínűségét változtassuk meg.

A térkép 2 dimenziós voltából adódóan minden cella más és más irányban helyezkedik el a robot mindenkor irányához képest. Ezt ugyancsak szinkronizálni kellett a robot irányához és a szenzorkúp tengelyéhez, hogy az egyes cellák a megfelelő területekbe essenek bele.

A 3.6. egyenlet két igen nagy hibával rendelkezik, melyet szükséges volt orvosolni a működőképes térképezés elérése végett.

- 1 *Hiba*: A térképezés bármelyik szakaszában előfordulhat, hogy a robot egy adott falszakaszt még nem érzékel, de annak helyére sorozatosan *üres* méréseket végez, így az egyenlet rekurzív tulajdonságából kifolyólag ezen cellákban az $f_{(x,y),i}$ értéke drasztikusan lecsökken egészen megközelítve a 0-t. A robot bizonyos számú mérés elteltével ezt már teljesen 0-nak tekinti, így ha egy *fal* mérés következne, a 0-val való szorzás következtében a $Pr_{(x,y),i}$ értéke is 0 lesz.

Megoldás: Minden egyes *fal* mérés alatt meg kell vizsgálnunk az $f_{(x,y),i-1}$ értéket. Ha ez egy előre meghatározott szint alá csökkenne, jelen dolgozat esetében ez a szint 0,1, akkor visszaállítjuk azt a megadott szintre, kiküszöbölve ezzel a 0-val való szorzás veszélyét.

- 2 *Hiba:* Egy adott falhoz közel a robot erről a falról megfelelő pontossággal végez méréseket, viszont ettől eltávolodva már pontatlan mérések is előfordulnak ugyanerre a falra. Ugyanazt az adott akadályt már távolabbinak érzékeli, mint az valójában megtalálható. Mivel ezen mérések alatt az adott magas valószínűségű cellák *üres* területbe esnek bele, a fenti egyenletből következőleg így elkezd lecsökkenni ezen cellák valószínűsége, gyakran egészen 0 közeli értékig.

Megoldás: Minden *üres* mérés alkalmával meg kell vizsgálnunk az adott cella $Pr_{(x,y),i-1}$ valószínűségét. Ha ez egy előre meghatározott szint fölött található, jelen dolgozat esetében ez a szint 0,9, akkor csak kisebb mértékben csökkentjük ezt a valószínűséget.

Ezen kívül megfigyeltem, hogy a 3.6. egyenlet lényegesen gyorsabban csökkenti egy cella valószínűségét, mint amilyen nagy mértékben emelné azt.

4. fejezet

Navigáció

4.1. Odometrián alapuló navigáció

Az odometrián alapuló navigáció a legszélesebb körben alkalmazott navigációs módszer egy robot pozíciójának meghatározására, mivel elfogadható rövid távú pontosságot biztosít, megvalósítása olcsó és magas mintavételi frekvenciát tesz lehetővé. Ebben az esetben nem a robotot körülvevő környezetből vesszük a szükséges információkat a szenzorainkkal ahhoz, hogy tudjuk pontosan hol vagyunk, hanem (mint jelen dolgozat esetében) egy adott kiindulási hely és irány, valamint a kerekek által megtett szögsebesség és sebesség segítségével határozzuk meg a robot jelenlegi helyzetét. Ehhez viszont ismernünk kell a kerekek sugarát, illetve a két kerék közötti távolságot. Az alapvető gondolata az odometriának, hogy a fokozatos mozgási információkat integrálja az idő előrehaladtával, ami viszont elkerülhetetlenül a hibák felhalmozódásához vezet. Különösen a robot fordulásánál keletkező orientációs hibák vezetnek pozíció hibákhoz [7], amiket arányosan növel a robot által megtett út is. Ennek ellenére a legtöbb kutató egyetért abban, hogy az odometria fontos része a robot navigációs rendszereknek, illetve, hogy a navigációs feladatok egyszerűbbé fognak válni, ha javítani tudjuk az odometria pontosságát.

Mielőtt megpróbáljuk mérni és csökkenteni az odometrikus hibákat, nagyon fontos dolog, hogy megértsük a szisztematikus és nem-szisztematikus odometriai hibák közötti különbséget. A szisztematikus hibák azok a hibák, amik eleve részei a robot kinematikájának, vagy vezérlő tulajdonságainak, függetlenül a robot környezetétől. A nem-szisztematikus hibák viszont függenek a robot környezetétől és különbözőek egyik és másik környezetben is. Azért fontos különbséget tennünk e kettő között, mivel mindkét csoport különböző módon hat a mobil platformokra, kiküszöbölésük különböző, valamint ami a legfontosabb, hogy mindkét csoport különböző mérési technikákat követel meg. A következőképp csoportosíthatjuk a hibákat:

Szisztematikus hibák

- Egyenlőtlen kerékátmérők.
- A tényleges kerékátmérők átlaga eltér a névleges kerékátmérők átlagától.
- A kerekek eltolódása, azaz a kerekek tényleges távolsága nem egyezik meg a névleges távolsággal.
- Bizonytalanság a tényleges tengelytávolsággal kapcsolatban, mivel nem egy ponton érintkeznek a kerekek a talajjal.

- A kerekek nem egy vonalban helyezkednek el.
- A motorok valódi teljesítménye eltérő a névlegestől.
- Szenzoradatok kvantálási hibája, alacsony mintavételi frekvenciából származó hiba.

Nem-szisztematikus hibák

- Mozgás egyenetlen padlón.
- A mozgás során váratlan tárgyak a padlón.
- Kerék csúszás, például csúszós padló, túlzott gyorsítás vagy egy gyors fordulat miatt bekövetkező megcsúszás következtében.
- Nem pontszerű kapcsolat a talajjal.
- Külső erők hatása, például nekimegy valaminek.
- Belső erők hatása, például beragad valamelyik kerék.

A kerekek csúszása jellemzően akkor fordul elő, ha a robot egy egyenes területen egy vizes felületen halad keresztül és a kerekek kipörögnek, vagy ha egy dombon felfele menet megáll, ezután pedig lecsúszik onnan. Előbbi esetben a robot a kerék-szenzorok információja alapján azt érzékeli, hogy halad miközben előfordulhat, hogy a kerekek kipörgéséből következőleg valójában áll, míg utóbbi esetben a leblokkolt kerekek miatt robot azt érzékeli, hogy áll, viszont a megcsúszásból adódóan valójában már nem ott található meg, ahol ő azt kiszámította.

4.2. Költségmátrixon alapuló navigáció

A 3.3.1. fejezetben bemutatott eljárás segítségével immár a robotunk képes megalkotni egy 2 dimenziós térképet környezetéről. Mindez viszont mit sem ér, ha nem tudja helyét megváltoztatni, hogy más pozíciókból is információkat szerezhesen méréseiből. Ennek megoldására implementáltam egy költségmátrixon alapuló navigációs eljárást, ami a felépített valószínűségi térkép alapján kiszámolja a következő mérés legoptimálisabb helyét, megtervezi az oda vezető utat, majd be is járja azt az elékerülő akadályokat kikerülve.

4.2.1. A mátrix megvalósítása

Ez az eljárás a Sutton és Barto könyvében is bemutatott értékiteráció Thrun által létrehozott foglaltsági hálóra alapozott verziójának általam módosított változata [8], [9]. Számtalan változatában eltérő módon számítják ki a költségmátrix értékeit. Az általam létrehozott verzió, mely a 4.1. kifejezésben látható, tulajdonképpen a térkép celláinak értéke alapján felépít egy költségmátrixot, a számítás során pedig kiértékeli, hogy az egyes cellák milyen messze találhatóak a robot jelenlegi helyzetétől.

Az eljárásban azok a cellák, amik valószínűsíthetőleg foglaltak 1-es értéket, míg azok amiket már bejártunk pedig egy épp elegendően nagy értéket, ezen dolgozat esetében 1000-et, kapnak. Azon cellák pedig amik még potenciális információforrásnak számítanak, azaz még nem jártuk be őket és nem foglaltak, egy kalkulált értéket kapnak, ami a szomszédos cellák valószínűségének és egy távolságnak a minimumából adódik. A távolság az adott cella és a robot jelenlegi helyének a távolsága között

azzal arányosan nő, minél távolabb helyezkedik el a robottól az adott cella. Ezt az eljárást mutatja be az alábbi egyenlet:

$\forall$ bejáratlan (x,y) -re:

$$V_{x,y} = \begin{cases} 1 & \text{ha } Pr(Occ_{x,y}) > Pr(Occ) \\ \min(1, \min_{\substack{i=-1,0,1 \\ j=-1,0,1}} \{Pr(Occ_{x+i,y+j}) + \delta\}) & \text{egyébként} \end{cases} \quad (4.1)$$

A robot a mátrixban található legkisebb értékű cellák fele fog folyamatosan lépegetni, így biztosítható a kicsapongások nélküli megtervezett mozgás. Ha megtaláltuk a legkisebb értékű cellát, akkor a robot az elkövetkezendőkben ismerttetett A^* kereséssel építi fel az oda vezető utat.

4.2.2. Felmerülő problémák és megoldásuk

Mivel a robot nem egy kiterjedés nélküli pont, így a költségmátrix számításánál figyelembe kell venni annak méreteit és kialakítani egy tiltott zónát a falak körül, hogy a robot elakadásmentesen tudjon közlekedni a falak mentén is. Ezt persze bármilyen felbontású térképre tudni kell alkalmazni.

Ezen kívül lehetséges hogy egynél több cella is tartalmazza a mátrixban előforduló legkisebb értéket, így egy kódrészlet közbeiktatásával szinkronizálom az adott cellák szögét a robot mindenkor irányához, valamint kiválasztom a robot jelenlegi irányához legközelebb eső cellát, hogy ezzel is lecsökkentssem a fentebb is említett feleslegesen nagy fordulásokból eredő odometriai hibákat.

4.3. Útkeresés

A 4.2. fejezetben kifejtett költségmátrix segítségével sikeresen megtaláltuk a cél cellát ahova a robotnak lépni kell a következő iterációban. Következő lépésként meg kell találnunk azokat a cellákat, amik a legrövidebb útvonalon helyezkednek el, valamint amiken végighaladva eléri ezt a cél cellát. Ennek megoldására implementáltam egy A^* kereső algoritmust, melyet a következőkben fogok ismertetni.

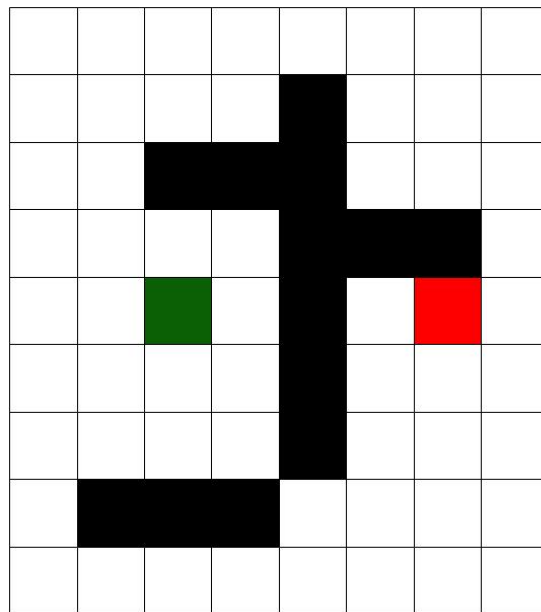
4.3.1. A^* algoritmus

A mesterséges intelligenciában ismert *mohó keresés* az adott cellából a cél cellába vezető út becslőjét minimalizálja, ezzel csökkentve a keresés költségét, viszont így nem optimális és nem teljes ez az algoritmus. Az *egyenletes költségű keresés* ezzel szemben minimalizálja az eddig megtett út költségét, aminek hatására optimális és teljes, viszont rossz hatékonyságú lesz. Ennek a két algoritmusnak a pozitív tulajdonságát ötvözzük az A^* keresésben [10].

Tételezzük fel, hogy el akarunk jutni A pontból B pontba. Tegyük fel, hogy ezt a két pontot egy fal választja el egymástól. Ezt mutatja be a 4.1. ábra, ahol zöld színnel jelöljük a kiindulási A pontot, piros színnel a B végpontot, míg feketével azokat a cellákat amiken fal található.

Az első lépés az út keresése a kiindulási és a végpont között. Ezt úgy tesszük meg, hogy a kiindulási ponttól indulva megvizsgáljuk annak szomszédjait, ezt pedig addig tesszük míg meg nem találjuk a végpontot. A következőképp indul a keresési algoritmus:

1. A kiindulási A pont felvétele egy nyílt listára, ami kezdetben még csak ezt az egy pontot tartalmazza, viszont ez a folyamat előrehaladtával változni fog. Ez a nyílt lista tartalmazza azokat a cellákat, amik benne lehetnek a végső útvonalban.
2. A kiindulási cella szomszédainak felvétele a nyílt listára és az úgynevezett szülő cellájuk megjelölése. Ez később fog nagy jelentőséggel bírni, mikor a folyamat végén a legrövidebb út visszakeresése történik majd.
3. A kiindulási cella kitörlése a nyílt listáról és felvétele egy úgynevezett zárt listába.



4.1. ábra. Az A* algoritmus kindulási állapota.

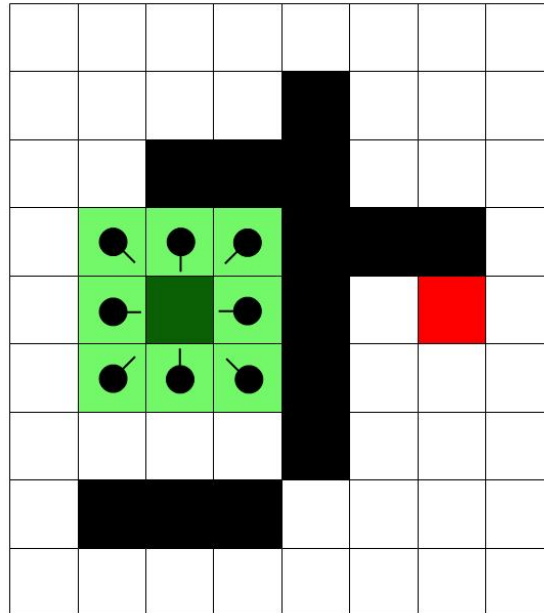
Az algoritmus folytatásaként ki kell számolni minden cella F értékét, ami megtalálható a nyílt listában. Ezt az alábbi, 4.2. egyenlet alapján tehetjük meg:

$$F = G + H, \quad (4.2)$$

ahol:

- G : A mozgás költsége a kiindulási A pontból az adott celláig. Jelen dolgozat esetén az ortogonális mozgást "10", míg a diagonális mozgást "1000" értékre állítottam be, hogy kellő mértékben büntessem a robot ilyen irányú mozgását, ezzel elérve, hogy a megtalált legrövidebb útban csak ortogonális lépéseket kapjunk.
- H : A becsült mozgási költség az adott cellától a B végpontig. Ezt gyakran egy heurisztikus becslésnek hívják, mivel nem tudjuk pontosan a valódi távolságot addig, míg meg nem találjuk az utat, mivel bármilyen akadály lehet az útban.

Jelen dolgozat esetén minden cellalépés "10" értéket jelent. Kizárólag ortogonális lépéseket használtam a H érték kiszámolásához, a *Manhattan távolság*¹ alapján.



4.2. ábra. Az A* algoritmus folytatása.

Ezeket az F értékeket az összes nyílt listában lévő cellára kiszámolva megkapjuk a legkisebb értékűt. A 4.2. ábrán látható az algoritmus működés közben. Folytatva az algoritmust és kiválasztva ezt a legkisebb F értékűt, a következő lépések jönnek:

4. Az adott cella törlése a nyílt listából, ezzel egyidőben felvétele a zárt listába.
5. A cella szomszédainak vizsgálata, figyelmen kívül hagyva azokat, amelyek elemei a zárt listának, vagy elérhetetlenek. Ezen szomszédos cellák felvétele a nyílt listába, feltéve, hogy még nem elemei annak. A kiválasztott cella megjelenése szülőként az újonnan felvett cellákban.
6. Ha egy szomszédos cella már eleme a nyílt listának, megvizsgáljuk a G értékét. Ha a jelenlegi szülő cellából ebbe a cellába vezető út költsége kisebb, akkor ennek a szomszédos cellának a szülőjét átállítjuk a legutóbbi szülő cellára, majd az F érték újraszámoljuk.

Az algoritmus addig ismétli a fentebb leírt lépéseket míg:

- a cél cella be nem kerül a zárt listába.
- a nyílt lista elemei el nem fogynak úgy, hogy a zárt lista nem tartalmazza a cél cellát. Ebben az esetben nincs út a két pont között.

¹A New York-i Manhattan kerület hálóra emlékeztető utcai földrajzán alapul, ahol két pont közötti távolság szigorúan horizontálisan és/vagy vertikálisan megengedett lépésekből adódik (azaz a rácsvonalak mentén), a diagonális (légvonalban értett) lépésekkel ellentétben. Maga a távolság ezen vízszintes és függőleges lépések egyszerű összege.

Az algoritmus működésének bemutatása az alábbi pszeudokód segítségével:

```

A* ALGORITMUS(kiindulasiCella, celCella)
1  NYILTLISTAHOZ(kiindulasiCella)
2  while igaz
3      do if hossz[nyiltLista] = 0
4          then return hamis
5          minCella ← MINIMUMF(nyiltLista)
6          HOZZAADASZARTLISTAHOZ(minCella)
7          TORLESNYILTLISTABOL(minCella)
8          if minCella = celCella
9              then return igaz
10         szLista ← SZOMSZEDOK(minCella)
11         for i ← 0 to hossz[szLista]
12             do SZULO(szLista[i], minCella)
13             for i ← 0 to hossz[szLista]
14                 do if ZARTLISTABAN(szLista[i]) = hamis
15                     then if NYILTLISTABAN(szLista[i]) = hamis
16                         then if TILTOTT(szLista[i]) = hamis
17                             then FSZAMOL(szLista[i], celCella)
18                                 SZULO(szLista[i], minCella)
19                                 NYILTLISTAHOZ(szLista[i])
20                             else ujG ← GSZAMOL(szLista[i], minCella)
21                                 if ujG < GERTEK(sz[i])
22                                     then GSZAMOL(szLista[i], minCella)
23                                     FSZAMOL(szLista[i], celCella)
24                                     SZULO(szLista[i], minCella)

```

Abban az esetben viszont, ha megtaláltuk a cél cellát, egyszerűen ettől a cellától kezdve vissza kell lépkedni a szülő jelzések segítségével a kiindulási celláig és meg is van a legrövidebb utunk a két pont között. Megkezdődhet a robot mozgása a cél cella felé.

5. fejezet

Lokalizáció

5.1. Bevezetés

A térkép építése és a navigáció megvalósítása után a következő felmerülő probléma, ahol a robot helyzetét kell megtalálni és követni, a lokalizáció, ami lehet lokális vagy globális. Az elsőként bemutatott, és még manapság is számos projekt során alkalmazott módszer a probléma megoldására az odometria, amit már korábban említettünk a 4.1. fejezetben. Ez a mobil robot helyzetének rekonstrukciója, azaz pozíciója és orientációja a kerekek méréseinek segítségével. Kezdve egy ismert pozícióból, a robot jelenlegi pozíciója és orientációja megkapható a kerekek általi szögsebességből és a robot sebességéből [29]. Előnye a szenzorok alacsony költsége, jó pontossága rövid távon, valamint más pozicionáló módszerekkel való kompatibilitása.

Dolgozatomban egy kétkerekű differenciál-hajtású, azaz két hajtott és egy támasztó kerékkel rendelkező, mobil robot pozíció követését valósítottam meg a kerekek odometriai kalibrációjának és az ultrahang szenzor mérési adatainak egyesítésével létrehozott bővített kiterjesztett Kalman szűrő segítségével. Az odometriai részben a pozíció becslése főként odometriai hibákkal terhelt. Ezek a hibák a 4.1. fejezetben bemutatásra kerültek.

5.2. Bővített kiterjesztett Kalman szűrő

5.2.1. Kalman szűrő

A matematikai eszközök egy jelentős eszköztárában, ami képes sztochasztikus becslésekre zajos szenzor mérésekből, az egyik legismertebb és leggyakrabban alkalmazott eszköz a *Kalman szűrő* [30], mely a zajos szenzor mérések valódi értékére és a hozzájuk tartozó számított adatok várható értékére ad becsléseket. Ezen kívül becsli a várható érték bizonytalanságát, valamint a mért és várható érték alapján kiszámol egy súlyozott átlagot. Ezzel a módszerrel előállított becslések közelebb állnak a valódi értékekhez, mint az eredeti mérések, mivel a súlyozott átlag jobb becsült bizonytalansággal rendelkezik, mint bármelyik érték.

A szűrőt Rudolf E. Kálmánról nevezték el, habár Thorvald Nicolai Thiele [31, 32] és Peter Swerling kifejlesztett egy hasonló algoritmust korábban. A Southern California-i egyetemről Richard S. Bucy nagyban hozzájárult az elméleti háttéréhez, ezért nevezik gyakran ezt Kalman-Bucy szűrőnek. A szűrő legelső implementálását pedig gyakran Stanley F. Schmidt nevéhez fűzik, ami még az egyik Kalman-nál

tett látogatásakor történt a NASA Ames Research Center-nél, mikor felismerte ezen módszer alkalmazhatóságát az Apollo-program pálya becslési problémájában, ami végül az Apollo navigációs számítógépébe is beépítésre került.

A Kalman szűrők ezek mellett nagy jelentőséggel bírnak a U.S. Navy¹ nukleáris ballisztikus rakétahordozó tengeralattjárók navigációs rendszereinek implementációiban, valamint a cirkálórakéták irányítási és navigációs rendszereiben, mint például a U.S. Navy Tomahawk rakétaiban és a U.S. Air Force² levegőben indítható cirkálórakétaiban. Ugyancsak használható a NASA³ űrsiklóinak, valamint a Nemzetközi Űrállomás irányító és navigációs rendszereinél.

A Kalman szűrő lineáris dinamikus rendszerek időbeli diszkretizáltján alapszik. *Markov-lánc* mintájára modellezték, gauss, azaz normál eloszlású zajokkal terhelve, valamint a rendszer állapota valós számokból álló vektorral reprezentált.

Annak érdekében, hogy a Kalman szűrő megbecsülje egy adott folyamat belső állapotát, csak zajos mérések sorozatára és a folyamat modelljére van szükség, ami összhangban van a szűrővel. Ez azt jelenti, hogy meg kell határoznunk az alábbi mátrixokat:

- $\mathbf{A}_k$: állapot-átmeneti modell
- $\mathbf{H}_k$: megfigyelési modell
- $\mathbf{Q}_k$: az adott folyamat zajának kovarianciája
- $\mathbf{R}_k$: a megfigyelési zaj kovarianciájára
- $\mathbf{B}_k$: a bemeneti modell

minden k -edik lépésben, az alábbiak szerint:

A Kalman szűrő feltételezi, hogy a valódi állapot a k -edik időpillanatban a $k - 1$ -edik állapot alapján alakult. Tehát:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k u_k + \mathbf{w}_k, \quad (5.1)$$

ahol:

- $\mathbf{A}_k$: az állapot-átmeneti modell, melyet az előző $\mathbf{x}_{k-1}$ állapottal alkalmazzuk.
- $\mathbf{B}_k$: a bemeneti modell, melyet az u_k bemeneti vektorral alkalmazzuk.
- $\mathbf{w}_k$: az adott folyamat nulla várható értékű, többváltozós normál eloszlású, $\mathbf{Q}_k$ kovariancia mátrixú fehér zaja, azaz $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$

A k -edik időpillanatban tett $\mathbf{z}_k$ megfigyelés, vagy mérés, mely a valós $\mathbf{x}_k$ alapján jön létre:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \quad (5.2)$$

ahol

- $\mathbf{H}_k$: a megfigyelési modell, mely összeveti a valós állapot teret a megfigyelt térrel.
- $\mathbf{v}_k$: a megfigyelés nulla várható értékű, normál (Gauss) eloszlású, $\mathbf{R}_k$ kovariancia mátrixú fehér zaja, azaz $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$

¹Az Amerikai Egyesült Államok Haditengerészete

²Az Amerikai Egyesült Államok Légierője

³Az Amerikai Egyesült Államok Nemzeti Légügyi és Űrhajózási Hivatala

A Kalman szűrő egy rekurzív becslő, ami azt jelenti, hogy csupán az előző időpillanatbeli becsült állapot és a jelenlegi mérés szükséges az adott állapotbecslés kiszámításához. Minden iterációban két fázis játszódik le:

Predikciós szakasz

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{A}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k u_k \\ \mathbf{P}_{k|k-1} &= \mathbf{A}_k \mathbf{P}_{k-1|k-1} \mathbf{A}_k^T + \mathbf{Q}_k,\end{aligned}\tag{5.3}$$

ahol sorban az egyenletek:

- Az előzetes⁴ állapot becslés
- Az előzetes becslés kovarianciája

Korrekción szakasz

$$\begin{aligned}\tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{S}_k &= \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1},\end{aligned}\tag{5.4}$$

ahol sorban az egyenletek:

- Az innováció
- Az innováció kovarianciája
- Az optimális Kalman erősítés
- A frissített⁵ állapot becslés
- A frissített becslés kovarianciája

A formula a frissített becslés és kovariancia fölött, csak az optimális Kalman erősítés esetében érvényes. Más erősítési értékek használata esetében bonyolultabb képletet kell használni.

5.2.2. Kiterjesztett Kalman szűrő

A Kalman szűrő azzal az általános problémával foglalkozik, hogy megpróbálja megbecsülni egy lineáris sztochasztikus differencia egyenletekkel leírt diszkrét idejű rendszer $\mathbf{x} \in R^n$ állapotát. Ugyanazon ötlet kiterjesztéseként a kiterjesztett Kalman szűrőt nem-lineáris dinamikus rendszereknél alkalmazzuk. Ezen szűrő tulajdonképpen a jelenlegi becslési hiba középértékének és kovarianciájának linearizációján alapul.

Tételezzük fel, hogy adott egy folyamat, mely rendelkezik egy állapot vektorral $\mathbf{x} \in R^n$ és egy $\mathbf{u}$ bemenőjellel, valamint nem-lineáris sztochasztikus differencia egyenletek által szabályozott:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k, w_{k-1})\tag{5.5}$$

és egy $\mathbf{z} \in R^m$ méréssel:

$$\mathbf{z}_k = h(\mathbf{x}_k, v_k),\tag{5.6}$$

⁴a priori

⁵a posteriori

ahol a két véletlen változó w_k és v_k reprezentálják a folyamat- és a mérési zajt, valamint azt feltételezzük róluk, hogy függetlenek egymástól, $\mathbf{Q}$ és $\mathbf{R}$ kovariancia mátrixú, normál eloszlású fehér zajok. Bizonyítható, hogy a kiterjesztett Kalman szűrő predikciós (time update) egyenletei a következők:

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0) \\ \mathbf{P}_k^- &= \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^T\end{aligned}\quad (5.7)$$

ahol $\hat{\mathbf{x}}_k^-$ a priori (előzetes) állapot becslés. Ezek a predikciós egyenletek megadják az állapot és a kovariancia becslését a jelenleg k -adik időpillanatban, az előző $k-1$ -edik időpillanattól kifolyólag. Tulajdonképpen ez az a rész, ahol az odometria adatok alapján kiszámoljuk egy előzetes becslését a robot jelenlegi pozíciójának, majd ezt az előzetes becslést fogjuk frissíteni az alábbi korrekciós (measurement update) egyenletek segítségével:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T)^{-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0)) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-, \end{aligned}\quad (5.8)$$

ahol $\mathbf{A}$, $\mathbf{W}$, $\mathbf{H}$ és $\mathbf{V}$ Jakobi mátrixok, $\mathbf{K}$ a Kalman erősítés vektora. Ezek a korrekciós egyenletek javítják az állapot és a kovariancia becslését a z_k mérés segítségével. A 5.7. és 5.8. egyenletrendszerben megtalálható Jakobi mátrixokat az alábbiak szerint kapjuk:

- Az $\mathbf{A}$ mátrix az f parciális deriváltjainak, $\mathbf{x}$ függvényében vett Jakobi mátrixa:

$$\mathbf{A}_{[i,j]} = \frac{\partial f_{[i]}}{\partial \mathbf{x}_{[j]}}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, 0), \quad (5.9)$$

- A $\mathbf{W}$ mátrix az f parciális deriváltjainak, $\mathbf{w}$ függvényében vett Jakobi mátrixa:

$$\mathbf{W}_{[i,j]} = \frac{\partial f_{[i]}}{\partial \mathbf{w}_{[j]}}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, 0), \quad (5.10)$$

- A $\mathbf{H}$ mátrix az h parciális deriváltjainak, $\mathbf{x}$ függvényében vett Jakobi mátrixa:

$$\mathbf{H}_{[i,j]} = \frac{\partial h_{[i]}}{\partial \mathbf{x}_{[j]}}(\hat{\mathbf{x}}_k, 0), \quad (5.11)$$

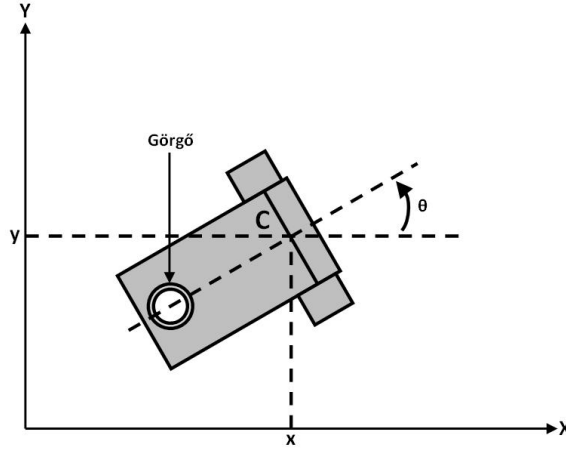
- A $\mathbf{V}$ mátrix az h parciális deriváltjainak, $\mathbf{v}$ függvényében vett Jakobi mátrixa:

$$\mathbf{V}_{[i,j]} = \frac{\partial h_{[i]}}{\partial \mathbf{v}_{[j]}}(\hat{\mathbf{x}}_k, 0), \quad (5.12)$$

A 5.2.3. fejezetben a szűrő megvalósításának magyarázata következik.

5.2.3. Rendszermodell

Egy mechanikai rendszer helyzetét és sebességét írja le a kinematika, a tömegére és tehetetlenségére való hivatkozás nélkül. Tekintsünk a 5.1. ábrára, ahol láthatjuk egy differenciál-meghajtású mobil robot működését sík felületen. Az x és y jelzi a robot két meghajtott kerekének és a robot tengelyének metszéspontjának helyzetét a globális koordináta rendszerben, melyet X -el és Y -al jelölünk, míg θ -val jelöljük a robot irányát az X tengelyhez viszonyítva.



5.1. ábra. Differenciál-hajtású mobil robot kinematikája.

Definiáljuk mindennek előtt a 5.1. ábrából kifolyólag a robot $\mathbf{x}$ konfigurációját:

$$\mathbf{x} = [x, y, \theta]^T \quad (5.13)$$

A két fő kerék külön motorral hajtott, míg a harmadik egy bolygó kerék, amely nem jelent kinematikai korlátozást. A robot mozgását a 5.14. pontban látható differenciálegyenlet-rendszer írja le:

$$\begin{aligned} \dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta, \\ \dot{\theta} &= \omega \end{aligned} \quad (5.14)$$

ahol v a robot abszolút sebessége, ω pedig a szögsebesség a bemenetek függvényében.

A bal és a jobb kerék mozgása leírható az egyes kerekek szögsebességével, ω_L (bal kerék szögsebessége) és ω_R (jobb kerék szögsebessége), míg a v és ω ezektől a szögsebességektől függenek a következők szerint:

$$\begin{aligned} v &= \frac{\gamma_R}{2} \omega_R + \frac{\gamma_L}{2} \omega_L \\ \omega &= \frac{\gamma_R}{d} \omega_R - \frac{\gamma_L}{d} \omega_L, \end{aligned} \quad (5.15)$$

ahol d jelöli a két kerék közötti távolságot, γ_L és γ_R pedig a bal és a jobb kerék sugarát.

A rendszer diszkrétidejű modellje levezethető a 5.14-ban bemutatott differenciál-egyenletekből, így az $\mathbf{x}$ állapot vektor dinamikai egyenlete a következő:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_k + \begin{bmatrix} T v_k \cos(\theta_k + T \omega_k / 2) \\ T v_k \sin(\theta_k + T \omega_k / 2) \\ T \omega_k \end{bmatrix} \quad (5.16)$$

ahol k a k -adik időpillanatot, míg T a mintavételi időt jelenti. Az $\mathbf{A}$ Jakobi mátrix minden egyes iterációban újra kiszámításra kerül az alábbiak szerint:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \mathbf{A}_{13} \\ 0 & 1 & \mathbf{A}_{23} \\ 0 & 0 & 1 \end{bmatrix}, \quad (5.17)$$

$$\begin{aligned}\mathbf{A}_{13} &= \frac{\partial f_1}{\partial \theta} = -\frac{1}{2}T(\gamma_L\omega_L + \gamma_R\omega_R)\sin\theta \\ \mathbf{A}_{23} &= \frac{\partial f_2}{\partial \theta} = \frac{1}{2}T(\gamma_L\omega_L + \gamma_R\omega_R)\cos\theta\end{aligned}\quad (5.18)$$

Tegyük fel, hogy a robot a saját helyzetét és irányát közvetlenül tudja mérni a következőképp:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (5.19)$$

ahol $\mathbf{v}_k = [v_x, v_y, v_\theta]^T$ reprezentálja a mérési zajt. Valójában az egyenlet a 5.6. egyenletnek felel meg, viszont a mi esetünkben ez lineáris. A $\mathbf{z}_k$ mérés tulajdonképpen, az x és y irányban mért távolságot, a giroszkóp által mért θ elfordulást tartalmazza, valamint mindhárom esetben egy véletlen zajt:

$$\mathbf{z}_k = \begin{bmatrix} x + v_x \\ y + v_y \\ \theta + v_\theta \end{bmatrix}, \quad (5.20)$$

A kimeneti $\mathbf{H}$ mátrix a következő alakú:

$$\mathbf{H} = [\mathbf{I}_3] \quad (5.21)$$

ahol $\mathbf{I}_3$ egy 3×3 -as egységmátrix.

5.2.4. Bővített kiterjesztett Kalman szűrő

Mivel a kerekek közötti távolságot és a kerekek sugarát nem ismerjük pontosan (a gumik keménységének változásából és a kerekek tengelyére nehezedő súlyból eredően), így ez szisztematikus hibákhoz és ez által a Kalman szűrő teljesítményének romlásához vezethet. Hogy orvosoljuk ezt a problémát, alkalmaznunk kell ugyanazt a bővített Kalman szűrőt, mint amit Martinelli [14] és Larsen [13] mutatott be. Ezt a 5.22. pontban láthatjuk:

$$\mathbf{x}_a = [x, y, \theta, \gamma_L, \gamma_R, d]^T \quad (5.22)$$

A kibővített $\mathbf{x}_a$ állapotvektorból kifolyólag a 5.16.-ben bemutatott $\mathbf{x}$ állapotvektor dinamikai egyenlete a következőképpen módosul:

$$\begin{bmatrix} x \\ y \\ \theta \\ \gamma_L \\ \gamma_R \\ d \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ \theta \\ \gamma_L \\ \gamma_R \\ d \end{bmatrix}_k + \begin{bmatrix} Tv_k \cos(\theta_k + T\omega_k/2) \\ Tv_k \cos(\theta_k + T\omega_k/2) \\ T\omega_k \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (5.23)$$

ahol k a k -adik időpillanatot, míg T a mintavételi időt jelenti. Az 5.17.-ben bemutatott $\mathbf{A}$ Jakobi mátrix a bővített állapotvektor hatására az alábbiak szerint módosul:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \mathbf{A}_{13} & \mathbf{A}_{14} & \mathbf{A}_{15} & 0 \\ 0 & 1 & \mathbf{A}_{23} & \mathbf{A}_{24} & \mathbf{A}_{25} & 0 \\ 0 & 0 & 1 & \mathbf{A}_{34} & \mathbf{A}_{35} & \mathbf{A}_{36} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5.24)$$

$$\begin{aligned} \mathbf{A}_{13} &= \frac{\partial f_1}{\partial \theta} = -\frac{1}{2}T(\gamma_L\omega_L + \gamma_R\omega_R)\sin\theta \\ \mathbf{A}_{14} &= \frac{\partial \gamma_L}{\partial f_1} = \frac{1}{2}T\omega_L\cos\theta \\ \mathbf{A}_{15} &= \frac{\partial \gamma_R}{\partial f_1} = \frac{1}{2}T\omega_R\cos\theta \\ \mathbf{A}_{23} &= \frac{\partial f_2}{\partial \theta} = \frac{1}{2}T(\gamma_L\omega_L + \gamma_R\omega_R)\cos\theta \\ \mathbf{A}_{24} &= \frac{\partial \gamma_L}{\partial f_2} = \frac{1}{2}T\omega_L\sin\theta \\ \mathbf{A}_{25} &= \frac{\partial \gamma_R}{\partial f_2} = \frac{1}{2}T\omega_R\sin\theta \\ \mathbf{A}_{34} &= \frac{\partial \gamma_L}{\partial f_3} = -T\frac{\omega_L}{d} \\ \mathbf{A}_{35} &= \frac{\partial \gamma_R}{\partial f_3} = T\frac{\omega_R}{d} \\ \mathbf{A}_{36} &= \frac{\partial f_3}{\partial d} = T\frac{\gamma_L\omega_L - \gamma_R\omega_R}{d^2} \end{aligned} \quad (5.25)$$

A kimeneti $\mathbf{H}$ mátrix a következő alakúra módosul:

$$\mathbf{H} = [\mathbf{I}_3 \quad \mathbf{O}_3] \quad (5.26)$$

ahol $\mathbf{I}_3$ egy 3×3 -as egységmátrix, míg $\mathbf{O}_3$ egy 3×3 -as null mátrix, ami azt jelenti, hogy ezek együttesen egy 3×6 -os mátrixot adnak.

A kiterjesztett Kalman szűrő rekurzív természetéből adódóan az állapot vektort inicializálnunk kell az algoritmus kezdetekor, amit az első mért értéként kezelünk. Emellett d , γ_L és γ_R névleges értékét egy kezdeti becslésként választjuk meg. Valamint adjuk meg a kezdeti bizonytalanságot mindezek mellett:

$$\mathbf{P}_0 = \mathbf{I}_6 \quad (5.27)$$

Ezen kívül definiálnunk kell a $\mathbf{Q}$ és $\mathbf{R}$ mátrixokat is:

$$\mathbf{Q} = 0.5 \cdot \text{diag} [0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1] \quad (5.28)$$

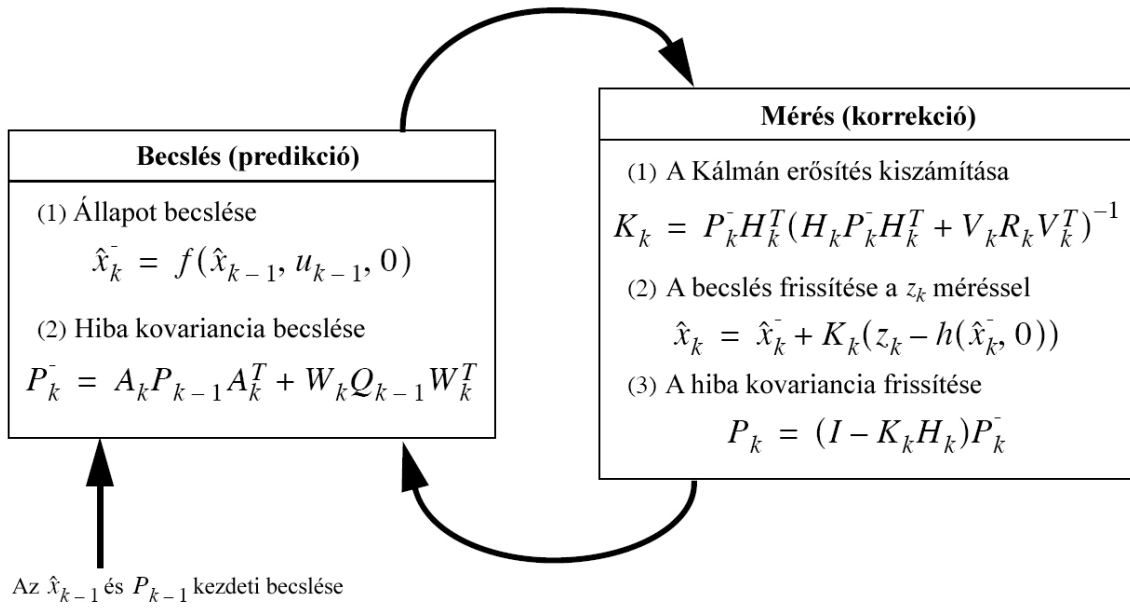
$$\mathbf{P} = 300\mathbf{I}_3 \quad (5.29)$$

Végezetül pedig meg kell adnunk $\mathbf{W}$ és $\mathbf{V}$ mátrixokat is:

$$\mathbf{W} = \begin{bmatrix} \frac{\gamma_R}{2} \cos(\theta_k) & \frac{\gamma_L}{2} \cos(\theta_k) \\ \frac{\gamma_R}{2} \sin(\theta_k) & \frac{\gamma_L}{2} \sin(\theta_k) \\ \frac{\gamma_R}{d} & -\frac{\gamma_L}{d} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (5.30)$$

$$\mathbf{V} = \mathbf{I}_3 \quad (5.31)$$

Az 5.2. ábra mutatja az algoritmus iteratív működését.



5.2. ábra. A bővített kiterjesztett Kalman szűrő működése.

Ezzel minden szükséges elemet megkaptunk egy kétkerekű differenciál-hajtású mobil robot bővített kiterjesztett Kalman szűrőjének implementálásához.

6. fejezet

Esettanulmány

6.1. Grafikus felhasználói felület

A grafikus felhasználói felületet (GUI) a Swing Java osztállyal készítettem el, melyet a [A.1.](#) ábrán láthatunk. A robothoz való kapcsolódás előtt szükséges adatok:

- Manuális vagy automata térképezési mód,
- A feltérképezendő terület szélessége és hosszúsága,
- A cellák mérete, mely megadja az elkészült térkép felbontását

A robothoz való kapcsolódás alatt egy állapotjelző ad tájékoztatást a folyamat állásáról, amely így felhasználóbarát környezetet teremt a program használoinak. Kapcsolódás után természetesen bármikor lehetőség van a robotról való lekapcsolódásra.

A *Start* gomb egyszeri¹, illetve többszöri² lenyomásával indul el a térképezés, mely során minden lépésben a program tájékoztatást ad a robot mindenkorai pozíciójáról, irányultságáról, valamint az adott lépés számáról.

A térképezés befejeztéről a program tájékoztatást ad a felhasználónak, ezután pedig lehetősége nyílik rá, hogy az egyes lépéseken végghaladva megtekintse a folyamatosan felépülő, formálódó térképet 3 különböző nézetben:

- Valószínűségi térkép nézet: a valószínűség értékeket tartalmazó térkép
- Költségmátrix nézet: a költségmátrix által felépített környezet
- Megtalált út nézet: az A* keresés által megtalált legrövidebb út alakja

Minden nézetnek megvan a rá jellemző színösszeállítása, melyek segítségével információkat közlünk a felhasználóval. Ezen színek jelentései:

- Valószínűségi térkép nézet:
 - Piros: a robot jelenlegi pozíciója
 - Szürke: az ismeretlen, még fel nem térképezetlen területet jelöli
 - Fekete: a valószínűleg foglalt cellákat jelenti
 - Fehér: a valószínűleg üres cellák tartoznak ide
- Költségmátrix nézet:

¹Automata térképezési módban

²Manuális térképezési módban

- Piros: a már meglátogatott cellák
- Szürke: az ismeretlen, még fel nem térképezetlen területet jelöli
- Fekete: a tiltott terület
- Fehér: a szabad, még fel nem térképezett terület
- Megtalált út nézet:
 - Piros: a kiindulási cella
 - Zöld: a cél cella
 - Sárga: a legrövidebb köztes út

Az egyes nézetekben a felhasználó kényelmesen tud navigálni az *Első*, *Előző*, *Következő*, *Utolsó* és *Ugrás* gombok segítségével, valamint az ezek felett elhelyezett csúszka használatával.

6.2. A térkép rajzolása

A program a térkép kirajzolásához egy általam implementált rajzoló algoritmust alkalmaz, mely nem tárolja el minden lépésben a valószínűségi térképet és a költségmátrixot (csökkentve ezzel a felhasznált erőforrásokat), hanem az egyes lépésekben megvizsgálja sorban az összes cellát, ha az adott cella valószínűségi vagy költség értéke változott az azt megelőző lépéshez képest, akkor ez a változás kerül eltárolásra. A térképek felépítésénél pedig az adott lépésig összegzi minden cellába ezeket az eltéréseket, így visszkapjuk az adott lépésbeli térképeket.

Következő lépésben a valószínűségi térkép vagy költségmátrix celláit felfogjuk egy kép pixeljeiként és az alábbi egyenlet segítségével kiszámoljuk minden cella/pixel RGB³ értékeit:

$$R = G = B = (1 - Pr_{x,y})255, \quad (6.1)$$

ahol $Pr_{x,y}$ az adott cella valószínűsége.

6.3. Tesztek

A legelső tesztek során karton dobozokat alkalmaztam a labirintus falainak felépítésére, ami hibás térképet eredményezett, az anyag jó hangelnyelő tulajdonságainak köszönhetően. A hibás visszaverődésekből adódóan nem voltak pontosak a megtalált falak vonalai a felépített térképen, valamint több alkalommal előfordult, hogy már egy megtalált fal szakaszt egy újabb mérés-sorozat után eltolva rajzolt be a térképbe a régi fal szakasz közvetlen közelébe. Ennek a problémának az orvoslására fa lapokat alkalmaztam az addigi karton lapok helyett, ami lényegesen pontosabb térképet eredményezett.

6.3.1. Egyszerű *L*-alakú labirintus

Az első feltérképezendő területet típus, amit a robot számára állítottam össze egy viszonylag egyszerűbb *L*-alakú labirintus egy kikerülendő akadállyal a közepén, mely

³Az angol vörös, zöld, kék szavak rövidítése.

a [A.3.](#) képen látható. A kiindulási térképen az egész terület szürke színű⁴, mivel a robot még nem végzett méréseket, viszont az idő előre haladtával egyre tisztább képet kapunk a területről. Az egyes lépések a következők:

- **1. lépés:** a [A.5.](#) képen látható az első 12×10 ultrahang mérés eredménye. Jól kivehető, hogy a robot még ebből a kevés mérésből is érzékelt 3 falat maga körül, valamint egy lehetséges akadálymentes utat az egyik irányba. Tisztán kivehető a két oldalsó irányban végzett ultrahang mérés, a szenzorkúp alakja, valamint az érzékelt akadály.
- **2. lépés:** a [A.8.](#) képen látható ezen lépés eredménye. A robot immár nagyobb biztonsággal észleli az előző lépésnél detektált falakat, valamint az *üres* terület is kezd letisztulni.
- **15. lépés:** ebben a lépésben a robot még csak a kiindulási pontja körüli cellákat térképezte fel, itt már közel 0 valószínűséggel rendelkeznek az *üres* és több, mint 0,9-es valószínűséggel rendelkeznek a *fal* cellák. Mindez a [A.6.](#) képen látható.
- **26. lépés:** a robot elindult a szabad területek irányába, mely a [A.9.](#) képen látható és sikeresen térképezve fel a két oldalt lévő falszakaszt.
- **28. lépés:** mely a [A.7.](#) képen látható. Ebben a lépésben a robot sikeresen megtalálta a szabad terület elkanyarodását, valamint már méréseket végzett az akadályról is.
- **40. lépés:** ebben a lépésben már sikeresen megtalálta a felállított akadályt a labirintus sarkában, valamint elkezdte feltérképezni a megmaradt labirintus részt, mely a [A.10.](#) képen látható.
- **61. lépés:** megfigyelhető a [A.11.](#) képen, hogy egyes *fal* cellák, a fals mérésekből következőleg eltűntek a labirintus felső részének falából.
- **80. lépés:** az előző lépésben taglalt cellákat a [A.14.](#) képen viszont sikeresen megtalálta a robot.

Ezzel előállt a felépített egyszerű *L-alakú* labirintus valószínűségi térképe. Tekintsük meg ennek a labirintusnak a költségmátrixát az egyes lépésekben, valamint egy-két, térképezés során megtalált utat.

- **1. lépés:** a [A.12.](#)-en látható az első szenzoradatokból felépített valószínűségi térképből előállított költségmátrix. Megjelenik a 3 oldalon sikeresen megtalált fal körül vont tiltott zóna.
- **11. lépés:** a robot a kiindulási pont körüli terület feltérképezése után elindul a szabad, azaz fehérrel jelzett területek felé. Ezt a [A.15.](#) képen láthatjuk.
- **30. lépés:** a [A.13.](#) képen is láthatjuk, hogy a robot mindig a hozzá legközelebb levő üres cella fele lépked, mindig az orientációjának megfelelő irányban, így kapjuk a piros színnel jelölt egyenes utat.
- **35. lépés:** ebben a lépésben láthatjuk a [A.16.](#) képen, hogy a lépésenként alakuló valószínűségi térképből felépített tiltott terület terjeszkedésével olyan terület is tiltottá válhat, amin előtte a robot haladhatott.
- **81. lépés:** a tiltott terület itt már teljesen körbevette a robotot, amit a [A.17.](#) képen láthatunk.

⁴Ez a szín jelöli a 0,6-os valószínűséget

- **95. lépés:** a [A.19.](#)-en pedig már az algoritmus vége látható, ahol nem maradt újabb feltérképezetlen terület.

A következő [A.18.](#) és [A.20.](#) képen pedig a **15.** és **83.** lépésben megtalált utat láthatunk.

6.3.2. Egyszerű négyzet-alakú labirintus

Az második feltérképezendő területet típus, amit a robot számára állítottam össze egy egyszerűbb négyzet-alakú labirintus hengeres oszloppal a közepén, mely a [A.4.](#) képen látható. A térképezés nehézségét és érdekességét az adja, hogy az ultrahang szenzor segítségével csak akkor kaphatunk érdemben is használható távolság adatokat, ha a szenzor merőlegesen áll az előtte húzódó akadállyal. Minél nagyobb a kettő közötti bezárt szög, annál rosszabb, fals eredményeket kaphatunk a nem megfelelő visszaverődésekből kifolyólag. Lássuk tehát, hogy az általam implementált algoritmus mennyire használható ilyen körülmények között. Az egyes lépések a következők:

- **1. lépés:** a mellékletben megtalálható [A.21.](#) képen láthatjuk, hogy a robot már az első ultrahangos mérésekkel megtalálta a két oldalsó falat, valamint észrevette a labirintus közepén fekvő oszlopot. Itt is jól kivehető az ultrahang szenzor alakja, ahogyan a kapott távolsági adatokkal megegyező távolságban lévő cellák valószínűségi értékét megnövelte, míg az ezek előtt lévő celláknak lecsökkentette.
- **10. lépés:** a mellette található [A.24.](#) képen látható, hogy a robot szépen megtalálta a kiindulási pontban lévő sarkokat, valamint újabb méréseket végzett az egyre formálódó oszlopról. Itt szeretném megjegyezni, hogy az egyes képeken látható, a robot mögött kialakuló megnövekedett valószínűségekből álló *csík*, a robot mögött található irdatlan mennyiségű kábel elhelyezkedéséből adódik, melyeket a robot néhány maga mögé tett ultrahang mérés alatt érzékel. Ebből adódik a csík egészen megfigyelhető ugyanakkora valószínűség. Ezen teszt során mutatom be a térképészeti algoritmusom igazi erősségét, miszerint ezeket a véletlen távolságokat játszi könnyedséggel tünteti el egy másik mérés segítségével.
- **30. lépés:** a [A.22.](#) képen láthatjuk a kialakult 2. sarkot, a formálódó oszlopot, valamint az előző saroknál feltűnt *csík* eltűnését. Látható ezen kívül a labirintuson kívül kialakuló falszakasz, mely az ultrahang szenzor fals méréseiből adódik. Ez a későbbiek során semmilyen gondot nem fog okozni.
- **70. lépés:** a [A.25.](#) képen láthatjuk a 70. lépésben kirajzolódott térképet láthatjuk, melynél a két megtalált falszakasz már közel egyenesnek tekinthető. Észrevehető valamint, hogy a kiindulási pontban lecsökkentette a sarok valószínűségét, mely az előző lépésben említett labirintuson kívülre kerülő mérésekből adódik.
- **103. lépés:** a robot a [A.23.](#) képen már sikeresen megtalált 3 falat, valamint kivehető a közepén elhelyezkedő oszlop is. Látható ezen kívül egy újabb labirintuson kívüli falszakasz.
- **109. lépés:** a következő [A.26.](#) képen pedig újabb robot mögött feltűnő *csík* látható, melyet a robot idővel (a 119. lépésben is látható módon) szintén kijavít majd. Az oszlop alakja közel véglegesnek tekinthető, alakja már csak kis

mértékben fog változni a költségmátrix lépéseinek bemutatása alatt felderített ok miatt.

- **119. lépés:** az előző lépésben említett *csík* javítása megtörtént, valamint sajnálatos módon kialakult egy 4. labirintuson kívüli falszakasz is. Mindez a [A.27.](#) képen látható.
- **149. lépés:** ebben a lépésben pedig már a végleges valószínűségi térkép látható, ahogy azt a [A.30.](#) kép is mutatja.

Végül tekintsük meg ennek a labirintusnak a költségmátrixát az egyes lépésekben, valamint egy-két, térképezés során megtalált utat.

- **2. lépés:** a [A.30.](#) képen a kezdeti költségmátrix található. Látható, hogy a robot számára 2 potenciális útvonal is lehetséges, ezek közül viszont a jelenlegi irányultságához közelebb lévő utat választja.
- **64. lépés:** ugorva egy nagyot elértünk a [A.28.](#) képen látható lépésre, ahol a robot már feltérképezte a kiindulási és a jelenlegi pozíciója között fekvő területet, valamint látható, hogy az algoritmus a labirintuson kívül eső falszakaszt is belevette a tiltott területbe.
- **106. lépés:** az érdekesség ezzel a lépéssel kezdődik, miszerint a [A.31.](#) képen látható, hogy a robot a jelenlegi költségmátrix alapján tervezhetne egy utat a kiindulási pontja felé, valamint a labirintuson kívülre vezető utat is használhatná. Utóbbi eset nem történne meg, mivel mindig a hozzá legközelebb eső cellát részesíti előnyben, így mire eljutna a kivezető úthoz, az már lezáródna előtte.
- **128. lépés:** itt (a [A.29.](#) képen) már látható a labirintusból kivezető szabad út lezáródása, a robot így csupán egy irányban mehet tovább.
- **132. lépés:** tovább fokozódva a helyzet a [A.32.](#) képen látható, hogy az irány, amerről a robot elért ide a valószínűségi térkép változásaiból adódóan a jelenlegi lépésben lezárult. Ez egy későbbi lépés során újból megnyílhat, de lépünk inkább az utolsó lépésre.
- **141. lépés:** itt már tisztán látható, hogy a térképezési algoritmusunk pár lépés múlva leállhat, hacsak nem változnak a valószínűségek olyan irányban, hogy a robot számára újra megnyíljanak az utak. Ez sajnos nem következett be, így a 149. lépésben az algoritmus leállt, pedig még bőven akadna a labirintuson belül olyan be nem járt cella, ahonnan még új információkat szerezhetne a robot a környezetéről. Ez a viselkedés az algoritmusom tulajdonságából adódik, így semmilyen hátrányt nem jelent a térképezés folyamán. Mindez a [A.35.](#) képen látható.

Ez a viselkedés a robot és a rendelkezésemre álló terem területének méretéből adódik, miszerint a robot méreteihez képest sajnos csak egy éppen megfelelő méretű labirintust nyílt lehetőségem építeni. Az algoritmusban beállított tiltott zóna mérete korlátozza a robot mozgását (természetesen pontosan ez a dolga), így egy viszonylag kis méretű labirintusban ehhez hasonló útlezáródások figyelhetők meg. Ez elkerülhetetlen és egyben természetes viselkedés. A tiltott zóna méretének csökkentésével ezek kiküszöbölhetővé válnának, ami viszont újabb problémákat vetne fel, miszerint a robot nekimegy a labirintus falainak, esetleg valamilyen módon fennakad rajtuk.

A fennmaradó A.34. és A.36. képen pedig a 65. és 143. lépésben megtalált utat láthatunk.

7. fejezet

Összegzés

7.1. Konklúzió

Dolgozatomban *megvalósítottam a szimultán lokalizáció és térképészet problémáját* az egyes részproblémáival¹ egyetemben egy *saját építésű* LEGO Mindstorms NXT alapú, két hajtott kerékkel rendelkező háromkerekű, differenciál-hajtású mobil roboton, amelyik a környezetből származó információt egy ultrahang szenzor, egy giroszkóp és egy érintés érzékelő segítségével kapja, valamint melynek *programozása Java leJOS nyelven történt*.

Mind a térképkészítés, mind pedig a lokalizáció során *kétdimenziós esetet dolgoztam fel*. A megvalósított robot a térképet az úgynevezett *foglaltsági háló alapú térképezési eljárással építi fel*, mely során *a használt alapegyenletek hiányosságait és hibáit is korrigáltam, javítottam*.

A robot mozgása egy *általam módosított, jelen környezetre optimalizált költségmátrix alapján*, A* algoritmus segítségével történik, mely segítségével a robot *egyenletes, kicsapongások nélküli mozgása* biztosított.

A lokalizáció esetében pedig egy *bővített kiterjesztett Kalman szűrőn*² alapuló lokalizációs eljárás került implementálásra, melyet a robot környezetéhez és szenzoraihoz optimalizáltam.

Implementáltam a SLAM-et és ezzel együtt annak részproblémáit *megoldó szoftvert*, mely során a robot memóriakapacitási korlátai miatt egy *megfelelő szoftverarchitektúrát alakítottam ki*. Ez a roboton futó programból, mely a szenzorok és motorok kezelését látja el, és egy PC oldali főprogramból áll, mely a beérkező információkat dolgozza fel és alakítja a felhasználók számára befogadható formára.

Létrehoztam ezen kívül egy *saját, erőforrás kímélő térképrajzoló algoritmust*, valamint egy *felhasználóbarát grafikus felhasználói felületet*, mely minden lépésben prezentálja az addig elkészült térképet.

Megoldottam a szoftverarchitektúrából adódó *kommunikációs problémákat* Bluetooth kapcsolat segítségével. Az ebből adódó esetleges *hibák kiküszöbölése ugyancsak megtörtént*.

Megépítettem *két különböző felépítésű és tulajdonságú tesztpályát* a rendelkezésre álló anyagok felhasználásával, melyekben az elkészült robot és program *tesztelése, illetve erősségeinek felderítése megtörtént*.

Eredményeimet *dokumentáltam*, kitűzött céljaimat *elértem*.

¹Térképezés, navigáció és lokalizáció.

²Augmented Extended Kalman Filter

7.2. Lehetséges további munka

Ez a probléma a robotikán belül egy igen érdekes, sokoldalú és nehezen megoldható probléma, mely még rengeteg kihívást, leküzdendő akadályt és izgalmas újdonságokat jelent az ezzel foglalkozó tudósoknak. Az általam megoldott és implementált probléma is rengeteg lehetséges további munkát tartalmaz, többek között:

- A navigációs eljárás javítása egy lézer, valamint egy kamera segítségével, ahol az előre mutató lézert a roboton elhelyezett kamera detektálná, azután pedig a feladat lenne egy képfeldolgozási algoritmus segítségével meghatározni a robot mindenkor távolságát egy adott akadálytól a lézer és az érzékelt kép szélei között lévő távolság alapján.
- Az előző pontban említett eljárásnál előforduló adatforgalom számára megfelelő sávszélességet biztosítani.
- Egyéb lézeres eljárások segítségével pontosabban meghatározni a robot mindenkor helyzetét.
- A jelenlegi 2 dimenziós térképet 3 dimenziósra bővíteni, mely során a jelenlegi foglaltsági háló alapú eljárást bővíteni ki a szükséges dimenziókra.
- A fentebb említett 3 dimenziós térkép rajzolása egyéb térképezési algoritmusok segítségével, mint például topológiai térképezési eljárás segítségével.
- A jelenleg meglévő A* algoritmus mellé más útvonalkereső algoritmusok implementálása, összehasonlítása.
- A térképezés során felmerülő apróbb finomítások elvégzése.
- A robot felépítésének javítása, újratervezése és megépítése nem előre legyártott építő elemek segítségével.
- A robot méretének csökkentése, ezzel egyidőben számának növelése.
- Az előző pontban említett létszám növelése következtében a robotok közötti szinkron, csoportos munka kiépítése.
- Egy statikus környezet feltérképezése helyett akár egy folyamatosan változó környezetben is tökéletesen működő algoritmus létrehozása.

Egy tökéletesen megvalósított SLAM problémát alkalmazó mobil robot segítségével nem csak labirintusok térképezhetőek fel, hanem akár egy katasztrófa súlytotta terület átvizsgálására is lehetőségünk nyílik esetleges túlélők után kutatva, lecsökkentve ezzel a megtalálásukhoz szükséges időt, ami nagy százalékban megnövelné a túlélési esélyeiket, de emberi életek kockáztatása nélkül lehetne mindezt véghezvinni.

Irodalomjegyzék

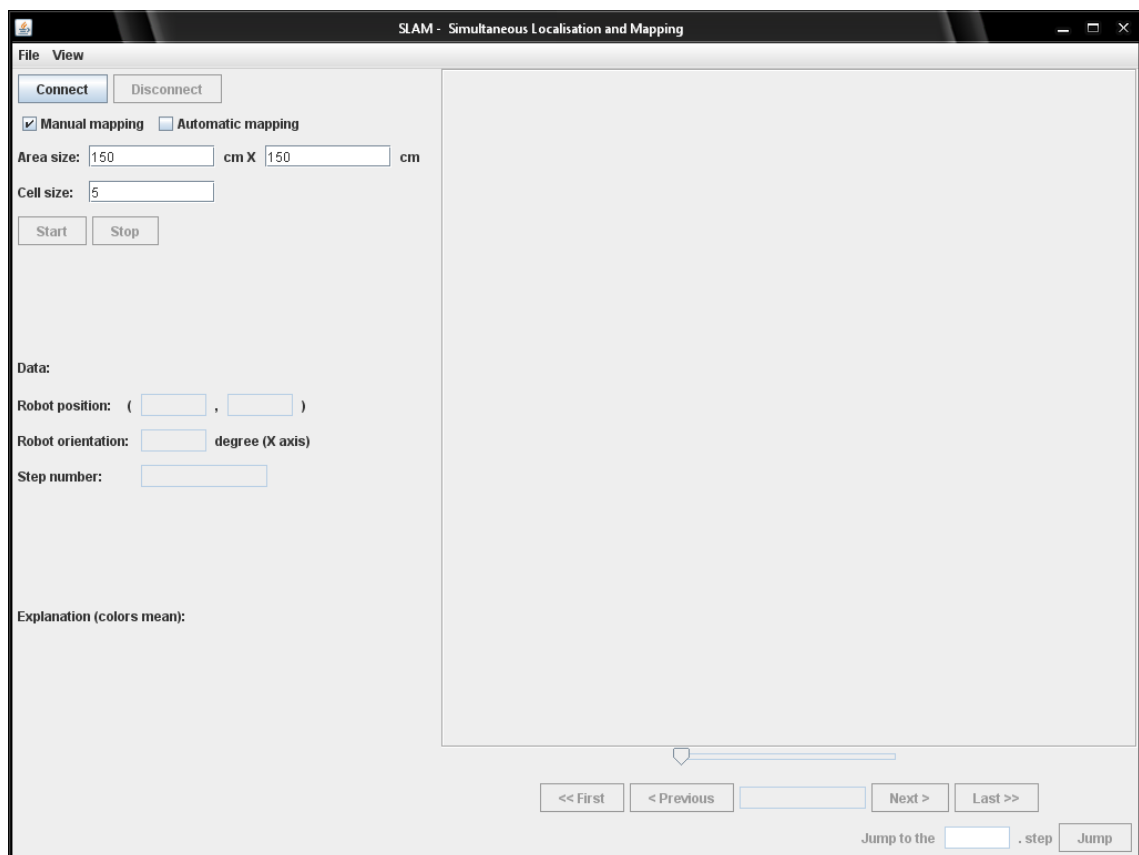
- [1] <http://krono.inaplo.hu/index.php/inter/8-networkstudies/289-mesterseges-intelligencia-es-robotika> SZTANKAY ZSOLT (2011. július 11.) Mesterséges intelligencia és robotika.
- [2] <http://www.nhit-it3.hu/images/tagandpublish/Files/it3-2-2-4-u.pdf> KÖM-LŐDI FERENC (2011. július 11.) Autonóm mobil robotok.
- [3] <http://mindstorms.lego.com/en-us/default.aspx> LEGO Mindstorms (2011. augusztus 20.) LEGO Mindstorms Homepage.
- [4] Durrant-Whyte, H., Rye, D., Nebot, E. (1996) Localisation of automatic guided vehicles. In: Robotics Research: The 7th International Symposium (ISRR'95), Giralt, G., Hirzinger, G. (ed.): New York: *Springer Verlag*, 613-625.
- [5] Thrun, S., Fox, D., Burgard, W. (1998) A probabilistic approach to concurrent mapping and localization for mobile robots. *Mach. Learning*. 31(1): 29-53.
- [6] Thrun, S. (1998) Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*. 99(1): 21-71.
- [7] Borenstein, J. (1995) Internal Correction of Dead-reckoning Errors With the Compliant Linkage Vehicle. *Journal of Robotic Systems*. 12(4): 257-273.
- [8] Sutton, R., Barto, A. (1998) *Reinforcement Learning: An Introduction*. Bradford Book, MIT Press.
- [9] Thrun, S. (1998) Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*. 99(1):21-71.
- [10] Russell, S. J., Norvig, P. (1995) *Mesterséges intelligencia modern megközelítésben* Panem Kiadó.
- [11] Smith, R., Cheesman, P. (1987) On the representation of spatial uncertainty. *International Journal of Robotics Research*. 5(4): 56-68.
- [12] Durrant-Whyte, H.F. (1988) Uncertain geometry in robotics. *IEEE Transactions Robotics and Automation*. 4(1): 23-31.
- [13] Larsen, T.D., Hansen, K.L., Andersen, N.A., Ravn, O. (1999) Design of Kalman filters for mobil robots; evaluation of the kinematic and odometric approach. IN: IEEE. International Conference on Control Applications. 1021-1026.
- [14] Martinelli, A., Tomatis, N., Siegwart, R. (2007) Simultaneous localization and odometry self calibration for mobil robot. *Autonomous Robots*. 22(1): 75-85.

- [15] <http://www.ifr.org/home/> IFR - International Federation of Robotics. (2011. szeptember 07.) IFR Homepage.
- [16] LEGO Mindstorms NXT User Guide. (2011. július 20.) The LEGO Group.
- [17] <http://lejos.sourceforge.net/nxj.php> leJOS. (2011. július 11.) leJOS, Java for LEGO Mindstorms Homepage.
- [18] Csorba, M. (1997) "Simultaneous Localisation and Map Building", Ph.D. dissertation, Univ. Oxford, 1997.
- [19] Csorba, M. Durrant-Whyte, H. (1996) A new approach to simultaneous localisation and map building. In: Proceedings of SPIE Aerosense. 115-125.
- [20] Leonard, J.J., Jacob, H., Feder, S. (1999) A computational efficient method for large-scale concurrent mapping and localisation. In: Proceedings of the Ninth International Symposium on Robotics Research. 169-176.
- [21] Castellanos, J.A., Martnez, J.M., Neira, J., Tardós, J.D. (1998) Experiments in multisensor mobile robot localization and map building. In: In 3rd IFAC Symposium on Intelligent Autonomous Vehicles. 173-178.
- [22] Castellanos, J.A., Tardós, J.D., Schmidt, G. (1997) Building a global map of the environment of a mobile robot: The importance of correlations. In: Proceedings IEEE International Conference Robotics and Automation. 1053-1059.
- [23] Guivant, J., Nebot, E.M., Baiker, S. (2000) Localization and map building using laser range sensors in outdoor applications. *Journal of Robotic Systems*. 17(10): 565-583.
- [24] Williams, S.B., Newman, P., Dissanayake, G., Durrant-Whyte, H. (2000) Autonomous underwater simultaneous localisation and map building. In: Proceedings IEEE International Conference Robotics Automation (ICRA). 1793-1798.
- [25] Chong, K.S. Kleeman, L. (1999) Feature-based mapping in real, large scale environments using an ultrasonic array. *The International Journal of Robotics Research*. 18(1):3-19.
- [26] Deans, M., Hebert, M. (2000) Experimental comparison of techniques for localization and mapping using a bearing-only sensor. In: Proc. of the ISER '00 Seventh International Symposium on Experimental Robotics. 395-404.
- [27] Durrant-Whyte, H., Bailey, (2006) T. Simultaneous Localization and Mapping: Part I. *IEEE Robotics and Automation* 13(2): 99-108.
- [28] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002) Fastslam: A factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence. AAAI. 593-598.
- [29] Antonelli, G., Chiaverini, S. (2007) Linear estimation of the physical odometric parameters for differential-drive mobile robots. *Autonomous Robots*. 23(1):59-68.

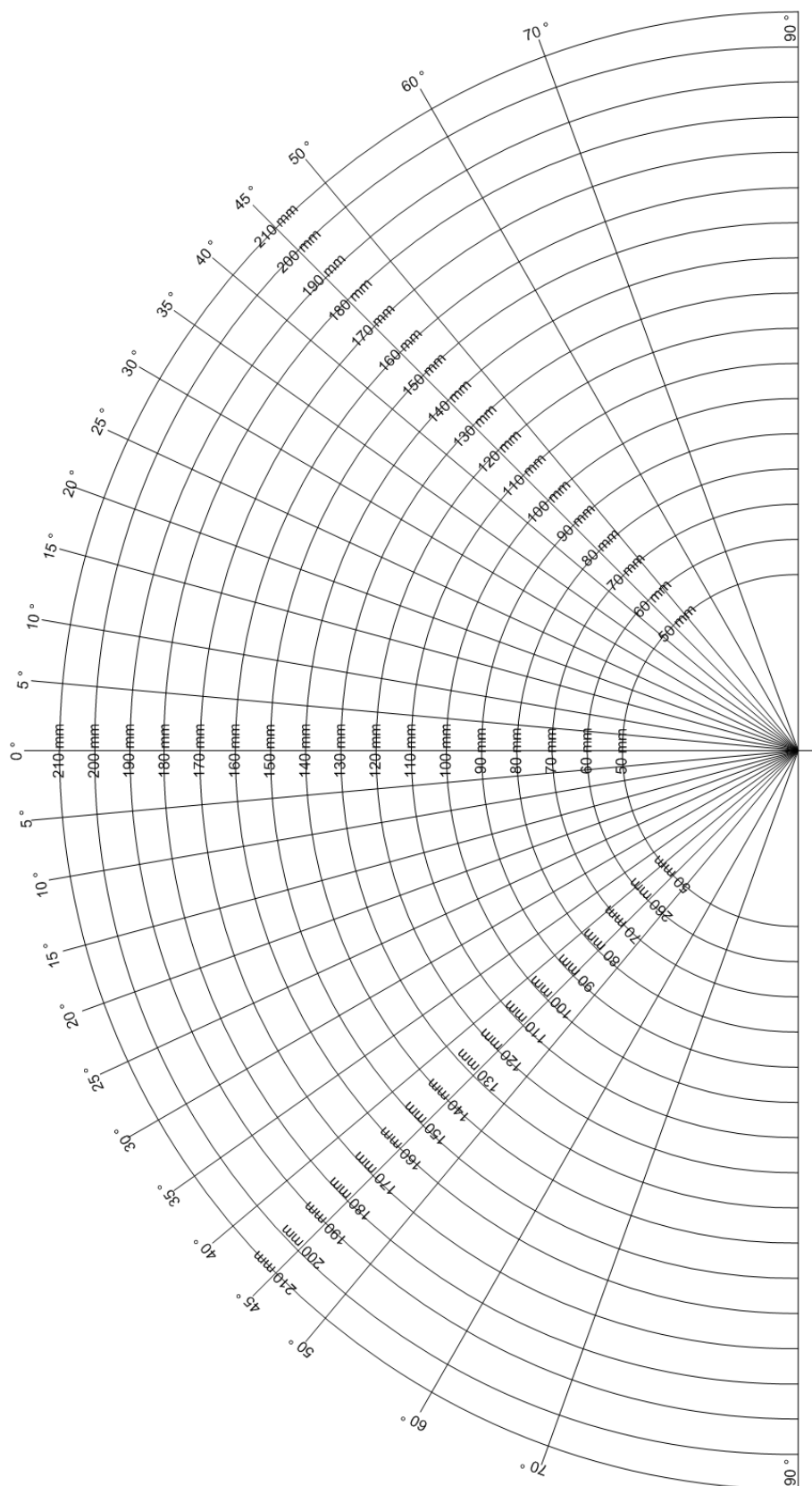
- [30] Kalman, R. E. (1960) A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*. 82(Series D): 35-45.
- [31] Lauritzen., S. L. (1981) Time series analysis in 1880. A discussion of contributions made by T.N. Thiele. *International Statistical Review*. 49(3): 319-333.
- [32] Steffen L. L., Thiele (2002) *Pioneer in Statistics*. Oxford University Press.
- [33] Martinelli, A., Nguyen, V., Tomatis, N., Siegwart, R. (2007). A relative map approach to SLAM based on shift and rotation invariants. *Robotics and Autonomous Systems*. 55(1): 50-61.
- [34] Wurm, K. M., Stachniss, C., Grisetti, G. (2010). Bridging the gap between feature- and grid-based SLAM. *Robotics and Autonomous Systems*. 58(2): 140-148.
- [35] Chatterjee, A., Ray, O., Chatterjee, A., Rakshit, A. (2011) Development of a real-life EKF based SLAM system for mobile robots employing vision sensing. *Expert Systems with Applications*. 38(7): 8266-8274.
- [36] Morena, L., Garrido, S., Blanco, D., Muñoz, M. L. (2009) Differential evolution solution to the SLAM problem. *Robotics and Autonomous Systems*. 57(4): 441-450.
- [37] Haoyao, C., Dong, S., Jie, Y. (2009) Global localization of multirobot formations using ceiling vision SLAM strategy. *Mechatronics*. 19(5): 617-628.
- [38] Begum, M., Mann, G. K. I., Gosine, R. G. (2008) Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots. *Applied Soft Computing*. 8(1): 150-165.
- [39] Cadena, C., Neira, J. (2010) SLAM in $O(\log n)$ with the Combined Kalman-Information Filter. *Robotics and Autonomous Systems*. 58(11): 1207-1219.
- [40] <http://www.cs.utexas.edu/~kuipers/slides/L13-occupancy-grids.pdf> Benjamin Kuipers Occupancy Grids
- [41] <http://www.cs.brown.edu/~tld/courses/cs148/02/uncertainty.html> Reasoning Under Uncertainty

A. Függelék

Egyebek



A.1. ábra. A grafikus felhasználói felület.



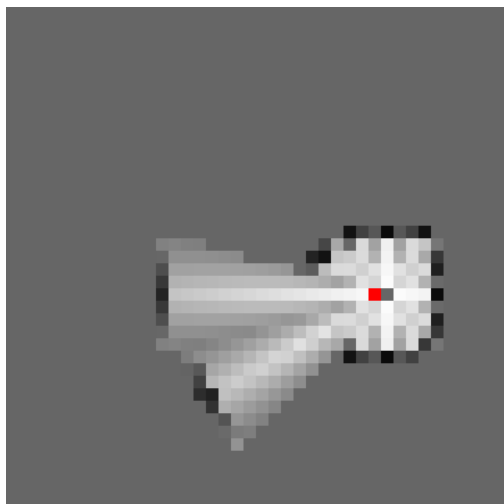
A.2. ábra. Az ultrahang szenzor teszt pályája.



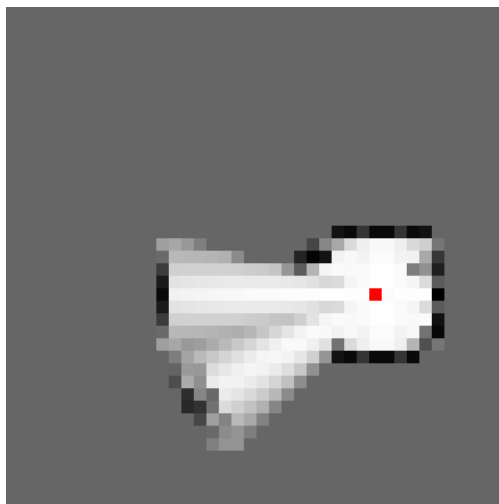
A.3. ábra. Az L-alakú labirintus a kikerülendő akadállyal.



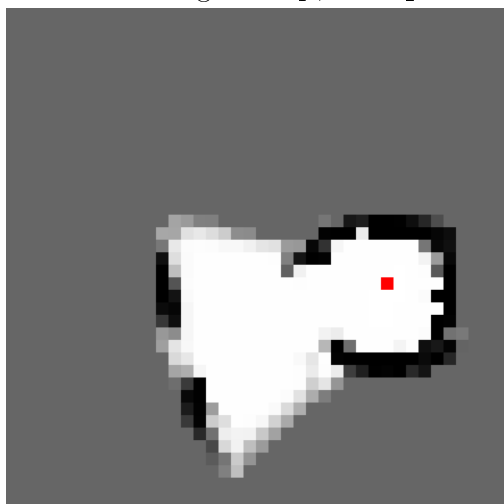
A.4. ábra. A négyzet-alakú labirintus, közepén egy oszloppal.



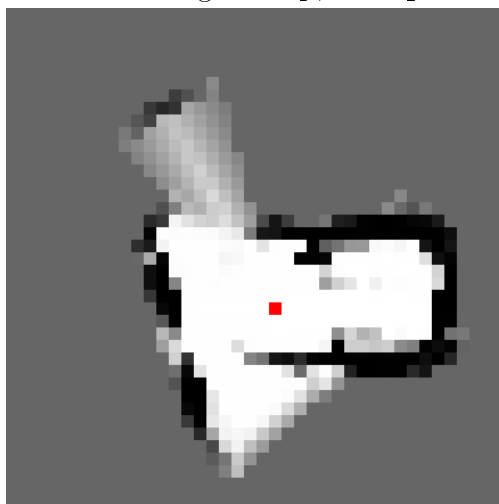
A.5. ábra. *L* labirintus,
valószínűségi térkép, **1. lépés.**



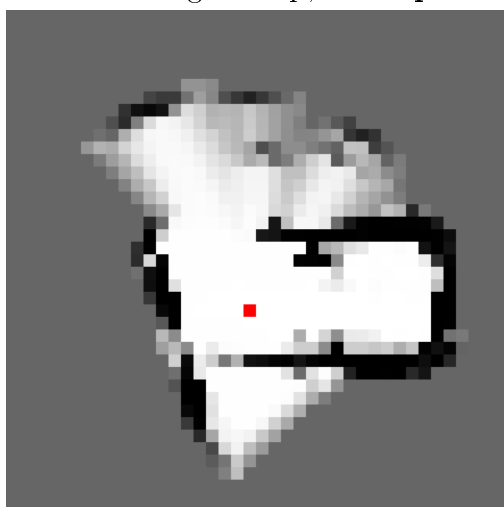
A.8. ábra. *L* labirintus,
valószínűségi térkép, **2. lépés.**



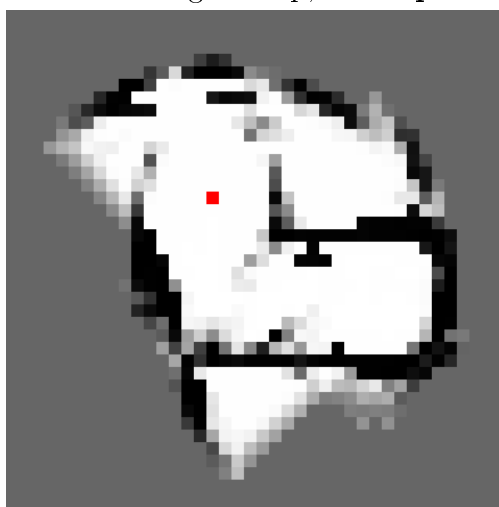
A.6. ábra. *L* labirintus,
valószínűségi térkép, **15. lépés.**



A.9. ábra. *L* labirintus,
valószínűségi térkép, **26. lépés.**



A.7. ábra. *L* labirintus,
valószínűségi térkép, **28. lépés.**



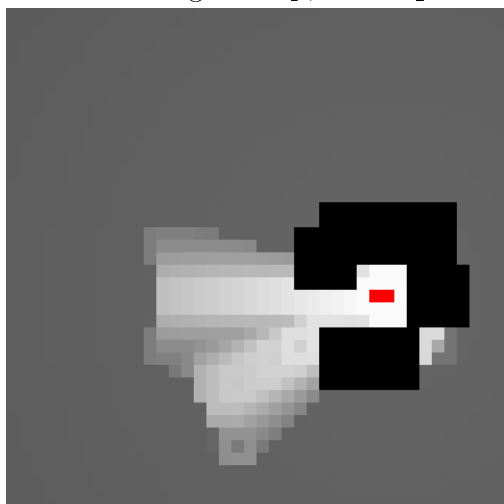
A.10. ábra. *L* labirintus,
valószínűségi térkép, **40. lépés.**



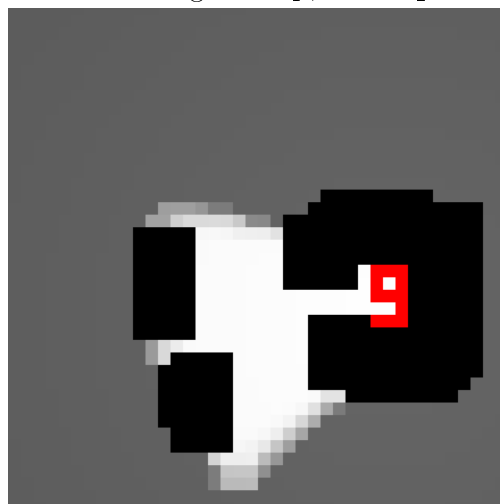
A.11. ábra. *L* labirintus,
valószínűségi térkép, **61. lépés.**



A.14. ábra. *L* labirintus,
valószínűségi térkép, **80. lépés.**



A.12. ábra. *L* labirintus,
költségmátrix, **1. lépés.**



A.15. ábra. *L* labirintus,
költségmátrix, **11. lépés.**



A.13. ábra. *L* labirintus,
költségmátrix, **30. lépés.**



A.16. ábra. *L* labirintus,
költségmátrix, **35. lépés.**



A.17. ábra. L labirintus,
költségmátrix, **81. lépés.**



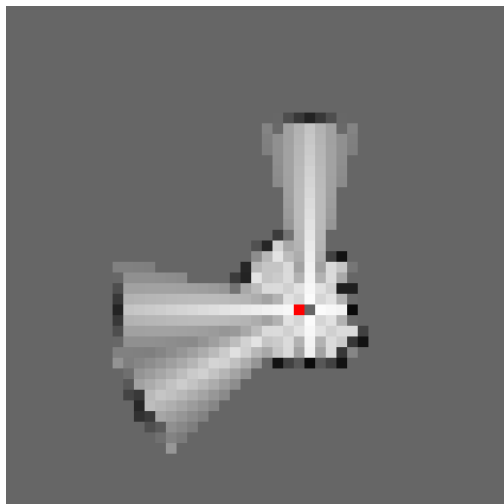
A.19. ábra. L labirintus,
költségmátrix, **95. lépés.**



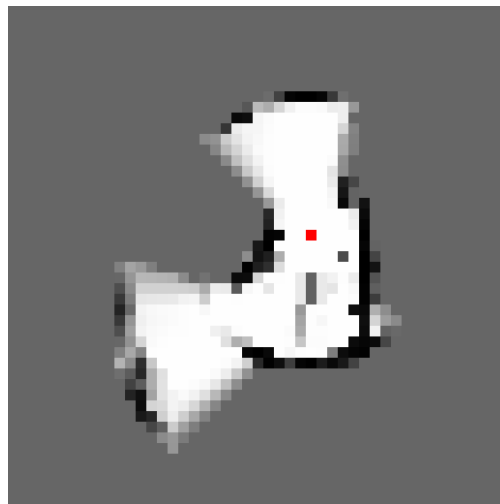
A.18. ábra. L labirintus, talált
legrövidebb út, **15. lépés.**



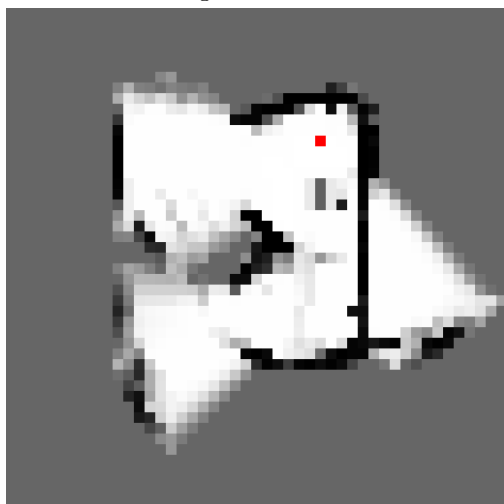
A.20. ábra. L labirintus, talált
legrövidebb út, **83. lépés.**



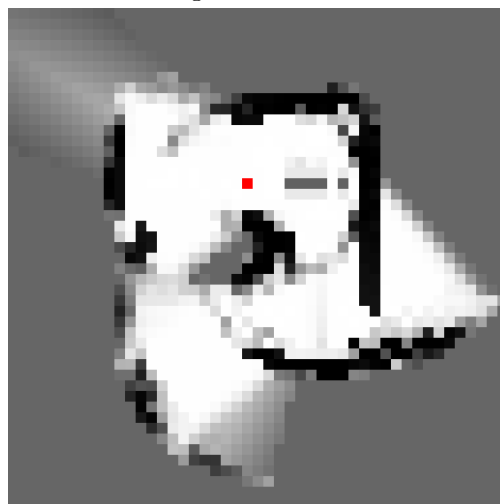
A.21. ábra. *Négyzet labirintus*, valószínűségi térkép, **1. lépés.**



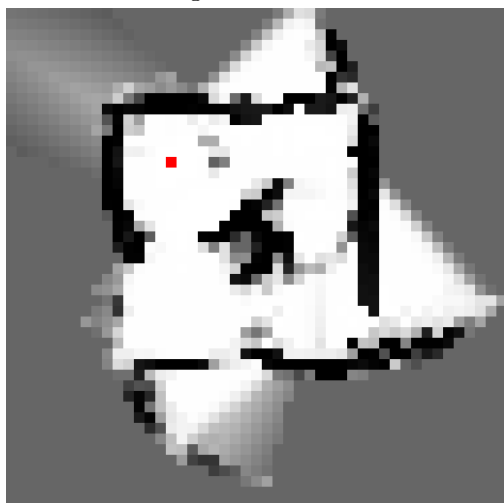
A.24. ábra. *Négyzet labirintus*, valószínűségi térkép, **10. lépés.**



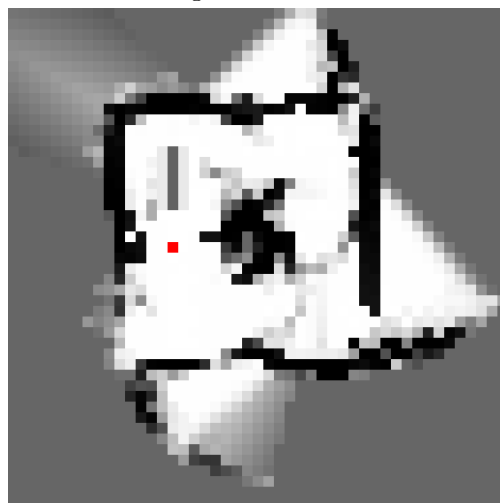
A.22. ábra. *Négyzet labirintus*, valószínűségi térkép, **30. lépés.**



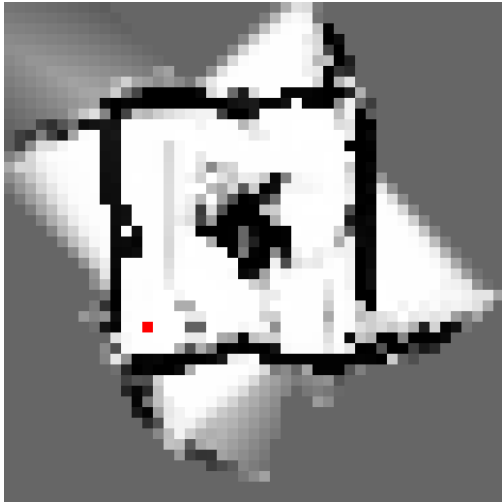
A.25. ábra. *Négyzet labirintus*, valószínűségi térkép, **70. lépés.**



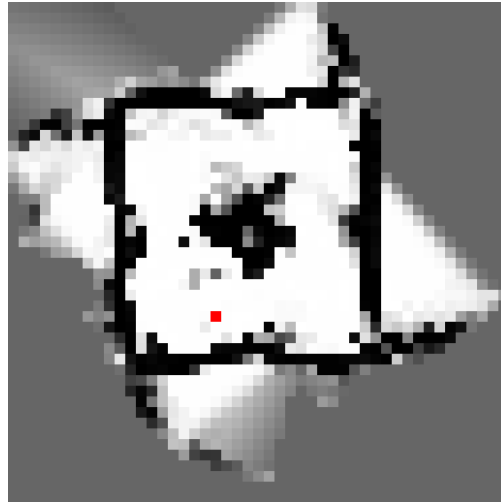
A.23. ábra. *Négyzet labirintus*, valószínűségi térkép, **103. lépés.**



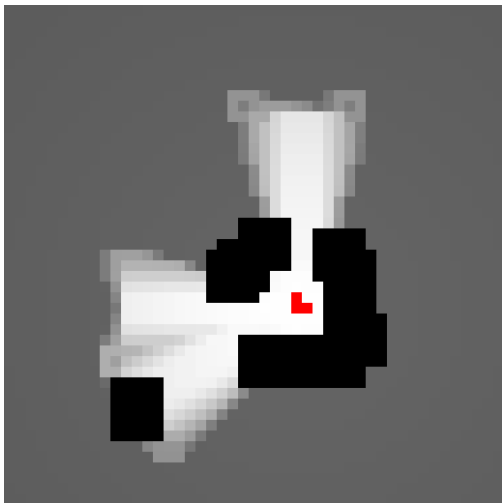
A.26. ábra. *Négyzet labirintus*, valószínűségi térkép, **109. lépés.**



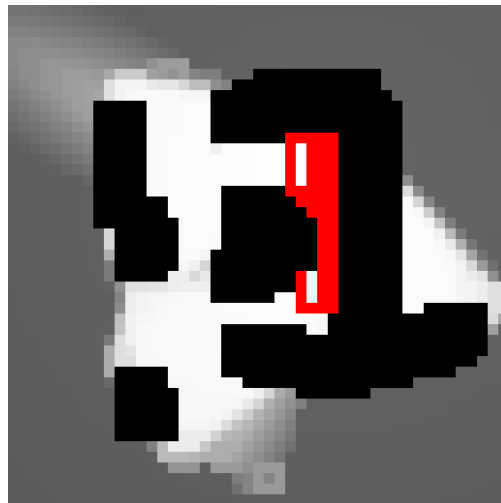
A.27. ábra. *Négyzet* labirintus, valószínűségi térkép, **119. lépés.**



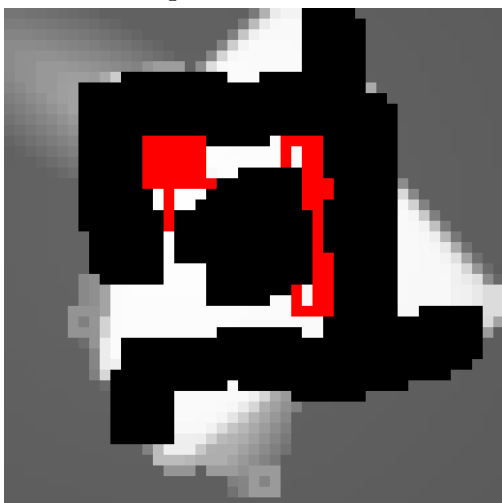
A.30. ábra. *Négyzet* labirintus, valószínűségi térkép, **149. lépés.**



A.28. ábra. *Négyzet* labirintus, költségmátrix, **2. lépés.**



A.31. ábra. *Négyzet* labirintus, költségmátrix, **64. lépés.**



A.29. ábra. *Négyzet* labirintus, költségmátrix, **106. lépés.**



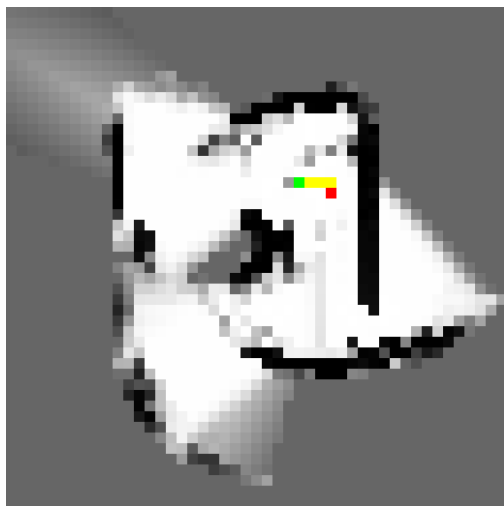
A.32. ábra. *Négyzet* labirintus, költségmátrix, **128. lépés.**



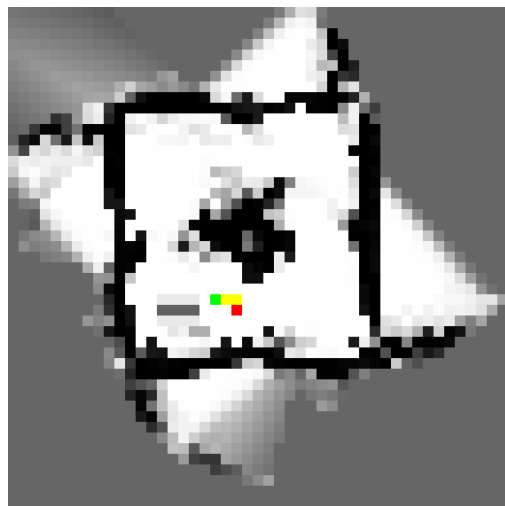
A.33. ábra. *Négyzet* labirintus,
költségmátrix, **132. lépés.**



A.35. ábra. *Négyzet* labirintus,
költségmátrix, **141. lépés.**



A.34. ábra. *Négyzet* labirintus,
talált legrövidebb út, **65. lépés.**



A.36. ábra. *Négyzet* labirintus,
talált legrövidebb út, **143. lépés.**