



SZEGEDI TUDOMÁNYEGYETEM
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
MÉRNÖKI KAR

KÜLTÉRI LOKALIZÁCIÓ MEGVALÓSÍTÁSA A MAGNI MOBIL ROBOTTAL

TDK DOLGOZAT

Kovács Imre

mechatronikai mérnök MSc

hallgató

Témavezetők:

Dr. Odry Ákos

főiskolai docens

Dr. Sarcevic Péter

főiskolai docens

Pesti Richárd

egyetemi tanársegéd

SZEGED

2024.

TARTALOMJEGYZÉK

1.	BEVEZETÉS	3
2.	SZAKIRODALMI ÁTTEKINTÉS	4
2.1	Mobil robot lokalizációs technológiák	4
2.1.1	Inerciális mérőegység (IMU)	4
2.1.2	Enkóder.....	6
2.1.3	GPS modul.....	8
2.1.4	LiDAR	11
2.1.5	Kamera	12
2.2	Szenzorfüziós módszerek	14
2.2.1	Valószínűség alapú megközelítés.....	14
2.2.2	Kalman-szűrő	15
2.2.3	További módszerek	18
3.	MÉRŐRENDSZER ÉS ADATGYŰJTÉS.....	20
3.1	Általánosságban.....	20
3.2	Ubiquity Robotics – Magni mobil robot.....	21
3.3	U-BLOX C94-M8P-3 RTK GPS Modul	23
3.4	UBLOX NEO-6M GPS Modul	25
3.5	Inerciamérő egységek	26
3.5.1	XSENS MTI-670 IMU modul.....	26
3.5.2	ICM-20948 IMU tömb	28
3.5.3	3D nyomtatott tartó.....	29
3.6	A mérés menete	30
4.	MÉRÉSI ADATOK FELDOLGOZÁSA ÉS KIÉRTÉKELÉSE	32
4.1	A mérés során mentett fájlok kibontása és konvertálása	32
4.2	Referencia koordináták kiszámítása (Ground Truth)	33

4.3	Kiterjesztett Kalman-szűrő (EKF).....	34
4.3.1	Mozgásmodell	35
4.3.2	Megfigyelésmodell	37
4.3.3	Inicializáció, optimalizálás előtti eredmények	38
4.4	PSO-alapú optimalizált Kalman-szűrő beállítás.....	40
4.4.1	A PSO algoritmus általánosan.....	40
4.4.2	A futtatott PSO algoritmus	41
4.5	Talajtípus klasszifikáció megvalósítása neurális hálózattal	44
4.5.1	A klasszifikációs neurális hálózatról általánosan	45
4.5.2	Nyers IMU adatok alkalmazása.....	45
4.5.3	A legpontosabb megközelítés.....	46
4.6	Fuzzy rendszer alkalmazása a szűrő és a neurális hálózat összekötésére.....	49
4.6.1	A megvalósított Fuzzy következtetési rendszer	50
4.6.2	Eredmények.....	51
4.7	Adaptív Kalman-szűrő neurális hálózattal és Fuzzy rendszerrel.....	52
4.8	Az eredmények értékelése és összehasonlítása hasonló rendszerekkel.....	53
4.8.1	Lokalizáció	53
4.8.2	Klasszifikáció	53
5.	ÖSSZEFOGLALÁS.....	55
	IRODALOMJEGYZÉK.....	56

1. BEVEZETÉS

A dolgozatban alacsony költségű szenzorokkal, szenzorfüziós algoritmussal megvalósított kültéri lokalizáció kerül implementálásra. Ehhez kapcsolódóan először áttekintésre kerülnek a kapcsolódó témakörök szakirodalmi, mint például a lokalizációhoz kapcsolódó szenzortípusok vagy a szenzorfüziós algoritmusok. Ezt követően bemutatásra kerül a megvalósított mérőrendszer, melyet egy Ubiquity Robotics Magni mobil robotra szerelek fel. Felszerelésre kerül egy XSENS MTI-670 felsőkategóriás IMU modul, és egy UBLOX C94-M8P-3-as RTK-s GPS-modul, melyek referenciaként működnek, emellett illesztettem 9 darab ICM-20948 IMU tömböt, és egy UBLOX NEO-6M GPS modul, mely a rendszer alacsony költségű részeit foglalják magukba. Az IMU-k egy 3D nyomtatott tartóra kerülnek felszerelésre. Az adatgyűjtés a Magni roboton és egy vele szinkronizált Laptopon a Robot Operating System (ROS) segítségével valósul meg. A mérés dinamikus trajektóriákon, – azaz különböző sebességekkel lassabb és gyorsabb mozgással – és változatos talajtípusokon (például fű, aszfalt, homok) kerül megvalósításra.

Az adatfeldolgozási részben először kifejlesztésre kerül egy Kalman-szűrő, mely a vezérlősebességet (/cmd_vel topic) használja mozgásmodealként, a NEO-6M GPS modult megfigyelésmodealként. A valós mozgás modellezésére az RTK-s GPS modul kerül felhasználásra ground_truth adatsorként. A zajmátrixok először általánosan kerülnek beállításra, amelyet a következő részben optimalizálok részecske-raj algoritmus (PSO) segítségével. Ezt követően betanítok egy klasszifikációs neurális hálózatot, mely az IMU tömbből használ fel adatokat a talajtípus meghatározására jellemzők kinyerésével. Ezt követően a két rendszer összekötésre kerül Fuzzy következtetési rendszer segítségével.

Az önálló munka végén létrejön egy alacsony költségű Kalman-szűrő algoritmus, mely adaptívan képes változtatni zajmátrixait a talajtípus klasszifikáció eredményének megfelelően. Az adaptív rendszer, amely a korábban tárgyalt talajtípusokon átlagosan több mint 10 százalékos javulást mutat a lokalizációs teljesítményben. Ez a módszer egy széleskörű elemzés eredménye, amely 40 különböző trajektóriát és scenáriót vizsgált. Az adaptív megoldás előnye, hogy az aktuális környezeti viszonyokhoz igazodva képes optimalizálni a kovariancia mátrixokat, ezáltal kezelve a parazita gyorsulások, környezeti zavarások, kerekek csúszása, egyenetlen talajviszonyok és vibráció okozta hatásokat. Ennek köszönhetően talajtípustól és környezeti tényezőktől független működés valósítható meg ezzel a low-cost (körülbelül 10 ezer Forintos) megoldással.

2. SZAKIRODALMI ÁTTEKINTÉS

A mobil robotok esetében a pontos lokalizáció problémája az egyik, hanem a legfontosabb területet fedi le, hiszen lokalizáción a robot elhelyezkedését értjük a térben. Bármely mobil robottal megvalósított feladat alapja, hogy minden pillanatban tudjuk a feladatot kielégítő magabiztossággal a robot helyzetét. A pontos lokalizációhoz szükségünk van számos információra a robotról (pl. odometria) és annak környezetéről is, melyhez a különböző szenzoradatokkal juthatunk hozzá.

Ebben a fejezetben tehát a különböző érzékelők mobil robotokon való alkalmazásának technológiájával kapcsolatos témakörök kerülnek kifejtésre. Azon szenzorok, amelyek utána később a kutatási részben használatra is kerülnek, például az inerciális mérőegységek (inertial measurement unit, röviden IMU), mely a robot mozgástani jellemzőit érzékeli, az enkóder, mely a kerekek (vagy esetünkben a motorok) elfordulását érzékeli, a GPS modulok, melyek a földrajzi helyzetről szolgáltatnak információkat, a LiDAR szenzorok, melyek lézersugarak segítségével térképeznek, vagy a kamerák, melyek vizuális információkat szolgáltatnak.

Azonban egy szenzor önálló alkalmazása a legtöbb esetben nem elég a kellően pontos lokalizációhoz, mivel korlátozottak és bizonyos hibaszázalékkal is rendelkeznek. Például egy kerékenkóder esetében, megcsúszás esetén igen gyorsan elszállnak az mozgásértékek, holott a robot közel állóhelyzetben is tartózkodhat. Ilyenkor például az IMU adatok felhasználásával könnyen következtethetünk egy valóshoz közeli mozgásra. Az ilyen következtetési rendszereket, ahol több érzékelő adatait valamilyen módon ötvözve javítunk a lokalizáció pontosságán, szenzorfüziós algoritmusoknak nevezzük. Így lehetővé tesszük a különböző szenzorok előnyös tulajdonságainak kiemelését és hátrányaik háttérbe szorítását.

Vannak szenzorok, melyek csak kültéri, vagy csak beltéri környezetben használhatóak, illetve vannak, amelyek mindkét környezetben alkalmazhatóak. A szakirodalmi áttekintés a dolgozat témájához kapcsolódva jobban kiemeli a kültéri használatban is alkalmazható szenzorokat, valamint említést tesz a további technológiákról is. A Magni robottal megvalósított mérőrendszer és adatgyűjtés során IMU, enkóder és GPS modul került felhasználásra, így ezek a témakörök a hangsúlyosabbak.

2.1 Mobil robot lokalizációs technológiák

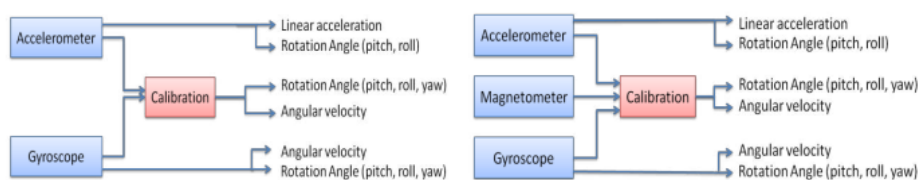
Ebben az alfejezetben kerülnek részletezésre a mobil robotok lokalizációjában alkalmazott technológiák, az IMU, az enkóder, a GPS, a LiDAR és a kamera.

2.1.1 Inerciális mérőegység (IMU)

Az inerciális mérőegységek, azaz IMU-kat először a múlt század első felének közepén kezdték használni, legelőször repülőgépek navigálására. Ekkor még drágák és a mai

méretükhöz képest sokkal nagyobbak voltak, ezért csak és kizárólag nagyméretű alkalmazásokra (haditechnika, azon belül is repülőgépek) volt lehetőség. A mikroelektronika későbbi térhódításával azonban már alkalmazhatóvá vált széles körben, hiszen kevesebb feldolgozási erőforrásigénnyel és kisebb mérettel rendelkeznek, valamint vannak olcsó változataik is, azonban jelenleg is folyik a verseny a legjobb IMU-k gyártásáért, például az X-Sens a Microstain, vagy a Honeywell között, amelyek még jelenleg is borsos árakkal rendelkeznek. (Ahmad, és mtsai. 2013.)

Az inerciális mérőegységek elsősorban sebességet, orientációt és gravitációs erőt mérnek, mégpedig úgy, hogy egy tokozásba integrálják a szenzorokat, melyek képesek a fent említett paramétereket mérni. A régebbi alkalmazásokban két szenzoros IMU-kat használtak (1. ábra bal oldal), melyek mindhárom térbeli tengely mentén egy-egy giroszkópot és gyorsulásmérőt használtak, míg később minden tengelyre felszerelésre került egy-egy magnetométer is, megalkotva a háromszenzoros IMU-t (1. ábra jobb oldal). A kétszenzoros előnye, hogy nem zavarja meg akármilyen, az útjába kerülő ferromágneses anyag a magnetométereket hamis adatot szolgáltatva, viszont a giroszkópnál driftelés is felléphet, amelyet pedig a háromszenzoros modell képes korrigálni. Tehát mindig az alkalmazástól függően érdemes választani a szenzort. (Ahmad, és mtsai. 2013.)



1. ábra A két- és háromszenzoros IMU összehasonlítása Forrás: Ahmad, és mtsai. (2013.)

Az utóbbi években elterjedtek a mikro-elektromechanikai rendszereken (MEMS) alapuló felhasználói IMU-k, amelyek akár már néhány ezer Forintos darabáron is kaphatóak, kevésbé pontosak, viszont egy tömbben alkalmazva őket, megfelelő kalibrációval akár labor körülmények között biztosítható egy csúcskategóriás IMU teljesítménye. Ehhez persze szükség van szoftveres oldalon egy sokkal komplexebb utómunkára, amely a zajok kiszűrésére irányul legfőképp. Segíthet a valószínűségeen alapuló megközelítés, az IMU-tömb méréseinek felhasználása mesterséges neurális hálózat tanítására, illetve komplex matematikai modellek használata is. (Harindranath és Arora 2024)

A kutatás során a mérőrendszerbe is beépítésre került egy csúcskategóriás IMU (X-Sens) és egy, kilenc olcsóbb IMU-t tartalmazó tömb is, mely majd a megfelelő fejezetben

kifejtésre is fog kerülni. A szakirodalmakban is található rengeteg féle alkalmazás IMU használatára, a következő bekezdésekben pár említésre méltó konstrukciót fogok bemutatni.

Kurpath és mtsai. (2024) az emberi kar kinematikájával foglalkoztak, modellezve azt csuklók és mozgástípusok szerint, és ebből próbáltak egy robotkart modellezni, mely ugyanolyan mozgásokra képes, mint egy emberi kar. Ebben a kutatásban három különböző pontra szerelt inerciális mérőegységet használtak a mozgási adatok érzékelésére.

Aslam és mtsai. (2020) egy SLAM (egyidejű lokalizáció és térképezés) feladatot oldottak meg, IMU szenzor adatokkal és különböző más szenzorok (LiDAR, GPS) fúziójával egy autonóm navigációval ellátott mobil roboton, mely felépítése hasonló a dolgozatban használt robothoz.

Li, Su és Yang (2019) egy alacsony költségű IMU-t egészített ki odometriával és LiDAR rendszerrel, egy mobil robot pozicionáló algoritmus fejlesztéséhez, és egyszerű kísérletekkel vizsgálták a nyers IMU adatok, valamint a odometria segítségével fuzionált, Kalman szűrőn alapuló algoritmus pontossága közötti különbséget. Hasonló területen dolgoztak Li és mtsai. (2022) is, ahol négykerekű robotot teszteltek különböző terepviszonyok között.

Guo és mtsai. (2024) kifejezetten az alacsony költségű IMU-t használtak egy AGV (Automatad Guided Vehicle) vezérléséhez, és modell alapú mélytanulással próbálták az mérési eredményekből kielégítő szinten kiszűrni a mérési zajokat. Olyan problémákkal szemben állva, mint az alacsony érzékenység, a nagy, véletlenszerű zajok mozgás közben, vagy a tengelyeltolódások.

Patel és Faruque (2021) pedig szintén az alacsony költségű IMU-kkal foglalkoztak, és azt vizsgálták, hogy több alacsony költségű IMU méréseinek fúziójával ki lehet-e váltani egy IMU és GPS modul fúzióját.

A fenti bekezdések tehát rávilágítanak arra, hogy az IMU-k számos kutatási területen használhatóak, és hogy van lehetőség egy megfizethetőbb, ugyanakkor komplexebb feldolgozási lépésekből álló rendszerek kiépítésére. Mindemellett az is belátható, hogy a legtöbb alkalmazásban nem célszerű csak ezt az egy szenzortípust használni, hanem érdemes több szenzor fúzióját alkalmazni a megfelelő mértékben pontos rendszer kialakításához.

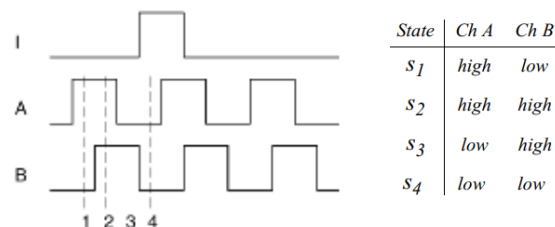
2.1.2 Enkóder

Inkrementális helyzetérzékelőnek, röviden enkódernek nevezzük azokat a szenzortípusokat, melyek kis szögelfordulások esetén egy-egy impulzust adnak. Az, hogy egy teljes kör, azaz 360° szögelfordulásához hány impulzus tartozik, az enkóder felbontásának nevezzük. Egy enkóderrel a pozíciót, azaz az impulzusszámot tudjuk csak megmondani, a

forgási irányt nem, viszont, ha a másikhoz képest eltolásban felszerelünk még egy enkódert, akkor már a forgási irányra is lehet következtetni, ezt nevezzük két csatornás enkódernek. Ebben az esetben az enkóder lehet abszolút, azaz a pozíciót előre és hátra is számolni képes. Egy csatornás esetben pedig mindig inkrementális enkóderről beszélhetünk, mely nulláról indul, és csak felfelé képes számolni. Pontosabb helymeghatározás érdekében kettőnél több csatornás enkódereket is használnak. Kivételük szerint léteznek optikai elven működő és mágneses elven működő enkóderek. (Bánlaci és Lovas 2012.)

Mobil robotok esetében az inkrementális adóknak hatalmas szerep jut, általában velük oldják meg a szögsebesség- és pozíció-mérést, amelyből következtetnek a mobil robot útvárá és helyzetére, azaz az odomertiára. Az enkóderek proprioceptív érzékelők, tehát a helyzetbecslésük a saját koordináta-rendszerükben a legbiztosabb, aminek következménye, hogy lokalizációs probléma alkalmazása esetén pontatlanok, korrekcióra szorulnak. (Siegwart és Nourbakhsh 2004.)

Leggyakrabban optikai elven működő enkódereket alkalmaznak mobilis robotok esetében. Működésének alapja, hogy az egyik oldalon fényforrás, míg a másik oldalon fénydetektor található, mely közé egy perforált tárcsát helyeznek, amely forgás közben megszagatja a fény útját, ezáltal vagy szinuszos, vagy négyszögjel alakú impulzusokat generál. A korábban említett felbontás, azaz fordulatonkénti impulzusszám értéke akár 2000-10000 is lehet. Viszont a korábban említettnek megfelelően sokszor kétcsatornás, vagyis kvadrátikus kerék enkódereket alkalmaznak. Ezzel a forgási irány is lekövethetővé válik, valamint a felbontás is négyszereződik. Ekkor egymáshoz 90° eltolásban alkalmaznak két enkódert és a feldolgozásra szánt kimenet a 2. ábra alapján alakul. Láthatóan négy állapotot különböztetünk meg, vagyis a 2000 fordulatonkénti impulzus, ebben az esetben 8000-ré nő. (Siegwart és Nourbakhsh 2004.)



2. ábra A kvadrátikus enkóder idődiagramja és állapotai. Forrás: Siegwart és Nourbakhsh (2004.)

A szakirodalmakat tekintve Rustam és mtsai. (2022.) egy differenciális meghajtású mobil robothoz fejlesztettek önvezető algoritmust egy MPU9250-es IMU és optikai kerék enkóderek alkalmazásával. A fő kihívást a kerékcsúszások kiküszöbölése jelentette.

Madhloom, Raheem és Kareem (2022.) csupán gyorsulásmérővel egészítette ki a kerék enkóder adatokat, és egy módosított Kalman-szűrővel egy elfogadhatóbb szintre csökkentette a pozícionálási hiba értékét. Ők tehát egyszerűbb konstrukcióval dolgoztak, és sikerült körülbelül 20 centiméteres hibahatáron belül maradniuk. Ez kis távolságok esetén jelentős hibának tűnhet, de ha hosszabb távokban gondolkodunk akkor már elfogadható.

Yu és Jiang (2020.) IMU és kerék enkóder adatokat fúzionált, megalkotva egy navigációs rendszert mobil robotok számára, alacsony költségű, de pontos navigációs feladatok megoldására. Az algoritmus akadályelkerülő rendszerrel is rendelkezik, megvalósítását egy kiterjesztett Kalman-szűrő (EKF) segíti.

Fazekas és Gáspár (2024.) pedig egy neurális hálózat alapú relatív súlyozást végez az odometriai modell kalibrálására. Ezzel összességében javítható a rendszer performanciája, a zajok és a pontatlanságok kiküszöbölése érdekében.

Belátható, hogy a kerékre szerelt enkóderek használata elengedhetetlen mobilis robotok esetében, sok esetben a mozgásmodell az enkóderek alapján kerül meghatározásra. Azonban az IMU-hoz hasonlóan csupán ezen szenzortípus használata kevésnek bizonyul. Viszont, ha a már bemutatott két típust kombináljuk, akkor a kerékcsúszással kapcsolatos problémák nagyrészt ki tudjuk küszöbölni. Hiszen, ha csúszik a kerék, az odometriai értékek hamarosan irreálisan nagyra válnak, miközben az IMU, akár álló helyzetnek megfelelő adatokat is szolgáltathat.

2.1.3 GPS modul

A GPS használata ma már teljes mértékben általános a közlekedést tekintve, hiszen akár autóval szeretnénk egy célra a leghatékabban eljutni, akár bármilyen másik módon, néhány másodperc alatt megismerhetjük a teljes oda vezető utat, melynek alapja, hogy tudjuk hol is vagyunk most a világban. Ebben segít a Global Positioning System, azaz a GPS, mely egy helymeghatározó rendszer.

Először katonai használatra alakították ki, de ma már széles körben elérhetővé vált. A rendszer alapja a minden időpillanatban legalább huszonnégy működő GPS műholddal, körülbelül húszezer kilométer magasan keringő műholdból legalább négygel a Föld bármely pontjáról képesek vagyunk kommunikálni. A kommunikáció alatt azt értjük, hogy egy megfelelő vevőegységgel adatcsomagokat tudunk fogadni a műholdaktól, ami tartalmazza a műhold pozícióját a térben, illetve ennek a pozíciónak az aktuális idejét. Ezért a GPS vevőegységek passzív, exteroceptív, azaz külső érzékelőként működnek. Ha a vevőegység több műholdtól (minimum három) megszerzi az adatukat, akkor az érkezési idő különbségek alapján képes saját pozíciójának meghatározására. (Siegwart és Nourbakhsh 2004.)

Alapvetően ezzel a megközelítéssel pár méteres pontosságot tudunk elérni a pozíció meghatározása során. Amely mondjuk egy autó navigációjánál, ahol a lényeg, hogy megmondja éppen melyik utcában vagyunk egy időpillanatban, tökéletes. Viszont egy precíz lokalizációhoz önmagában kevés, hiszen lehet, hogy kicsiny mozgástér esetében jelentős hibának is számíthat.

A hiba kiküszöbölésének érdekében számtalan további pontosítási algoritmus létezik, azonban a dolgozat és alapvetően a mobil robot lokalizáció szempontjából a legfontosabb a Real-Time Kinematic (RTK), vagyis a valós-idejű kinematika rendszere. Ennek a rendszernek az alapja, hogy nem csupán egy földi állomással dolgozunk, hanem kettővel, amely közül az egyik álló (bázisállomás), a másik mozgó. A statikus állomás pontos helyét ismerjük, így ott az idő és a mozgó állomás közötti távolságot használjuk fel, hogy korrigáljuk a műholdas kapcsolat hibáit. Ezáltal a helymeghatározás hibája néhány centiméteressé csökkenthető (tipikusan 2 cm). Az RTK rendszerrel való navigációt széles körben alkalmazzák, például a legnagyobb mezőgazdasági vállalatok külön üzletágakban foglalkoznak RTK telepítéssel, üzemeltetéssel és karbantartással. (Kaplan és Hegarty 2006.)

Mobil robotok lokalizációjában is használatos, de általában nem önmagában, mivel nagy limitációnak számít az ilyen léptékben lassúnak számító 1-2 Hz-es mintavételi frekvencia. Míg odometria és IMU-k esetében akár több száz Hz is lehet ez az érték, a jelentős számítási igény, valamint a korrekciós üzenetek küldése miatt különösen az RTK-s rendszerekre jellemző, de a pontatlanabb hagyományos GPS-ek esetében is a mintavételi frekvencia maximálisan 10 Hz körül van. Dinamikus mozgások esetén a valóságban megtett út rekonstrukciója nem megvalósítható ilyen ritkán mintavételezett adatbázis segítségével. Ezért a GPS-eket sem önmagukban, hanem például a két korábban említett szenzortípussal szokták kombinálni.

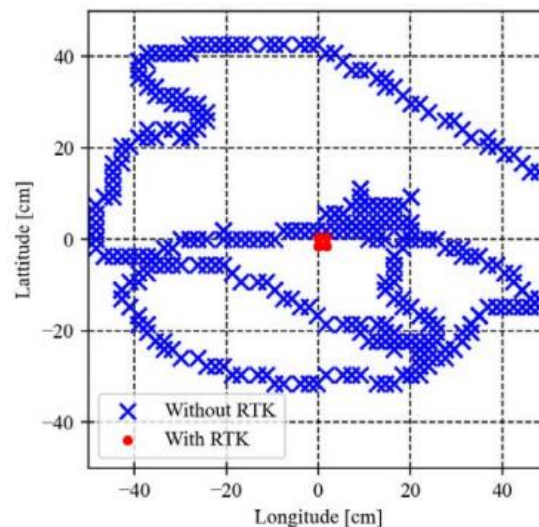
Az előző bekezdést megerősítette Panigrahi és Bisoy (2021.) munkája is, akik egy általános áttekintést készítettek autonóm mobil robotok lokalizációjának témakörében. Ebben a tanulmányban is elhangzik, hogy GPS nem képes önmagában pontos lokalizációra, kiegészítésre szorul. Emellett bemutatják valószínűség alapú környezetben a szenzoreredmények összedolgozásának algoritmusait is.

Moeller, Deemyad és Sebastian (2020.) burgonya ültetvényeken dolgoztak, ahol először egy quadkopterrel körüljárták a földterületeket és elmentették, hogy melyik pozíciókon találtak egy megadott vírussal fertőzött növényeket. A beteg növények azonosítása után egy mobil robot a földön megközelítette őket, és a fertőzés terjedésének megakadályozása miatt eltávolította őket. A rendszer egy közös bázisú RTK-s GPS-re alapul, mellyel mentik és olvassák ki a lokációkat.

Lobazev, Maslennikova és Klovov (2024.) munkájuk során egy kerekes robot platform odometriájának hibáját csökkentették RTK-s GPS használatával. Ők is az odometria pontatlanságára, megbízhatatlanságára hívták fel a figyelmet, azt próbálták minimalizálni a GPS adatok kombinációjával.

Baharom és mtsai. (2020.) is autonóm mobil robotok lokalizációjával foglalkoztak szenzor fúziós algoritmusok segítségével. Bár ez csak egy előzetes kutakodás volt, hosszú távú céljuk minimálisan 10 cm-es pontosság elérése zsúfolt környezetben. Az algoritmusukat kifejezetten a pandémiai problémák megoldására szerették volna használni, azaz minimalizálnák a fertőzött helyeken az emberi jelenlétet, helyette autonóm mobil robotok végeznék a veszélyes feladatokat.

Stangebye és mtsai. (2020.) egy saját készítésű RTK-GPS modult fejlesztettek, mivel a közelmúltban ennek költségei kisebb alkalmazások esetében is megfizethetővé csökkentek. Egy szemléletes ábrán (3. ábra) bemutatták, az RTK-s GPS-ek jelentőségét. A pontosságot vizsgálták, ahol a piros körök az RTK-s eredményeket, míg a kék 'X' szimbólumok az RTK nélküli eredményeket szemléltetik.



3. ábra Az RTK-val és nélküle használt GPS pontosságának összehasonlítása.

Forrás: Stangebye, és mtsai. (2020.)

Az előzőekben tehát bemutatásra kerültek a GPS mobil robot lokalizációjában való használatának előnyei és limitációi is. A kutatásban alkalmazott mérőrendszerben is használók egy U-BLOX márkájú RTK-GPS modult referenciakoordinátaként, valamint egy Arduinora erősíthető RTK nélküli, néhány ezer forintos GPS modult, melyek a mérőrendszer fejezetében részletes bemutatásra kerülnek.

2.1.4 LiDAR

A lézerefény alapú távolságmérést, vagyis LiDAR-t (Light Detection and Ranging) mobil robotok esetében általában akadályok, mérőföldkövek érzékelésére szokták használni, jobbra beltéren. A hangsúly nem is a belső környezeten van, hanem inkább a vizsgált területek ismerete a fontos számunkra, ha a lokalizáció és a LiDAR kapcsolatát keressük. Azért, hogy bármiféle információt szerezzünk bizonyos mérőföldkövek távolságáról, tudnunk kell azt, hogy hol is vannak. Ezáltal adottnak kell lenni egy térképnek, és azon próbáljuk meg elhelyezni a mobil robotot. Használható még SLAM probléma megoldására is, mikor a térképet a lokalizációval együtt alkotjuk meg. Mivel én teljesen ismeretlen terepeken, térkép nélkül vizsgálgódtam, és nem is volt cél meghatározni egy térképet sem, így ezen technológiát nem használtam, de az alkalmazásokban betöltött szerepe miatt említést érdemel.

A LiDAR alapú távolságmérés alapja a repülési idő (Time of Flight) alapú mérési módszer, mely más típusú társainál – például ultrahangos távolságérzékelő – hasonlóan alkalmazott. Alapvetően ezek az érzékelők két részegységből állnak, az egyik a kibocsátó oldal, mely egy lézersugarat bocsát ki abba az irányba, amelyikbe az érzékelő éppen néz. A lézerefény, ha nincs mérőföldkő a közelben, akkor nem verődik vissza, innen tudjuk, hogy az adott irányban tiszta a terep. Ha adott egy mérőföldkő, akkor viszont a lézerefény egy része visszaverődik az mérőföldkőről az érzékelő felé. Ilyenkor a második részegység, azaz a fogadó oldal érzékeli. Mindkét esemény időpontját rögzítik, emellett ismert a lézerefény sebessége levegőben, tehát számítható a mérőföldkő és az érzékelő közötti becsült távolság. Emellett található az érzékelőn egy valamilyen tartományon mechanikai forgató berendezés (például 120° vagy 180°) és egy tükrök, amellyel széles tartományban tudja a környezetét vizsgálni az érzékelő. (Siegwart és Nourbakhsh 2004.)

Meng és mtsai. (2021.) egy effektív és megbízható globális lokalizációt valósítottak meg mobil robotokon LiDAR használatával. Kifejtik, hogy beltéren a 2D, kültéren a 3D LiDAR-t érdemes használni. A lokalizációt nagyobb, strukturált környezetben valósították meg, a megvalósításhoz többszintű térképeket használtak. Szimulációs és valós környezetben is alátámasztották kutatási eredményeiket. Liu és mtsai. (2022.) egy szenzorfüzió alapuló, fejlesztett LiDAR alapú lokalizációt valósított meg. Az érzékelésre IMU-t, odometriát, és 3D LiDAR-t használtak és egy fejlesztett Adaptív Monte Carlo Lokalizációt (AMCL) valósítottak meg. Az IMU és az odometria fúziója után kerülnek felhasználásra a 3D LiDAR-ból nyert információk a pozícióbecslés javítása érdekében. Szimulációs és valós környezetben is jobb eredményeket értek el, mint a hagyományos AMCL esetében, valamint alternatívát is nyújtottak a 3D LiDAR használatával a GPS kiváltására. Belkin, Abramenko és Yudin (2021.) egy kültéri

földi robottal foglalkoztak, munkájuk során LiDAR alapú valós idejű lokalizációt és SLAM-et valósítottak meg fényviszonyoktól független módon, ezáltal algoritmusuk akár teljes sötétségben is ugyanúgy működik, mint megvilágított környezetben. A robot pozíció hibája 10 cm körül, az orientáció hibája $0,7^\circ$ környékén alakult a valós kísérletek alapján.

Az eddig bemutatottak alapján kijelenthető, hogy a LiDAR (vagy bármely másik elven alapuló távolságmérő rendszer) széles körben használható mobil robotoknál, viszont lokalizáció esetében ismert térképre és mérföldkövekre van szükségünk, amikhez képest relatív távolságméréssel tudjuk a robotot elhelyezni a síkban. Mivel a dolgozatban megvalósított kutatásban ilyen problémamegoldásra nem volt szükség, ezért nem is került használatra semmilyen távolságmérő érzékelő és ezáltal relatív pozíciómérés sem valósult meg.

2.1.5 Kamera

A kamerák alkalmazása is elterjedt mobil robotok esetében hiszen, ha a fényviszonyok is megengedik, akkor egy külső objektumról a legjobb minőségű felvételeket, rekonstrukciót kamerák segítségével tudjuk megalkotni. Legyen szó akár mérföldkőről, akár akadályról, kamera felvételekkel és jó feldolgozó egységgel akár dinamikusan mozgó tárgyak, méretezett 3D észlelésére is van lehetőségünk, mely egy autonóm rendszer esetén az ütközés elkerülési feladat alapja. Természetesen van lehetőség lokalizáció vagy SLAM feladat megoldására is pusztán kamerák, vagy például LiDAR-ral fuzionált kameraadatok alkalmazásával is, de a fő feladata talán a korábban említett ütközés elkerülési problémák megoldásában keresendő. A következőkben kifejtésre kerülnek a legelterjedtebb ipari kamerák rekonstruálási módjai, ezek előnyei, hátrányai, majd pár ipari alkalmazás is ismertetésre kerül.

Az egyik lehetőség 3D rekonstrukcióra sztereó kamerák alkalmazása, amikor minimum két kamerával készítünk képeket, majd a közös pontok azonosításával számíthatóak a távolságok a képeken. Előnye a pontosság, hátránya a nagyobb számítási komplexitás, a drágább eszközhasználat. Az újabb kamerák esetében, akár egy tokozásban is lehet sztereó kameránk, nem szükséges fizikailag elválasztani a két kamerát. Egy másik lehetőség szerint a LiDAR-nál ismertetett TOF technológiát is használják, előnye a gyorsaság, kompaktság, viszont sokkal zajosabb a 3D rekonstrukció. A következő technológia a strukturált fény kivetítése, mikoris egy pontfelhő vagy négyzetrács formájú fényt vetítenek az akadályra, és annak deformációja alapján történik a 3D képalkotás. Előnye a sokkal pontosabb rekonstrukció, viszont nagy méretű eszközöket igényel, ez korlátozhatja az alkalmazhatóságot kisebb mobil robotok esetében. Alkalmazásunktól függően választhatjuk ki, hogy nekünk melyik a megfelelő típus. (Pérez, és mtsai. 2016.)

A gyakorlatban alapvetően RGB kamerákat, multispektrumú kamerákat és mélységi kamerákat használnak. RGB kamerák esetében mindhárom alapszín érzékelése valósul meg, majd a végleges kép ezek kombinációjával alakul ki. Multispektrumú kameráknál az érzékelés nem áll meg a látható tartomány hullámhosszán, hanem ezen kívül is érzékel, ezáltal számunkra nem látható adatokat is elérhetővé tesz. A mélységi kamerák pedig a 2D képhez hozzá teszik a távolság dimenziót, vagyis képeik sztereó hatást keltenek. (Li és Liu 2019.)

Ishihara és Takahashi (2022) kép alapú mobil robot navigációs algoritmust vizsgáltak, csak előre néző, omnidirekcionális, valamint mennyezetre néző kamerák alkalmazásával. Az előre néző kamerának mindig változik a nézőpontja, így nem homogén az előrejelzésük, ezt próbálták kiküszöbölni. Lee és Ahn (2020.) egy olyan alacsony költségű rendszert fejlesztettek, amely a korábbi drága és nagy költségvetésű, Kinect kamera alapú valós idejű emberi cselekvés detektálást váltaná ki mobil robot platformok esetében. Az emberi cselekvések észlelése és rá való reagálás a balesetek elkerülése érdekében fontos. A tanulmányban csupán egy RGB kamera és konvolúciós neurális hálózat alkalmazásával, az eddigieknél kevesebb költséggel valósították meg a feladatot. A rendszer másodpercenként 15 képkockát elemesz, így a beavatkozási idő emberi mértékben jelentős gyorsasággal rendelkezik. Skoczen és mtsai. (2021.) mezőgazdasági mobil robot akadály elkerülő rendszerét fejlesztették, RGB-D (RGB és mélységi kamera) kamerák segítségével. A mezőgazdasági robotalkalmazások mindig strukturálatlan környezetben, nem szokványos statikus és dinamikus akadályokkal kell megküzdeni. Ebben az esetben egy autonóm fűnyíró berendezésnél, a munkaterület pontos meghatározására, az azon álló akadályok (oszlopok) vagy a munkaterületre belépő élőlények azonosítására és kikerülésére alkalmazták a kamerás rendszert. Mu és mtsai. (2020.) egy SLAM feladat megoldására használt mélységi kamerákat. Mivel a kamerák ehhez nem szolgáltatnak elég információt, ezért az adatokat fúzionálták LiDAR-ral, enkóderrel és IMU-val. Pontosság növekedést és szélesebb alkalmazási lehetőséget kaptak ezzel a megoldással, miután UKF típusú Kalman-szűrőt alkalmaztak, a nemlineáris rendszerük elemzésére. Kísérleteiket szimulációban és fizikai körülmények között is validálták.

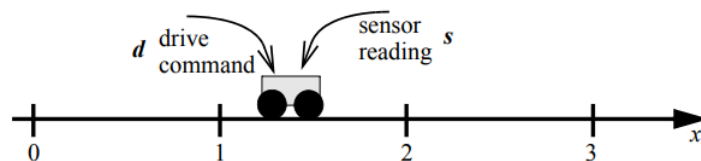
A szakirodalmak alapján a kamerák alkalmazása kiemelt fontossággal rendelkezik, főleg autonóm rendszerek navigálásában vagy akadályelkerülésben. A kutatás során a navigációt nem autonóm módon, hanem Joystick segítségével végeztem, míg akadályelkerülésre nem volt szükség, így kamera adatokat nem használtam, bár a Magni robotnak van beépített kamerája. A későbbiek során, ha autonóm rendszert szeretnék belőle készíteni, akkor mindenképpen használni fogom ezt a berendezést.

2.2 Szenzorfüziós módszerek

Akár egy többszáz oldalas könyvet lehetne szentelni a különféle szenzorfüziós algoritmusok bemutatására, viszont ebben a fejezetben inkább azokat a fejezeteket fogom kiemelni, amelyek közvetlenül kapcsolódnak majd a dolgozat alapkoncepciójához. Ezáltal lesz egy kis betekintés a valószínűség alapú lokalizáció témakörébe, majd részletesebben kifejtésre kerülnek a különböző Kalman-szűrő típusok. Kisebb betekintést is szeretnék nyújtani a szakmai folyóiratcikkekből, amelyek a jelenlegi és a jövőbeli szenzorfüziós alkalmazásokkal és problémamegoldásokkal foglalkoznak. Már a korábbi fejezetben, ahol bemutatásra kerültek az egyes technológiák külön-külön, szinte csak és kizárólag példák kerültek bemutatásra, ahol szenzorfüziót alkalmaznak, hiszen manapság már csupán egyféle szenzortípus alkalmazására nem szokott sor kerülni megbízhatatlansága miatt. Ebből kifolyólag azon folyóiratcikk ebben a fejezetben is szerepelni fognak példaként, csak most nem egy-egy technológia lesz belőle bemutatva, hanem az alkalmazott szenzorfüziós algoritmus.

2.2.1 Valószínűség alapú megközelítés

A valószínűségeen alapuló lokalizáció alapja, hogy minden szenzor és aktuátor bizonyos mértékű hibával képes csak működni. Ha például 1 métert kellene mozogni, akkor lehet, hogy 1,0001 métert, de lehet, hogy 0,99999 métert sikerül ebből megvalósítani. Itt tehát fellép a bizonytalanság fogalma, amely a hagyományos robotikában figyelmen kívül hagyásra kerül. Erre az feltételezésre mutat példát a 4. ábra, ahol a d betűvel jelölt nyíl az aktuátor által meghatározott mozgás pontos végrehajtása esetén jelölné a pozíciót, valamint látható rajta s , ami pedig a szenzoradatok értelmezésével alakul ki. (Bräunl 2006.)



4. ábra A pozíció bizonytalan meghatározása. Forrás: Bräunl (2006.)

Valószínűségi alapon pedig egy átlagérték és egy valamilyen intervallum segítségével egy pont helyett akár tágabb tartományon is leírhatjuk a robot pozícióját, amelyre megadható egy konfidenciaszint, mely azt a bizonyossági szintet jelenti, amekkora valószínűséggel vagyunk a tartományon belül. Az intervallum becslésre lehet használni, például valamilyen alsó és felső értékkel jelölt terjedelem intervallumot, de általában szórási értéket szoktunk megadni. Lineáris esetben sokszor használunk egy feltételezett robotpozícióra Gauss-os eloszlású intervallumot, mely a jól ismert haranggörbe alapján írja le a robot helyzetének valószínűleges értékét. Ez megadható egy mozgásmodell (odometria) esetében, ahol tehát van egy becslésünk,

mely kizárólagosan a mozgás alapján történik. Ezt a becslést a priori becslésnek nevezzük. Ha az a priori becslés, azaz a megvalósított mozgás alapján valamilyen szenzorral vizsgáljuk a környezetet további információkat nyerhetünk, amelyekkel a megfigyelési modell értékeit kapjuk. Ha az a priori és a megfigyelési modell értékeit összemossuk, akkor kapjuk a korábbinál pontosabb becslést a robot pozíciójára, azaz a posteriori becslést. Az egész rendszer a feltételes valószínűség alapú Bayes-tétel alapján származtatható. A robotot n . pontban lévő helyzetének a priori becslére meghatározható, ha az $n-1$. pontban mért posteriori becslést meghajtjuk a mozgás modellel. Ezáltal egy feltételezett pontot kapunk, melyre alkalmazzuk a megfigyelés modellt, és megkapjuk az n . pontra a posteriori becslést. Leírva ez igen bonyolultnak hangzik, viszont ezt az alapfeltételezést nagyon könnyű implementálni. (Thrun, Burgard és Fox 2005.)

A fenti algoritmusnak nagy szerepe van a zajok kiszűrésében, és a fontos adatok előtérbe hozásában, éppen ezért az így megvalósított algoritmusokat sokszor szűrőköröknek nevezzük. Alkalmazásuk azért is fontos, mert egyféle keretrendszert alkotnak, megadják, hogy hogyan hasznosítsuk azt, hogy több, különböző forrásból származó információ áll rendelkezésre.

Ezek közül számunkra a legfontosabb talán a Rudolf E. Kalman, magyar származású kutató munkája során lefektetett Kalman-szűrő keretrendszer, amelynek több változata is részletesen kifejtésre kerül. Ez a szűrőtípus is a korábban kifejtett gaussi szűrőkhöz tartozik. Ezen a típuson kívül léteznek paraméter nélküli szűrők, például a hisztogram-szűrő vagy a részecske-szűrő, melyek szintén valószínűségi alapra helyezik lokalizáció feladatát, de nem normális eloszlást (Gauss-i) használnak a robot állapotának leírására. (Thrun, Burgard és Fox 2005.)

A kutatási részben Kalman-szűrőt fogok használni, így annak bemutatása sokkal hangsúlyosabb lesz, mint például a hisztogram- vagy részecske-szűrőé, mert ezek alkalmazása kissé eltér a témától, strukturált környezetben, vagy SLAM esetében jobban alkalmazhatóak.

2.2.2 Kalman-szűrő

Az 1950-es években, Rudolf Emil Kalman, magyar származású, de USA-ban dolgozó kutató a Bayes-tétel alapú szűrők közül talán a mai napig legjobb implementációt hozta létre lineáris rendszerek szűrésére és előrejelzésére. A neve Kalman-szűrő (KF), melyet később nemlineáris rendszerekre is megalkottak. Az algoritmus alapvetően öt különböző lépésből áll, melyet két fázisra szoktunk bontani. Első lépésben minden adott t időpillanatban egy feltételezett állapotvektort határozunk meg μ_t , azaz az átlag és Σ_t , azaz a kovariancia segítségével. A priori és a posteriori becslések normális eloszlásúak (Gauss), melyet három Markov-feltételezés biztosít. Emellett mind a mozgás modell alapján következtetett, becsült következő állapot kovarianciája, mind a mérési modell alapján frissített következő állapot

kovarianciája gaussi fehér zajjal terhelt. A fehér zaj definíció szerint 0 várható értékű (átlagú) valamekkora szórású zaj, melyet a becsült vagy frissített állapothoz adva nem változtat a becslés várható értékén, csak a bizonytalanságot veszi bele valamekkora intervallummal. Az algoritmust röviden az 5. ábra mutatja be, amely alatt részletesen kifejtésre kerül. (Thrun, Burgard és Fox 2005.)

```

1:   Algorithm Kalman_filter( $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$ ):
2:      $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$ 
3:      $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$ 
4:      $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$ 
5:      $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$ 
6:      $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$ 
7:     return  $\mu_t, \Sigma_t$ 

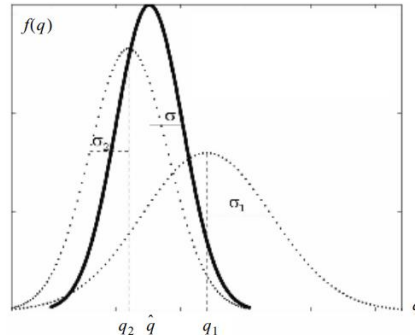
```

5. ábra A Kalman-szűrő lépései. Forrás: Thrun, Burgard és Fox (2005.)

Minden sorban különféle mátrixokkal való szorzások vannak (például A_t és B_t), melyek feladatspecifikusak, ezért az itt külön nem kerül kifejtésre, csak az elvi megvalósítás alapján kerül bemutatásra az algoritmus. Az első sorban láthatjuk a bemeneteket zárójelben, rendre az előző időpillanatban az állapot vektor (2D síkmozgás esetén x és y koordináta, valamint θ orientáció), az előző állapothoz tartozó kovariancia mátrix, a beavatkozás modellje (mozgásmodell) és a megfigyelési modell. A második sorban meghatározzuk a priori becslést, az előző állapot és a beavatkozás (valamint két mátrixszal szorzott értékének) összegével. Ezt követően a harmadik sorban meghatározzuk a priori kovariancia értékét is, melyhez itt kerül hozzáadásra a gaussi fehér zaj (R_t). Majd felhasználva a priori kovarianciát, a mérési modell szorzójaként szolgáló C_t mátrixot és a mérés hibáját Q_t formájában, a negyedik sorban meghatározzuk a Kalman erősítést, melyet K_t jelöl. Ez az erősítés veszi figyelembe az adatok pontosságát a kovarianciák mértékével, amely a frissítési fázis alapjaként szolgál a továbbiakban. Ha a mérés pontatlan lenne – például GPS esetén zavarás miatt hibás a mérés – akkor az a posteriori becslésen nem kell, hogy változtasson a priorihoz képest, ezt ebben az erősítési mátrixban állítjuk be. Végül az utolsó két (a return előtt) meghatározzuk a posteriori állapotbecslést és a kovariancia értékeket, amelyek majd a következő iteráció bemeneteként fognak szolgálni, valamint a végleges, javított állapotbecslést adják a robot pozíciójára.

A Kalman-szűrő tehát két szenzor adat fuzionálására alkalmazható, az első általában odometria alapú, tehát vagy enkóder adatok alapján számolt mozgással, vagy a sebesség alapú modell esetében, például differenciális meghajtás esetében lineáris sebesség és szögsebesség értékeket alakítunk át pozíció adatokra. Viszont a kerékcúsúszásokból származóan ez az odometria igen pontatlan, ezt jelzi a 6. ábrán q_1 . A Kalman-szűrő segítségével viszont egy

másik, mérésen alapuló módszerrel kiküszöbölhetjük az odometria pontatlanságát. Például GPS adatok segítségével egy külön mérést végzünk a robot feltételezett pozíciójára, ez a 6. ábrán q_2 . A kettőt a megfelelő beállításokkal $\hat{q}$ becsléssé alakíthatjuk, mely közelebb helyezkedik a valósághoz, mint a kettő külön-külön. (Siegwart és Nourbakhsh 2004.)



6. ábra A Kalman-szűrő kimenete. Forrás: Siegwart és Nourbakhsh (2004.)

A korábban említettnek megfelelően az alap Kalman-szűrő lineáris, gaussi zajjal rendelkező rendszerek esetében alkalmazható. Nemlineáris esetben használható a Kiterjesztett Kalman-szűrő (Extended Kalman Filter – EKF) algoritmus. Az algoritmus alaplépéseiben megegyezik a lineáris esetben alkalmazott szűrőkörrel, csak alkalmazás előtt a rendszert linearizálni kell. A linearizáláshoz a meg kell határozni a rendszer mátrixok (A, B, C) Jacobi-mátrixát (G és H). A Jacobi-mátrixok olyan lineáris közelítést adnak a rendszerről, amelyre kielégítő pontossággal alkalmazható a KF algoritmus. (Thrun, Burgard és Fox 2005.)

Egy másik lehetőség a nemlinearitások kezelésére az úgynevezett Unscented Kalman Filter (UKF) algoritmus, amely magyar fordítását nem nagyon használják, de valami olyasmi lehetne, hogy „előre nem meghatározott Kalman-szűrő”, mely a technikájára utal, de inkább UKF-ként fogok rá hivatkozni. UKF esetében nem történik linearizálás, ezért összességében pontosabb eredményre képes az EKF-nél. Az algoritmus alapja az unscented transzformáció, amely az állapot térben meghatározott sigma-pontokat választ és ezek alapján becsüli a nemlineáris függvények hatását. (Panigrahi és Bisoy 2022.)

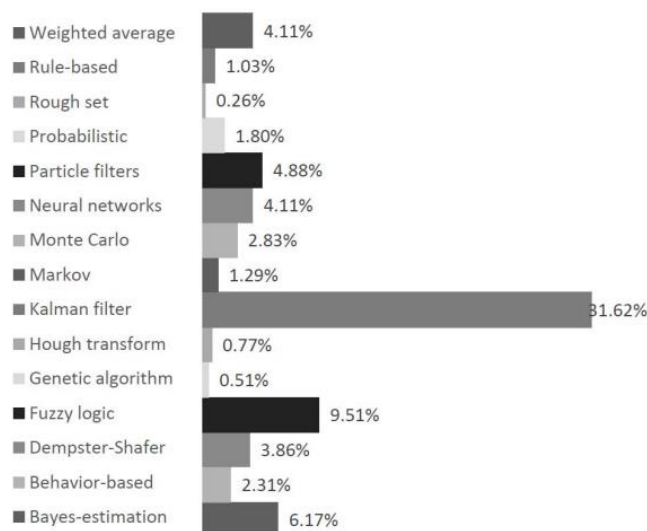
A korábban is használt tanulmányok közül példaként hoznám Madhloom, Raheem és Kareem (2022.) munkáját, akik pozícióbecslésre használtak egy Kalman-szűrőt, ahol az első fázisban kerék enkódereket, a frissítési fázisban gyorsulásmérőt használtak. A szűrőkör alkalmazásával 0,2 m pontosságot értek el különböző környezetekben. Li, Su és Yang (2019.) is hasonlóan az alap Kalman-szűrőt használták munkájuk során, itt a mozgásmodell odometriával támogatott IMU szenzorral valósult meg. A frissítési fázishoz itt LiDAR-t használtak, ahol a pozicionálási hiba 0,1 m körül alakult. Yu és Jiang (2020.) már az EKF algoritmust használták, ahol szintén odometria és IMU volt az első fázisban, míg a mérési

modellt kamerával valósították meg. Liu és mtsai. (2022.) EKF-et használ, IMU és odometria, valamint 3D LiDAR alkalmazásával, melyet később Monte-Carlo lokalizáció során hasznosítanak, itt a pontosságuk 0,02 m körül alakul. Mu és mtsai. (2020.) pedig UKF és EKF algoritmust is használ, az eddig említett szenzorok mellé még egy mélységi kamerát is alkalmazva. Az EKF algoritmusuk hibája 0,083 m környékén alakult (X irányban), míg ugyanezen érték UKF használata esetén 0,036 m, tehát jelentős különbségek vannak az eredményben.

Az előzők mellett Margin, Brito és Todt (2019.) a szenzorfüziós algoritmusok kerek mobil robotokon való lokalizációjával kapcsolatosan készített egy kutatást arról, hogy a kutatók az elmúlt harminc évben milyen tanulmányokat készítettek a témával kapcsolatban. Ez megmutatja, hogy magasan a Kalman-szűrő a legmagasabb arányban alkalmazott algoritmus. Huggi és mtsai. (2021.) IMU és GPS adatokat fuzionáltak mobil robotok esetében, Kalman-szűrő alkalmazásával. Az adatok feldolgozására szintén MATLAB-ot használták, amely ebben a dolgozatban is alkalmazott. Khairandish és mtsai. (2023.) két IMU és enkóder segítségével valósítottak meg egy olyan multimodellű Kalman-szűrőt, mely használ EKF-et és UKF-et is, 4 kerekű mobil robot esetében. Az általuk fejlesztett algoritmus performanciája jobb volt, mint a hasonló, külön-külön alkalmazott Kalman-szűrős technológiák.

2.2.3 További módszerek

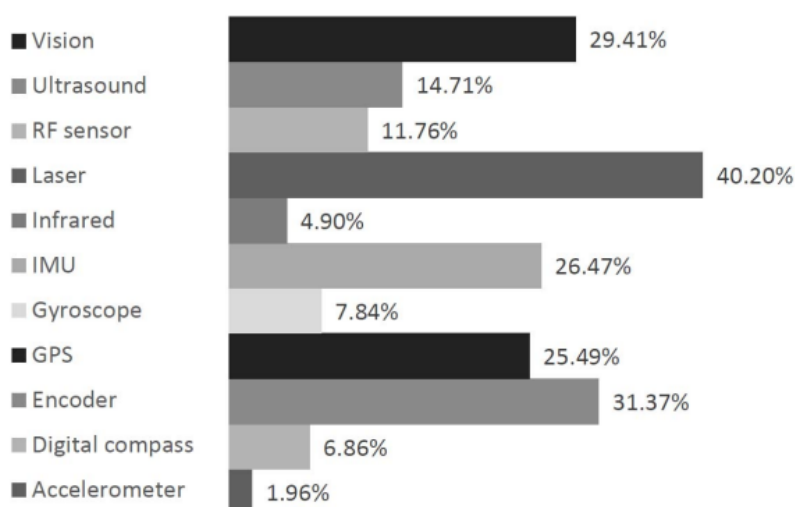
Ugyan a Kalman-szűrő magasan a legnagyobb arányban alkalmazott szenzor-fúziós algoritmus, számos más algoritmust is használnak mobil robot lokalizáció megvalósításához. Ezt az állítást erősíti a 7. ábra is, mely százalékos arányban mutatja az alkalmazott technológiákat.



7. ábra Az alkalmazott szenzorfüziós algoritmusok megoszlása a mobil robotok lokalizációja esetében. Forrás: Margin, Brito és Todt (2019.)

Az ábráról azonnal leolvasható a Kalman-szűrő egyeduralma, melyet kiegészít körülbelül 10%-os használati aránnyal az elmosódott halmazok logikáját használó, szabályzásoknál gyakran alkalmazott Fuzzy logika. A Fuzzy szabályrendszer könnyen alkalmazható és programozható algoritmusokat biztosít, ahol a felhasználó heurisztikus módon alkot egy szabályrendszert tagsági függvények segítségével. A Fuzzy logika empirikus megközelítésben könnyen jó eredményt adhat, de megfogalmazása jóval egyszerűbb a Kalman-szűrőnél. Az egyszerű Bayes-becsítő is felfért a dobogóra, mely a Kalman-szűrő alapjául is szolgál. Ezek mellett kiemelnék még három gyakori algoritmust: a neurális hálózatokat, mely működésének alapja az emberi idegrendszerből átvett metódusokat alkalmazása; a részecskeszűrőt, mely a számos lehetséges scenáriót részecskékkel, és azokon végrehajtott mozgásmodellel reprezentál; végül a súlyozott átlagot, mely egyszerű megközelítésre vezet vissza a bonyolult számításokat.

Ha a 2014-2019-es időszakra gondolunk, és megnézzük a szenzorfüziós algoritmusok során használt szenzorok eloszlását (8. ábra), akkor láthatjuk, hogy az általam bemutatott szenzorok (2.1 fejezet) mind előkelő helyen szerepelnek.



8. ábra 2014-2019 között alkalmazott szenzorok szenzorfüzió során.

Forrás: Magrin, Brito és Todt (2019.)

3. MÉRŐRENDSZER ÉS ADATGYŰJTÉS

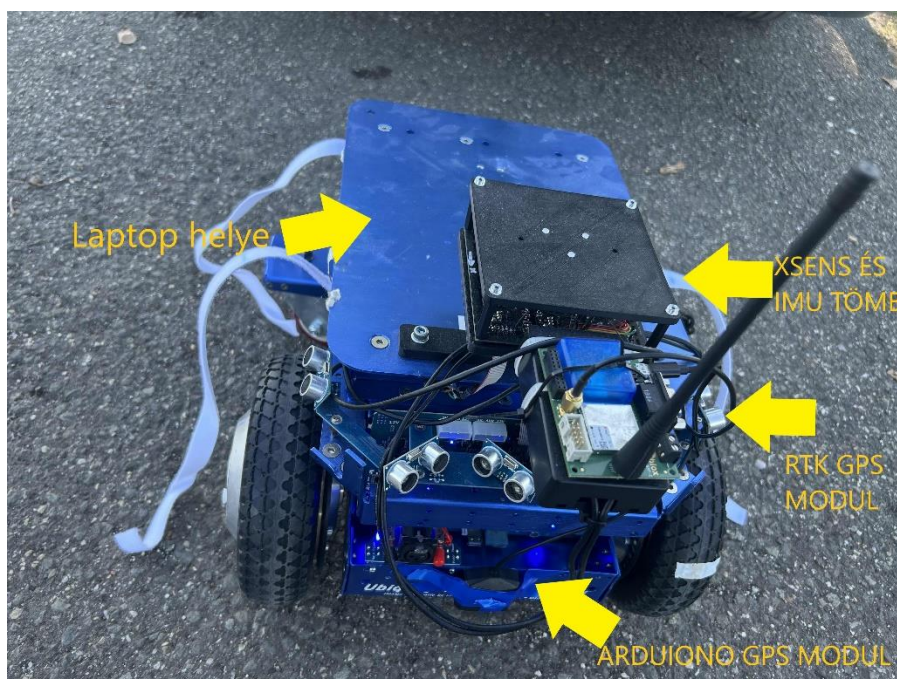
Ebben a fejezetben bemutatásra kerül a megépített mérőrendszer, az összes kiszolgáló egységével együtt, valamint bemutatom az adatgyűjtés folyamatát és a hozzátartozó programkódokat. Részletes bemutatást készítek az alkalmazott szenzorok pontos típusairól, a lépésekről, amelyeket el kellett végezni a mérés megvalósításához. Az adatok feldolgozásáról, alkalmazott képletekről, eredményekről ebben a fejezetben nem fog szó esni, azzal majd a következő fog foglalkozni.

3.1 Általánosságban

A mérőrendszer alapja egy, a Ubiquity Robotics által fejlesztett Magni Silver mobil robot. Erre a mobil robotra kerültek fel a következő szenzorok:

- U-BLOX C94-M8P-3 típusú RTK-s GPS modul mozgó állomása (A bázisállomás fix helyen, általában egy autó tetején került beállításra)
- U-BLOX NEO-6 Arduinohoz illeszthető GPS modul
- XSENS MTi 670 típusú inerciális mérőegység, mely a referencia mozgás mérését szolgáltatja. (Grund Truth koordináták)
- 9 darab, 3x3-as elrendezésű ICM-20948 alacsony-költségű inerciális mérőegység

Ezek mellett a mobil robotba bal és jobb kerekének motorjába beépített kerékenkóder méri az odometriát. Valamint a kiadott elvárt sebességparancsok is rögzítésre kerülnek ROS-on keresztül a cmd_vel topicban.



9. ábra Áttekintő kép a megvalósított mérőrendszerről

A 9. ábra egy áttekintő képet mutat a megvalósított mérőrendszerről, kiemelve az általam ráépített részeket.

A mobil robot vezérlése a Robot Operating Systemben Noetic disztribúciójával, Ubuntu 16.04-es Linux alapú operációsrendszerrel valósítható meg.

Az adatok gyűjtésének számításigényesebb része egy Asus TUF Gaming FX505G lappal történik.

A mobil robottal való manőverezésre egy Logitech gyártmányú F710-es vezeték nélküli kontrollerral van lehetőségünk, mely a ROS-on keresztül futtatott Teleoptwist csomag futtatásával van lehetőségünk.

Az adatgyűjtés a következő frekvenciákkal történik:

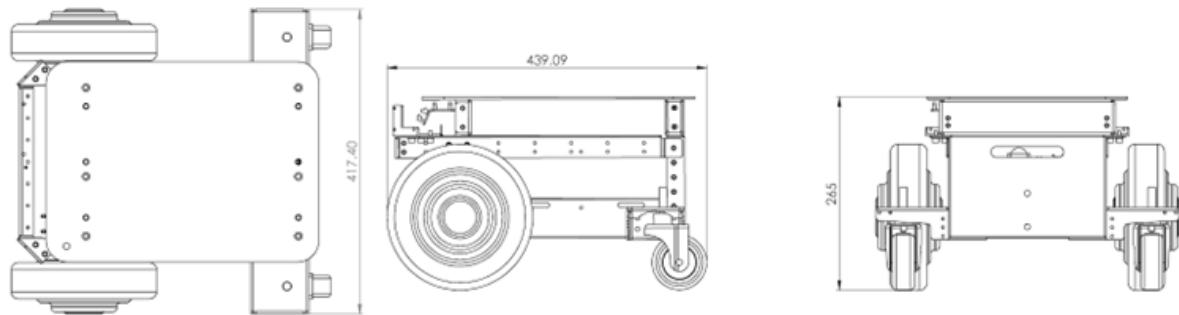
- Odometria, kerekenkóderek: 20 Hz
- U-BLOX GPS RTK módban: 1 Hz
- U-BLOX M6P GPS Modul: 1 Hz
- XSENS IMU: 100Hz
- 9 darab ICM-20948 IMU: 80 Hz

Először elég kevertnek tűnik a mintavételezés, de ha megnézzük a limitációkat akkor érthető lesz. A GPS-nek ez a maximális mintavételi frekvenciája RTK-s üzemmódban (nélküli is maximum 5 Hz lenne), így evidens ennek használata. A másik pontatlanabb GPS esetében is ezt alkalmaztam, hogy a GPS modulok illeszkedjenek egymáshoz. Az odometria a Magni robot gyártói limitációját használja, mely 20 Hz, és nem is lehet rajta. Az XSENS IMU 100Hz-e az adatlapjában írt stabilan tartható frekvencia, amely maximálisan ennek 4-szerese is lehetne, viszont akkor sokkal nagyobb baud-ratere lenne szükség. A Grund Truth koordinátáknak azonban ez a frekvencia is tökéletes. Az összerakott IMU tömbnek 80 Hz volt a maximális, odometria sokszorosával egyenlő, de még a rendszer által éppen elbírt frekvencia, itt ugyanis a 9 IMU adata egy csatornán keresztül folyik át, tehát ez a mintavételezés 9-szer 80, azaz 720 Hz. Mivel egyenként 9 „tengelyes” IMU-król beszélünk, valamint időbélyeggel és pár kiegészítő adattal minden üzenetet ellátunk, igen nagy teljesítményre készítettük így is a laptopot. A /cmd_vel topic rögzítése (mely majd a mozgásmodellt adja a feldolgozás esetében) 75 Hz-el valósult meg.

3.2 Ubiquity Robotics – Magni mobil robot

Az Ubiquity Robotics Magni mobil robotjával valósult meg a kutatási folyamat mozgó adatgyűjtés része. Ebben a fejezetben bemutatom mit érdemes tudni magáról a robotról, illetve milyen megoldásokat használtam a mérési rendszer kiépítéséhez a mobil roboton.

A robot alaprajza a 10. ábra alapján alakul, látható az ábrán a befoglaló mérete mm mértékegységben: 417,4 x 439,09 x 265 mm (sz/h/m). Kompaktsága miatt az egész mérőrendszer egy autó csomagtartójában elfér, és könnyen szállítható a mérési terepre.



10. ábra A Magni robot befoglaló méretei és alaprajza Forrás: Ubiquity Robotics (2022.)

A robot maximális teherbírása 100 kg, melynek természetesen a közelébe sem kerültem a mérés során, kiszolgáló egységekkel maximum 2 kg-mal terheltem meg a robot 13,5 kg (akkumulátorok nélküli) alapsúlyát. A meghajtása 2 darab 200 W motorral történik, differenciális hajtás formájában, mellyel maximálisan 1 m/s sebességet tud elérni, amely egy szoftveres limitáció, a motor természetesen ennél nagyobb sebességre is képes. A robotban két sorba kötött 12 V-os 7 Ah-s akkumulátorral van szerelve, így 24 V-os rendszerrel dolgozik. Az alapból elhasznált kimenetek mellett sok szabad 12 V-os és 5 V-os kimenettel is rendelkezik a kiszolgáló egységek táplálására. A robotra egy kamera – egy Raspberry Pi Camera Module V2 – és 5 ultrahangos távolságérzékelő (HC-SR04-es típusú) is fel van szerelve, de ezek nem kerültek felhasználásra. A robot rendelkezik egy Raspberry Pi 4-es számítógéppel is, melyet egy 4 magos ARM A9-es processzor hajt meg. Az operációs rendszere a korábban említett Ubuntu 16.04-es szoftver, melyre a ROS Noetic disztribúciója is fel van telepítve. További USB és Ethernet portok is találhatóak a számítógép oldalán a megfelelő kommunikációs protokollok alkalmazására a szenzorok esetében.

A robot sebessége 0-1 m/s intervallumon, szögsebessége 0-0.5 rad/s intervallumon állítható, mely alapvetően a programkódban megadott értéket veszi fel. A mérés kezdésekor a Teleop Twist Keyboard csomag teleop_twist_keyboard.py fájlját futtatva billentyűzet segítségével vezethetjük a robotot, és külön állíthatjuk a sebességet. Ez a konstrukció nem túl előnyös abból a szempontból, hogy a billentyűzet gombjai kétállapotúak (vagy be van nyomva, vagy nem), mely nem túl sok gyorsulási és lassítási lehetőséget biztosít a robot számára, amely kedvezőtlen IMU-k használata esetében. Ezt kiküszöbölendő beszerzésre került egy Logitech F710-es Kontroller vagy Joystick, amely analóg karja 10 részre osztja a gyorsulási tartományt, így azt billegtetve folyamatos gyorsulást és lassulást érhetünk el a robottal, amely folyamatosan

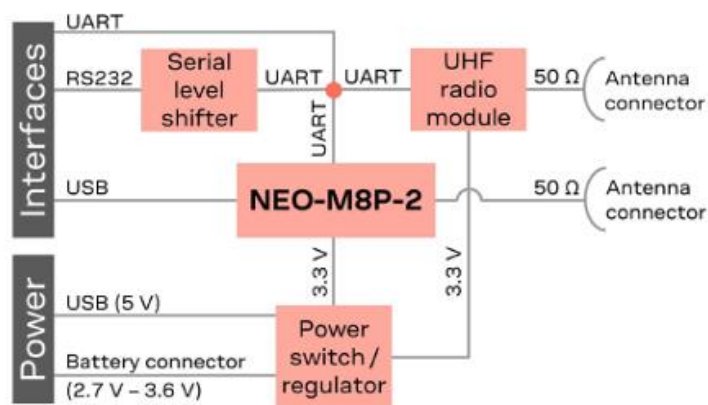
„mozgásban tartja” a gyorsulásmérő szenzorokat, amelyek az IMU-ba vannak integrálva. A magni_teleop könyvtárban található egy logitech.yaml paraméter fájl (1. sz. melléklet), mely segítségével beállítható könnyedén a maximális lineáris sebesség és szögsebesség értéke. A kontroller bal felső sarkában található „Deadman switch” gomb megfelel a robotkarok esetében használatos társának. Ha vezérelni szeretnénk a robotot, azt a gombot mindig le kell nyomni, és akkor aktiválódik az bal alsó analóg kar. Emellett található a kontrolleren a Deadman switch alatt egy turbó gomb, mely egy gyorsabb vezérlést biztosít, alapvetően 25%-kal gyorsabb, viszont én ezt teljesen átkonfiguráltam a mellékletben található forráskódnak megfelelően.

A programozás és a robot megismerése során a leghasznosabb forrás a Magni robot online dokumentációja volt, amely elérhető a Ubiquity Robotics honlapján. Bármely beüzemelési gond volt, vagy kérdés merült fel ezen honlapon mindig gyorsan sikerült választ találni a kérdésekre. (Ubiquity Robotics 2022.)

3.3 U-BLOX C94-M8P-3 RTK GPS Modul

A következő szenzor, amely a mérőrendszerbe került az egy U-BLOX C94-M8P-3-as RTK-s GPS modul, mely segítségével másodpercenként frissülve néhány centiméter pontosságú lokalizációt lehet elérni.

A szenzor két teljesen ugyanolyan egységgel rendelkezik, melyek közül egyet bázisállomásnak, egyet mozgó állomásnak kell felhasználni RTK-s üzemmódban. A szenzor típusában az utolsó 3-as számjegy jelenti, hogy az európai variánst használjuk, amely annyiban tér el a többi variánstól, hogy a két egység közötti rádióhullámú kommunikáció frekvenciája tér csak el. Ennek oka, hogy nincs olyan rádióhullámú frekvencia, amely mindenhol szabadon használható erre a célra. Európában a 433 MHz-et használja a modul. A modul alapkoncepciója az, hogy a bázisállomás pozíciója ismert, a mozgóállomás pedig a műholdjelek és a bázisállomás relatív pozíciója alapján képes korrigálni korábbi pár méteres pontosságát, pár centiméterre ideális, tiszta égi viszonyok között. A 11. ábrán látható egy egység blokkdiagramja, a használathoz a Power interfészek közül kell az egyiket használni a tápellátásra. A bázis állomás esetében nem kerül felhasználásra egyetlen kimeneti adat sem, csak a két egység kommunikál egymással, ezért elegendő egy Power-Bank használata, Micro-USB kimenettel a tápellátásra. A mozgó állomás esetében a tápellátás mellett szükség van adatgyűjtésre is, így itt a Laptopból kerül megáplálásra az egység. Valamint az RTK használat miatt szükség van mindkettő Antenna konnektor használatára is, a felső az UHF rádió konnektorra és az alsó a műholdas vevőegységre. (U-BLOX AG. 2021.)



11. ábra Az alkalmazott GPS modul blokkdiagramja. Forrás: U-BLOX AG. (2021.)

A következő lépés a szenzorok konfigurációja, mely azzal indult, hogy a bázisállomást megjelöltem, hogy a két különböző állomás azonosítható legyen. Ezután a 12. ábrának megfelelően a három csatlakozót be kell kötni. Ezt követően el kell végezni, majd követni kell a „C94-M8P Application Board Setup Guide” forrás utasításait. (U-BLOX AG. 2017.)



12. ábra A GPS modul bekötése. Forrás: U-BLOX AG. (2021.)

Ha minden sikeres volt, akkor a kék GPS jel LED mellett a mozgó állomáson időnként fel kell villannia a zöld LED-nek is, amely az RTK-s korrekciós jelek sikeres fogadására utal.

Ezután a mozgó állomás teljesen készen áll a munkára, viszont a bázis állomásnál a konfiguráció egy részét minden mérési helyszínen meg kell ismételni, mivel a fix pozícióját a modulnak ki kell mérni, a csak Windows operációs rendszeren elérhető U-BLOX U-Center szoftverének segítségével. Ez a művelet 5-10 percet vesz igénybe, ahol a survey-in algoritmussal súlyozott átlagot számít a program a mért pozíciókra az említett időtartamban. Be kell állítani a fent említett időkorlátot, és egy távolságkorlátot is, amely terepfüggő, 1-2 m között lehet optimálisan, és mindkettő teljesítésével lép ki a konfigurációs algoritmusból, meghatározva a fix pozíciót. Ezután a GitHubon elérhető egy – a GPS modulhoz tartozó – ROS driver (KumarRobotics/ublox), amellyel majd az adatgyűjtést valósíthatjuk meg. Itt állíthatjuk

be a pontos típust, és írhatunk egy szkriptet (mely a 3.6-os fejezetben kerül bemutatásra), amivel az szenzor mérését fogadni és menteni tudjuk.

A mérőkör referencia koordinátáit (Ground Truth koordináták) ezen GPS segítségével fogom megkapni, hiszen pontossága néhány centiméterre tehető, tiszta ég alatt referencia mérések sorozatával akár 1 centiméteres pontosságot is sikerült elérnem. Ezért joggal foglalhatja el a referencia koordináta szerepét interpoláció segítségével.

3.4 UBLOX NEO-6M GPS Modul

A Kalman-szűrő megfigyelésmo­delljeként pedig egy olcsó, akár már 2-3 ezer Forintos összegért beszerezhető UBLOX NEO-6M GPS modult fogok használni. A referencia pontokkal végzett mérések alapján a pontosság körülbelül átlagosan 1,7 méter volt. Azonban a szórás még szabad ég alatt is jelentősnek bizonyult, időnként 3-4 méteres eltérést is mutatott, amit alkalmanként 0,5 méteres pontosságú értékek ellensúlyoztak.

A modul hardveres beépítése igen egyszerűen adódott, csupán a kék tábla alján (13. ábra) található 4 csatlakozót (VCC, RX, TX, GND) kellett egy Arduino NANO-ba csatlakoztatni, amely ezután soros porton küldte körülbelül 1 HZ-el a GPS koordinátákat a mérőkör felé. Ezáltal az RTK-s GPS beszerzési költségének kevesebb mint 10%-áért rendelkezésre állt ez a pontatlanabb, de reményeim szerint egy Kalman-szűrő alkalmazása után már hasonló pontossággal rendelkező GPS-es lokalizációs megoldás.



13. ábra A NEO 6M GPS modul antennájával Forrás: Techfun (2024.)

Ugyan majd a következő fejezetben látható lesz, hogy alkalmazásra kerül egy ESP-32-es mikrovezérlő is, azonban mégis ennek külön mikrovezérlőt használtam. Az ESP 9 IMU adatát fogadja és továbbítja, hatalmas sebességgel, és a túlterhelést szerettem volna elkerülni, valamint a térbeli pozíciókülönbségből is adódik, hogy nem lehetne 1 mikrovezérlővel megoldani a problémát, hiszen az IMU-k és az ESP magasan, a robot tetején vannak, míg a

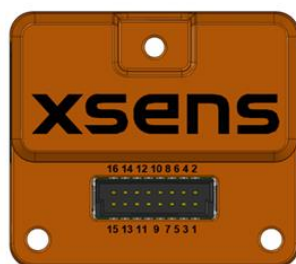
GPS-ek inkább a robot alján helyezkedtek el (ennek oka legfőképp az IMU-ban lévő magnetométer zavarjeleinek kiküszöbölése.)

3.5 Inerciamérő egységek

Az inerciamérő egységek közül ugyan két különböző típusú kerül felhasználásra, mégis ebben a fejezetben azok közösen kerülnek tárgyalásra, alcímek alatt. Erre azért van szükség, mert fontos szempont a két szenzor egytengelyűségének biztosítása, valamint a megfelelő tengelyállások (16. ábra) biztosítása a pontos mérések elvégzéséhez. Ezért a 3.4.1-es fejezet az XSENS MTI-670, majd a 3.4.2-es fejezet az ICM-20948-as szenzorok általános bemutatásáról, konfigurációjáról, valamint a mérés és adatgyűjtés folyamatáról szól. Majd ezek után egy közös fejezetben (3.4.3) kerül tárgyalásra az a 3D nyomtatott tartó, amely a két feljebb megfogalmazott követelményeknek eleget tesz, valamint védelmet is nyújt a szenzoroknak. Ebbe az egységbe kerül majd beépítésre az ICM szenzorok pillanatnyi értékét a Laptopra közvetítő ESP-32-es mikrovezérlő is.

3.5.1 XSENS MTI-670 IMU modul

A következő szenzor, amely felszerelésre került az egy XSENS MTI-670-es 9 tengelyes IMU, amely kompaktsága (befoglaló méretei 28mm x 31,5mm) mellett nagy pontossággal rendelkezik. Adottak tehát a három tengely mentén a gyorsulás, a szögsebesség és a magnetométer adatok. Kis mérete ellenére ára körülbelül 5-600 ezer Forint közé tehető, míg a 3.4.2-es fejezetben alkalmazott szenzorok esetében az ár mindössze néhány ezer Forint. Éppen ezért az alkalmazott rendszer referencia-koordinátáinak mérését ez a szenzor fogja végezni, vagyis a szenzorfúzióban ezek az adatok nem vesznek részt, hanem pontos pozícióként fognak számítani. A 14. ábra mutatja be a szenzor kinézetét, látható rajta a speciális csatlakozó, melyről a következő bekezdésben ejtek szót. (Xsens Technologies B. V. 2020.)



14. ábra Az XSENS MTI-670 felülnézeti ábrája. Forrás: Xsens Technologies B. V. (2020.)

Az IMU-hoz a gyártó készített egy fejlesztői készletet, mellyel könnyen lehetne csatlakoztatni a szenzort számítógéphez, de ez további sok százezer Forintba került volna, így olcsóbb megoldásként a szenzort önállóan használok. Azonban ezekhez is szükség volt kiegészítő egységekre, valamint forrasztásra és szerelésre. A két beszerzendő eszköz közül az

első a CA-MTI-FLAT Ribbon Cable volt, amely gyakorlatilag egy csatlakozó, mely egyik fele a szenzorban található pinekbe kerül, a másik fele pedig szabad kivezetésként, majd a második eszközhöz csatlakozik. A második eszköz pedig egy MIKROE-483 USB-UART kommunikáció konverter. A plusz költség ennek a két eszköznek a megvásárlásával 400 ezer Forint helyett 15 ezer Forintra és egy kis szakmai munkára csökkent, mert ebben az esetben nem olyan egyszerű az összeszerelés, mint ha a Development Kit-et használtam volna. A kábelből négy kivezetést, pontosan a VIN tápfeszültséget, a GND földelést, és az RX, TX kivezetéseket – melyek a soros porton történő kommunikációs protokoll vezetékei – kell a konverterbe forrasztani. A dokumentáció alapján a használandó kivezetések az 1-es, 2-es, 15-ös és 16-os. Egy elég érdekes probléma is itt derült ki, hiszen alapvetően egyféle kábelkiosztást lehet találni az szenzor adatlapján, de hosszas fórum böngészés után kiderült, hogy a CA-MTI-FLAT kábelből kétfélét gyártanak, és igazából nem nagyon lehet megkülönböztetni őket, külsőre pont ugyanolyanok, viszont a 15. ábrával és az 1. táblázattal szeretném bemutatni a problémát, amely végül egy olyan gondot generált, amit csak egy segítőkész Movella-s (az XSENS-et gyártó cég) munkatárs segítségével sikerült elhárítani, több hetes levelezéssel és hibakereséssel.



15. ábra A Két CA-MTI-FLAT kábel kiosztása, baloldalon a Phoenix, jobboldalon a HSF típusúval. Forrás: Movella (2020.)

Pirossal ki is emelték ebben a forrásban, hogy mire kell figyelni. A táblázatban pedig ezt meg is erősítem.

1. táblázat Kivonat a Phoenix és HSF csatlakozók számozásából. Forrás: Movella (2020.)

Vezeték	Azonosító	Phoenix Vezetékszám	HSF Vezetékszám
VIN	a1	1	2
GND	a2	2	1
UART_RX	a8	15	16
UART_TX	b8	16	15

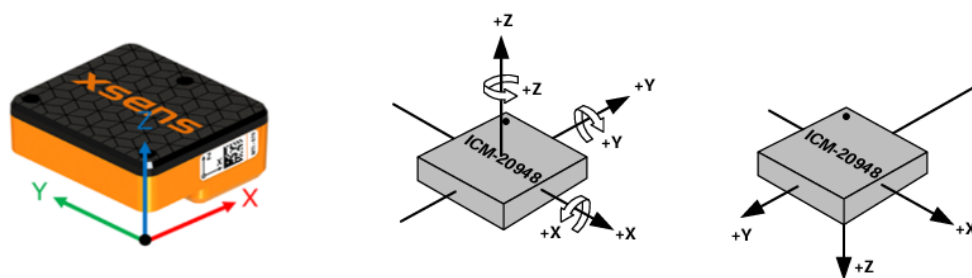
Látható tehát, hogy a két típus között a csatlakozók kialakítása miatt VIN pozitív tápfeszültség, és a GND földelés, valamint ennek megfelelően az UART két vezetéke is fel vannak cserélve. Mivel a rendelés során semmi megkülönböztetésre utaló jel nem volt, sőt arról sem volt információ, hogy ez a kettő lehet különböző is, így a szenzor beüzemelése hosszadalmasabb volt, mivel folyamatosan kontakthibára utaló jelek voltak. Természetesen a védőáramkörök egy ilyen drága szenzorban megvédik a szenzort az esetleges rossz bekötésektől, így a bosszúságon kívül más kár nem keletkezett.

Ezt követően a szenzorhoz tartozó MT Software Suite elnevezésű szoftver és az adatlap alapján már sikerült könnyen konfigurálni az IMU-t, beállítani a megfelelő kimeneteket, melyek alapján majd rekonstruálni lehet a robot állapotait. Fontos lépés volt itt is a mintavételi frekvencia beállítása, melyet az alapértelmezetten beállított 200 Hz-ről a felére csökkentettem az kommunikáció túlterhelésének elkerülése céljából. A következő lépés itt is a megfelelő ROS package letöltése volt, illetve annak beüzemelése. A hozzá tartozó csomag az XSENS MTI Driver package volt. A baud rate-t 921600 baud/s-ra állítottam, valamint beállítottam a rögzítésre kerülő topicokat is. Ennek a forráskódja megtalálható a 2. számú mellékletben.

3.5.2 ICM-20948 IMU tömb

Végül, de nem utolsó sorban az 9 darab alacsony költségű ICM-20948-as IMU-ból álló egység kerül bemutatásra. A használt szenzor adatlapja a „világ legalacsonyabb fogyasztású 9 tengelyes Mikro-Elektro-Mechanikai rendszerű mozgáskövető eszköze”-ként hivatkozik a szenzorra. Az ára egy darab szenzornak 3000 Forint alatt van, tehát összehasonlítva a másik IMU-val (XSENS) jogosan nevezhető alacsony költségűnek. Egy egységbe szerelve három oszlopban és három sorban összesen 9 darab ilyen szenzor került elhelyezésre. Egy pontosabb mérés kialakítása miatt volt szükség több szenzorok használatára, hiszen azért nem csak árban, de pontosságban is elmarad a szenzor a XSENS-eshez képest. Ami egy kisebb problémát okozott, és figyelmet érdemelt az a két IMU szenzor típus közötti egytengelyűség biztosítása, valamint a tengelyeltérések, mely a 16. ábráról leolvasható. (InvenSense 2021.)

Az ábráról látható, hogy megfelelő forgatással biztosítható a közös tengelyhasználat. A másik problémát pedig a következő fejezetben bemutatott 3D nyomtatott tartó fogja megoldani, amely egymás fölé, de az X-Y síkon ugyanabba a pozícióba és orientációba helyezi a két IMU-modult.

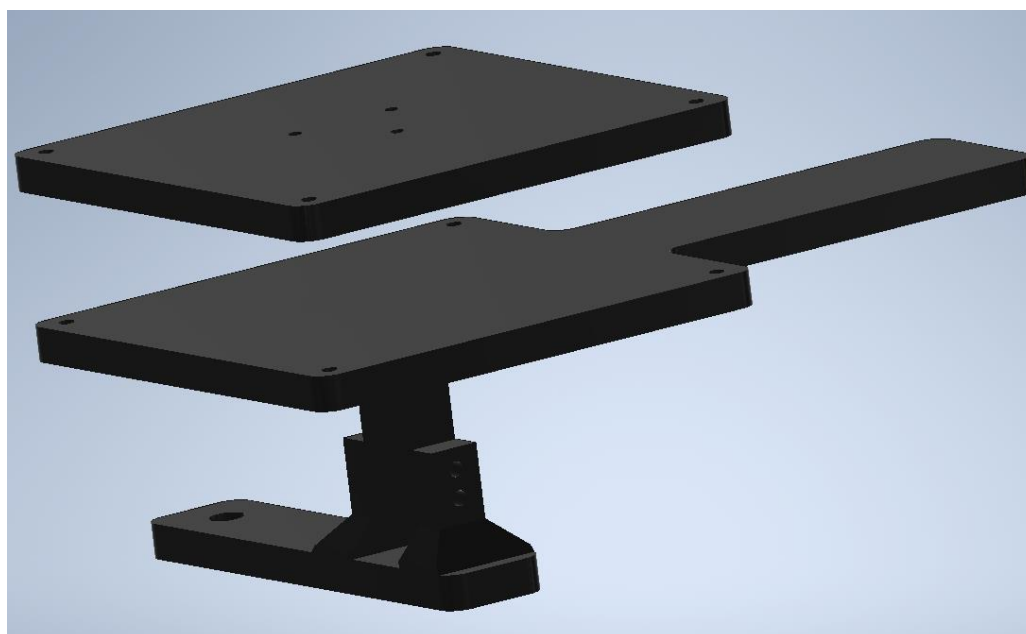


16. ábra Bal oldalon az XSENS tengelyelrendezése minden szenzorra Forrás: Movella (2020.), középen az ICM giroszkóp és gyorsulásmérő elrendezése, jobb oldalon az ICM magnetométerek tengelyelrendezése. Forrás: InvenSense (2021.)

Még mielőtt azonban áttérnénk a tartóra, az IMU tömb adatgyűjtésének folyamata kerül bemutatásra. Az IMU tömb adatait egy ESP-32-es mikrovezérlő fogadja, és továbbítja a laptopra a /imu topicban, 2 millió baud/s baud rate-l. Erre a hatalmas küldési sebességre az összességében 720 Hz-es mintavételezés miatt van szükség. A fogadás után pedig 9 darab vesszővel elválasztott adatfájl (.csv) kerül generálásra. Ennek a forráskódja a 3. számú mellékletben található meg, míg a mérés elindítása a korábban már hivatkozott 2. számú melléklet második blokkjában található meg.

3.5.3 3D nyomtatott tartó

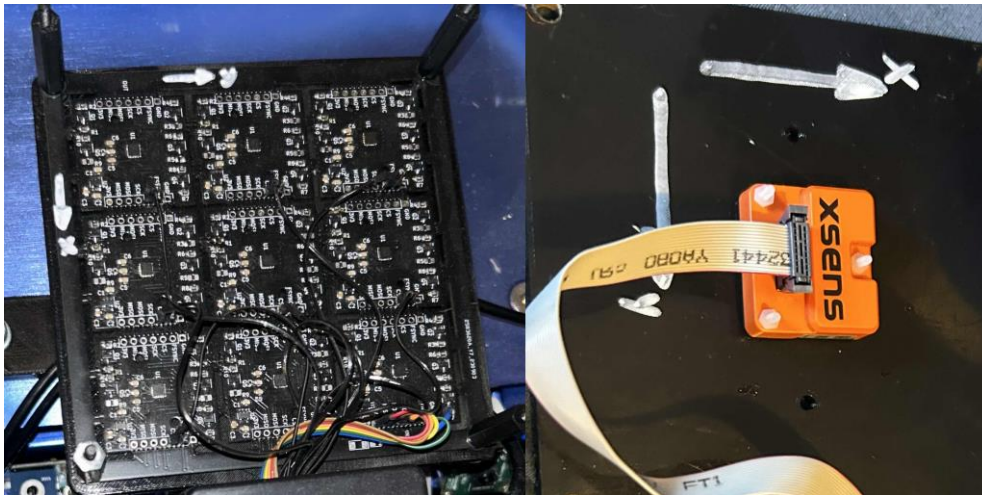
Az összesen 10 darab IMU-nak tehát 3D nyomtatásra került egy tartó, mely biztosítja az egyező tengelyállásokat, valamint kialakításából adódóan egy síkban, egymás felett helyezi el a két egységet. A 17. ábra mutatja be a tartó 3D modelljét.



17. ábra Az IMU tartók 3D modellje

A felső rész közepén található három furaton keresztül fogatom fel M2x40-es műanyag csavarral az XSENS IMU-t míg alulra kerül a tömb. Az alsó résszel csatlakozik a tartó a robothoz, a jobb oldali kinyúló rész pedig az ESP-32-t tartalmazza, mely a tömb méréseit fogadja és továbbítja a Laptop felé.

A valóságban pedig összeszerelés előtt a 18. ábra mutatja be a kész alját és tetejét összevágva, az IMU modulokkal. Balról a tömb, jobbról az XSENS.



18. ábra Az IMU modulok és tartójuk szétszerelve

3.6 A mérés menete

Ebben a fejezetben kerül bemutatásra egy mérési scenárió teljes menete. Majd ábrán keresztül bemutatásra kerül a mérőrendszer különböző talajtípusokon, végül egy általános összefoglaló a mérés során készült fájlokról.

Első lépés mikor megérkeztem a helyszínre az RTK-s GPS bázisállomás koordinátának survey-in módszerrel történő kiátlagolása. A következő lépésben a Magni robot bekapcsolása, a Laptop és az USB portok összekötése következik, majd több perces várakozás, mire a GPS két egysége felveszi egymással a kapcsolatot, és elkezdődik az RT korrekció. Ezt követően futtatásra kerül a `start_sensors.launch` fájl, mely a 2. számú mellékletben megtalálható. Ez a launch fájl elindítja a szenzorok publikálását a megfelelő topicokba, és ezáltal ellenőrzésre kerülhet minden szenzor működése. Ezt követően egy pár soros szkripttel elindítottam a mérést, a szkript tulajdonképpen három részből áll, először a 3. számú mellékletben található `main_imu.py` fájlt indítja és legenerálásra kerül a 9 darab .csv fájl, valamint elindítja a rosbag-elést mind a Laptopon, mind a Magni roboton.

A mérés során célnak tűztem ki, hogy a lehetőségeket tekintve minél több talajtípuson szeretném végezni a különböző mérési scenáriókat. A mérőrendszert alkalmaztam macskakövön, fűvön, aszfalton, szántóföldön és homokon, a 19. ábra ebben a sorrendben

ábrázolja a rendszert. A talajtípusoknak majd a feldolgozás részben lesz egy nagyobb szerepe, ahol klasszifikációs (osztályozós) neurális hálózatban kerülnek alkalmazásra.



19. ábra A mérőrendszer különböző talajtípusokon

A vizsgálódás során összesen 45 mérési scenáriót készítettem el, darabszámra körülbelül ugyanannyit minden típusú talajon. Minden scenárió esetében elkészült 11 darab fájl (9 darab .csv és 2 darab rosbag), melyeket strukturáltan mappába rendezve mentettem el, és a következő fejezetben feldolgozásra kerültek.

4. MÉRÉSI ADATOK FELDOLGOZÁSA ÉS KIÉRTÉKELÉSE

Ebben a fejezetben a mérési adatok feldolgozásának algoritmusai, módszerei kerülnek bemutatásra, valamint a kapott eredmények performanciájának kiértékelése is megvalósul. Először a mérés során készített fájlok kibontását, előfeldolgozását fogom bemutatni, ezt követi egy általános Kalman-szűrő implementációja, majd a szűrő optimalizálását fogom bemutatni részecske-raj optimalizációs (PSO) algoritmus segítségével. Ezután bemutatom az optimalizáció eredményét, majd átlagos zajmátrixokat számolok adott talajtípusokra az optimalizált eredményekkel. Ezután létrehozok egy klasszifikálós neurális hálózatot, ahol az IMU adatokat felhasználva próbálom osztályozni a talajtípusokat, és a neurális hálózat kimenetének segítségével beállítom a zajmátrixokat talajtípusnak megfelelően, mely a későbbiekben akár valós idejű lokalizáció esetében is alkalmazható lesz.

Az adatfeldolgozási rész Matlab R2022b verziójával valósult meg, melyhez egyetemi licencen keresztül jutottam hozzá.

4.1 A mérés során mentett fájlok kibontása és konvertálása

Ebben a fejezetben bemutatásra kerül, hogy hogyan lett a korábban már említett, minden mérés során legenerált 11 fájlból (9 IMU adat, laptop és magni bag fájlok) számomra hasznosítható formátumban, rendezve elmentve a mérési adatok, szenzoronként különválasztva.

A Matlab támogatja a rosbag fájlok kibontását, csupán a select függvény futtatásával kiválaszthatjuk a kívánt topicot, majd a readMessages függvény segítségével az összes erre a topicra érkező üzenetet kiolvashatjuk. Ha megvan az üzenetek kiolvasása, akkor már a Matlab az egyes üzeneteket automatikusan felosztja olyan mezőkké, mint például GPS esetében a Latitude vagy Longitude mezők, valamint, ha az üzenethez tartozott időbélyeg, az is itt megjelenik sec és nsec formátumba a szekundum és nanoszekundum mértékegységekre utalva. Ezután egy egyszerű for ciklussal kimenthetünk mindent, ami számunkra szükséges. A 20. ábrán látható összefoglalva az eddig említett forráskódrészlet egy példán keresztül, a teljes forráskód pedig a 4. számú mellékletben található. A kódrészlet végén a save függvénnyel elmenthetjük a kívánt formában a mérési adatokat.

```

robot_bag = rosbag(robot_bag_path);
cmd_vel_topic = select(robot_bag, 'Topic', '/cmd_vel');
cmd_vel_msgs = readMessages(cmd_vel_topic, 'DataFormat', 'struct');
for i = 1:num_msgs

    time = cmd_vel_topic.MessageList.Time(i);
    sec = floor(time);
    nsec = round((time - sec) * 1e9);

    cmd_vel_data(i).Sec = sec;
    cmd_vel_data(i).Nsec = nsec;

    % Linear Velocity
    cmd_vel_data(i).Linear.X = cmd_vel_msgs{i}.Linear.X;
    cmd_vel_data(i).Linear.Y = cmd_vel_msgs{i}.Linear.Y;
    cmd_vel_data(i).Linear.Z = cmd_vel_msgs{i}.Linear.Z;

    % Angular Velocity
    cmd_vel_data(i).Angular.X = cmd_vel_msgs{i}.Angular.X;
    cmd_vel_data(i).Angular.Y = cmd_vel_msgs{i}.Angular.Y;
    cmd_vel_data(i).Angular.Z = cmd_vel_msgs{i}.Angular.Z;
end
save(fullfile(save_folder, 'cmd_vel_data.mat'), 'cmd_vel_data');

```

20. ábra Részlet a `convert_bag_to_struct.mat` fájlból

Ezt a kódrészletet lefuttattam minden scenárió esetében, a `/cmd_vel`, az `/ublox_gps_node/fix`, a `/gps_data`, az `/xes_imu_mag_data` és az `/xsens_imu_data`, topicok esetében is. Ezt követően a jelenlegi munkához kapcsolódóan minden adatok kimentettünk a bag fájlokból, és létrejött összesen 5 darab `.mat` kiterjesztésű Matlab adatfájl.

Az IMU tömb esetében a korábban említett módon 9 darab `.csv` fájlt generálok, amely előfeldolgozása eltér az előbbieken bemutatottaktól. Ebben az esetben egy egyszerű táblázatbeolvasással (`readtable` függvény) letudhatjuk a feladatot. Mivel itt a fájl felépítését (a korábbi fejezetnek megfelelően) saját kezűleg írtam, így itt már az adott mezők a számomra kívánt struktúrában vannak, egy egyszerű mentéssel már meg is kaptuk a kívánt `.mat` kiterjesztésű Matlab adatfájlt. Ebben egy dupla `for` ciklust alkalmaztam, az első a scenáriókon megy végig, a második pedig a 9 `csv` fájlra.

Ezután az ebben a dolgozatban használt összes mérési adat rendelkezésre áll, megfelelő struktúrában, saját mappába mentve megtalálható, és elindulhat az adatfeldolgozás valódi része.

4.2 Referencia koordináták kiszámítása (Ground Truth)

Még mielőtt a Kalman-szűrőre rátérnék, ebben a részben bemutatom a performancia értékelésre szolgáló referencia koordináták – azaz Ground Truth – koordináták meghatározását.

A legpontosabb mérést számomra az UBLOX RTK-s GPS modulja szolgáltatja, így annak koordinátáit veszem alapul. A munkám során többször előkerül a GPS koordináták hosszúság mértékegységre való váltása, amely bemutatása is ebben a részben történik. Inicializáláskor az első GPS koordinátát veszem alapul, és annak megfelelően veszem fele a konverziós-rátát a két egység között. Tehát az első adatpont szélességi (latitude) és hosszúsági (longitude) koordinátáját használom a `ref_latitude` és `ref_longitude` koordináták

meghatározásához. Az ismert közelítő érték a szélességi fok méterbe történő átváltása esetén, mely konstans értéknek tekinthető: $111320 \left[\frac{m}{^\circ} \right]$

A hosszúsági fok esetében már nem beszélhetünk a közelítő értékről konstansként, mivel a hosszúsági köröknél a távolság nagysága már függ a szélességi kör értékétől is, ezért erre az alábbi képlet használható:

$$111320 \cdot \cos(lat) \left[\frac{m}{^\circ} \right] \quad (1)$$

ahol a lat a szélességi koordinátája az adott pontnak, fok mértékegységben. A konvertálásra a következő kódrészletet (21. ábra) használtam, ahol a gps_x és gps_y változóba mentettem a robot pozícióit méter mértékegységben, alaphelyzetben origóból kiindulva.

```
ref_latitude = gps_latitudes(1);
ref_longitude = gps_longitudes(1);
meters_per_deg_lat = 111320;
meters_per_deg_lon = meters_per_deg_lat * cosd(ref_latitude);
gps_x = (gps_longitudes - ref_longitude) * meters_per_deg_lon;
gps_y = (gps_latitudes - ref_latitude) * meters_per_deg_lat;
```

21. ábra A GPS koordináták konvertálása méter mértékegységbe

Ezt követően, mivel a Kalman-szűrő kimeneteként összességében a cmd_vel topicra érkezett adatsomagok számának megfelelő állapot (x , y , θ) fog rendelkezésre állni, a hiba számításához a Ground Truth koordinátáknak is ilyen hosszú tömböt kell alkotniuk. Ezt lineáris interpoláció segítségével értem el, ahol a végén koordinátákban kaptam meg a Ground Truth GPS értékeket. Ezt a 22. ábra kódrészlete szemlélteti:

```
num_gps_points = length(gps_latitudes);
num_cmd_vel_points = length(cmd_vel_times_sec);

gps_indices = linspace(1, num_gps_points, num_cmd_vel_points);

ground_truth_gps_lat = interp1(1:num_gps_points, gps_latitudes, gps_indices, 'linear');
ground_truth_gps_lon = interp1(1:num_gps_points, gps_longitudes, gps_indices, 'linear');
```

22. ábra Lineáris interpolációval megkapott Ground Truth koordináták

4.3 Kiterjesztett Kalman-szűrő (EKF)

Ebben a fejezetben bemutatásra kerül az első kifejlesztett algoritmus, amely egy kiterjesztett Kalman-szűrő (EKF) algoritmus. Ebben a szűrőkörben felhasználásra kerülnek a mérés során kiadott sebesség vezérlő utasítások $/cmd_vel$ topicból származtatva, lineáris sebesség és szögsebesség formájában, melynek megfelelője a mentett adatszerkezet lineáris sebességének 'X' komponense, valamint szögsebességének 'Z' komponense. Ezek az adatok képezik a mozgásmodell bemenetét. A megfigyelés modell alapját az RTK nélküli GPS koordináták adják, szélességi és hosszúsági koordinátákkal.

Először bemutatásra kerül a mozgásmodell, majd a megfigyelésmodell, ezt követi a szűrőkör általános felépítése. Végül a következő fejezetben az algoritmus optimalizálására is sor kerül.

4.3.1 Mozgásmodell

A szakirodalmak két fontosabb mozgásmodellel foglalkoznak, az egyik a pozíció bemenetű, mely általában enkóder adatokra épülő odometriával dolgozik, valamint a sebesség bemenetű, ahol a lineáris és szögsebesség értékek szolgálnak input paraméterként. Ebben a rendszerben én az utóbbit alkalmaztam, ahol a bemenetek a Joystickkel kiadott vezérlő sebességi parancsok voltak.

Tehát volt két bemenetünk a lineáris sebesség 'X' irányú komponenséből származó lineáris sebesség komponens, valamint a szögsebesség 'Z' irányú komponenséből származó szögsebesség komponens, jelük rendre v és ω . Ezek mellett bemenetként szolgált a robot előző állapota, $x_{n-1}(x, y, \theta)$ jellel és ennek kovarianciája Σ_{n-1} , 3x3-as diagonális mátrixként, a mozgás időtartama dt jellel, valamint a mozgásmodell bizonytalanságát gaussi fehér zajként figyelembe vevő 3x3-as diagonális mátrix Q jellel.

A függvény (modell) kimenete pedig a priori állapotbecslés és a hozzá tartozó priori kovarianciabecslés mátrixai lesznek, rendre 3x1-es és diagonális 3x3-as mátrixként.

A differenciális hajtású mobil robotok esetében a sebesség bemenetű mozgásmodell a szögsebesség függvényében két részre bontható, így ez a két eset külön kerül bemutatásra.

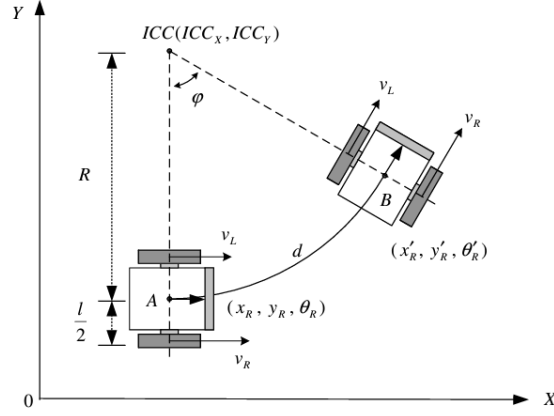
Abban az esetben, ha az adott bemeneti időpillanatban a szögsebesség értéke nem nullára ($\omega \neq 0$) adódik, akkor a mozgás köríven teljesül. Először tehát meghatározandó ennek a körívnek az pillanatnyi görbületi középpontja – szakirodalmakban ICC, azaz Instantaneous Center of Curvature-ként hivatkoznak rá – x és y koordinátával (ICC_x ; ICC_y), valamint a kör sugara (r). Ennek a három adatnak a kiszámolására az alábbi képletek használhatók:

$$r = \frac{v}{\omega} [m] \quad (2)$$

$$ICC_x = x_{n-1} - r \cdot \sin(\theta_{n-1}) [m] \quad (3)$$

$$ICC_y = y_{n-1} + r \cdot \cos(\theta_{n-1}) [m] \quad (4)$$

Azt, hogy ez a három adat pontosan mit is jelent, a 23. ábra mutatja be. Az r sugár itt R jelöléssel szerepel.



23. ábra A robot mozgását szemléltető ábra Forrás: Han, Choi és Lee (2008.)

Ebből a három adatból már számíthatóvá válik a priori, vagyis prediktált állapotbecslés:

$$\hat{x}_n = \begin{bmatrix} ICC_x + r \cdot \sin(\theta_{n-1} + \omega \cdot dt) \\ ICC_y - r \cdot \cos(\theta_{n-1} + \omega \cdot dt) \\ \theta_{n-1} + \omega \cdot dt \end{bmatrix} \quad (5)$$

Azonban megfigyelhető, hogy ez egy nemlineáris modell, miközben az egyszerű Kalman-szűrő esetében csak és kizárólag lineáris esetben működik. Ennek következtében linearizálásra van szükség, melyet a jól ismert Jacobi-mátrix számításával valósult meg. A Jacobi-mátrixhoz a fent becsült állapot mátrixát parciálisan deriválni szükséges, melyből végeredményként az F, állapotátmeneti mátrixot kapjuk:

$$F = \begin{bmatrix} 1 & 0 & r \cdot (\cos(\theta_{n-1} + \omega \cdot dt) - \cos(\theta_{n-1})) \\ 0 & 1 & r \cdot (\sin(\theta_{n-1} + \omega \cdot dt) - \sin(\theta_{n-1})) \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

A másik esetben, mikor a szögsebesség értéke nullára ($\omega = 0$) adódik, akkor a robot egyenesen halad tovább, megtartva korábbi orientációját. Ebben az esetben rögtön adódik az állapotbecslés és az állapotátmeneti mátrix:

$$\hat{x}_n = \begin{bmatrix} x_{n-1} + v \cdot \cos(\theta_{n-1}) \cdot dt \\ y_{n-1} + v \cdot \sin(\theta_{n-1}) \cdot dt \\ \theta_{n-1} \end{bmatrix} \quad (7)$$

$$F = \begin{bmatrix} 1 & 0 & -v \cdot dt \cdot \sin(\theta) \\ 0 & 1 & v \cdot dt \cdot \cos(\theta) \\ 0 & 0 & 1 \end{bmatrix} \quad (8)$$

Végül a priori kovariancia becslés meghatározása következik. Ennek egyenlete a következő:

$$\widehat{\Sigma}_n = F \cdot \Sigma_{n-1} \cdot F^T + Q \quad (9)$$

A mozgásmodell implementációja pedig a 5. számú mellékletben található. Programozási szempontból annyi megjegyzést tennék a forráskódhoz, hogy lebegőpontos számok a kerekítésekből adódó hibák, illetve zajok miatt nem a nulla értéket vettem az egyenesmozgás feltételének. A modell egyenes mozgásnak veszi azt, amikor a szögsebesség $0.00001 - \text{azaz egy százezred } \frac{\text{rad}}{\text{s}} - \text{érték alatti, amely a szokványos programozástechnikai eljárás.}$

4.3.2 Megfigyelésmodell

A megfigyelésmodell kimenetei a posteriori állapotbecslés és kovariancia mátrix, míg bemenetei a priori állapotbecslés és kovariancia mátrix, a mérési adatvektor 'z' jellel, amelyek a GPS koordináták átkonvertálva síkbeli ponttá (x és y értékek, méter mértékegységben megadva), valamint a megfigyelésmodell bizonytalanságát figyelembe vevő R, 2x2-es diagonális zajmátrix.

Definiálni szükséges a megfigyelési mátrixot, mely a H jelet kapta, és megadja, hogy az állapotvektor mely paraméterei kapcsolódnak közvetlenül a méréshez. Jelen esetben az állapotvektor három paramétert tartalmaz, melyek a robot síkbeli koordinátái, valamint orientációja rendre 'x', 'y' és 'θ' jellel. A megfigyelés GPS koordinátákon alapszik, melyet közvetlenül csak az első két paraméterre vonatkoztathatunk, tehát orientáció megfigyelésünk közvetlenül nincs (a mozgás során azonban közvetetten tudunk rá következtetni), ezért a megfigyelési mátrix:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (10)$$

A következő lépésben az innovációs kovariancia mátrix (S) kerül meghatározásra, mely a priori kovarianciabecslés és a mérési zajmátrix, valamint a megfigyelési mátrix segítségével számítható, és megadja, hogy a mérés és a becslés közötti eltérésben mennyire lehet megbízni. Az S mátrix tehát egy diagonális 2x2-es mátrix, melynek képlete:

$$S = H \cdot \widehat{\Sigma}_n \cdot H^T + R \quad (11)$$

Ezt követően a Kalman-erősítés (K mátrix) meghatározása következik, mely azt adja meg, hogy a mérési információ milyen mértékben használandó az állapot frissítésére. A K mátrix ebben az esetben egy 3x2-es mátrix, mérete egyébként az állapotvektor és a megfigyelésmodell bemeneti 'z' vektorától függ. Ha K nagy a frissítést markánsabban befolyásolja a mérés. A K mátrix az alábbi összefüggésből adódik:

$$K = \widehat{\Sigma}_n \cdot H^T \cdot S^{-1} \quad (12)$$

Még a posteriori becslések előtt meg kell határozunk az innováció mértékét is, mely a ténylegesen mért értékek (z) és az előre jelzett mérés különbségeként adódik:

$$y = z - H \cdot \widehat{\Sigma}_n \quad (13)$$

Ezt követően a kimeneti posteriori becslések már könnyen számíthatók:

$$x_n = \hat{x}_n + K \cdot y \quad (14)$$

$$\Sigma_n = (I - K \cdot H) \cdot \widehat{\Sigma}_n \quad (15)$$

A fenti képletből eddig egyedül az I egységmátrixról nem esett szó, amely egy 3x3-as méretű, a főátlóban 1-es, a többi helyen 0 értéket felvevő mátrix, amely azt hivatott biztosítani, hogy a frissítési művelet elvégzése során az eredmény megfelelő méretű legyen.

A megfigyelés modell implementációja megtalálható a 6. számú mellékletben.

4.3.3 Inicializáció, optimalizálás előtti eredmények

Ebben a fejezetben bemutatásra kerül az általános beállításokkal implementált Kalman-szűrő algoritmus performanciája a zajmátrixok optimalizálása előtt egy általános beállítással. Már ennél a megoldásnál is látható lesz egy szignifikáns eltérés a csupán mozgásmodellt (odometriát) alkalmazó és a Kalman-szűrt eredmény között. Majd az optimalizálás során talán kevésbé látványos, de számokkal (a hibaérték csökkenésével) az optimalizálás szükségessége is igazolva lesz. A Ground Truth koordináták (amihez viszonyítjuk a performanciát) pedig a korábban említett módon kerültek meghatározásra.

Az inicializáláskor a kezdeti pont koordinátáit és orientációját nulla értékre állítottam be, valamint beállítottam egy feltételezett alapkovarianciát, melynek értékei az főátló első két elemében m^2 a harmadik elemében rad^2 mértékegységben a következő mátrixot alkották:

$$\Sigma_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0,1 \end{bmatrix} \quad (16)$$

A Σ_0 mátrix tehát azt mutatja meg, hogy mennyire vagyunk biztosak a kezdeti állapotban. Egy általános megközelítés szerint a fenti értékeket használtam.

Továbbá szükséges inicializálni hasonló módon a mozgásmodell bizonytalanságát figyelembe vevő Q, valamint a megfigyelésmodell bizonytalanságát figyelembe vevő R mátrixot is, melyek értékei a következőképpen alakulnak:

$$Q = \begin{bmatrix} 0,1 & 0 & 0 \\ 0 & 0,1 & 0 \\ 0 & 0 & 0,01 \end{bmatrix} \quad (17)$$

$$R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \quad (18)$$

Ebben az esetben minden nemnulla érték m^2 mértékegységben van, a Q mátrix főátlójának harmadik elemét kivéve, ami az orientációra vonatkozik, tehát rad^2 mértékegységben van. Ezek az értékek azt feltételezik, hogy a mozgásmodell pontossága

nagyobb, míg a megfigyelésmo­dell, azaz a GPS által mért értékek pontossága kisebb, bizonytalansága nagyobb értéket vesz fel. Ez a beállítás jól működhet olyan helyen, ahol a kerék csúszására kevés esély található, tehát például aszfalton, de homokos terepen még eredmények nélkül is arra számíthatunk, hogy a GPS értékek pontosabban reprezentálják a mozgást.

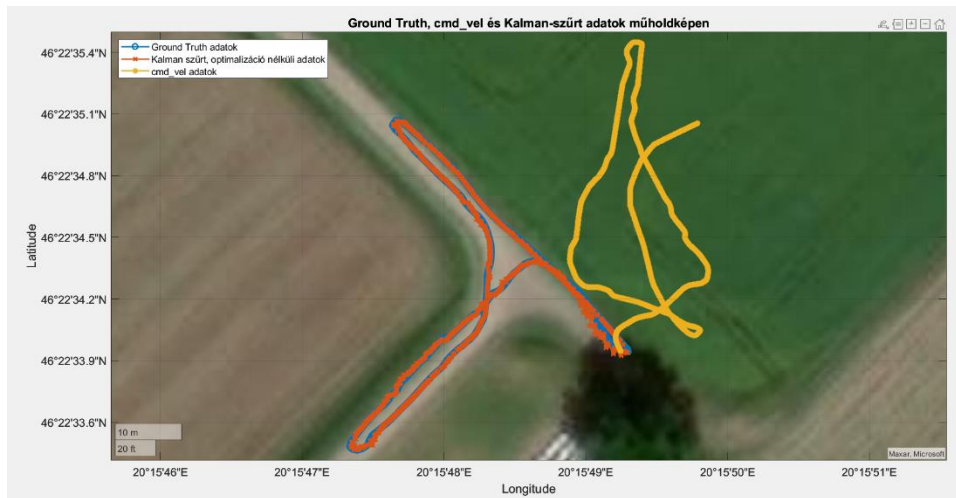
Ezt követően a Kalman-szűrő algoritmusának megfelelően futtatásra került a program, melynek forráskódja a 7. számú mellékletben található, a lényegi részek pedig bemutatásra kerülnek. Az első lépés az idők normalizálása volt, mivel a /cmd_vel topic a Magni roboton, a GPS koordináták pedig a laptopon kerültek rögzítésre, ezért a rendszeridejük különböző időponttal kezdődik, ezért azokat illeszteni kellett egymáshoz. Ezt követően a mozgások időtartamát kellett kivonni egymásból, vagyis a /cmd_vel topic két beérkezett üzenete közt eltelt időt kellett kinyerni, hiszen ez adja majd a 'dt' paraméter értékét. Ezt követően a mozgásmo­dell futtatásával megkaptuk a priori koordinátákat. Ezután ellenőrzésre került, hogy a jelenlegi időpillanat egy kis környezetében található-e GPS megfigyelés, ha nem akkor az iteráció befejeződött és nem került frissítésre a pozíció, mert nem volt megfigyelésünk. Abban az esetben, ha volt megfigyelés, akkor a megfigyelésmo­dell is futtatásra kerül, és kinyerjük a posteriori állapotbecslést. Ha lefutott minden iteráció, akkor a kapott koordinátákat átkonvertáljuk GPS koordinátákká, a műholdképen történő ábrázolás érdekében.

Az összehasonlítás­hoz a /cmd_vel topic üzeneteire külön is futtatásra kerül a mozgásmo­dell, ezután átkonvertáljuk őket szintén a GPS koordináták tartományába. Az így megkapott, csak és kizárólag odometriára támaszkodó görbe inicializálásra került a Ground Truth-nak használt pontosabb GPS modul alapján a következő módon: A kezdeti pont 'x' és 'y' koordinátája megegyezik a pontos GPS modul első méréséből adódó GPS koordinátákkal. Valamint a mérés során feltételeztem, hogy az első néhány másodpercben egyenes mozgás történt, vagyis a kezdeti orientáció nem változott, így az első 10 üzenetből vett mozgás orientációját meghatározva a kezdeti orientációt kaptam.

A következő ábra (24.) mutatja az eredményeket, egy szcenárióból aszfalton, kék színnel a Ground Truth koordinátákkal, sárga színnel az odometria és piros színnel a Kalman-szűrt koordinátákkal meghatározott trajektóriák láthatók. A képek nagysága miatt a többi talajtípusról 1-1 ábrát a 8. számú mellékletbe raktam, de sajnos nincs lehetőség az összes, 46 mérési szcenárióról képet hozni.

Ezen az ábrán látható, hogy már aszfalton is viszonylag könnyen kettéválik az odometria a valós és Kalman-szűrt eredményektől. Viszont a Kalman-szűrő és a Ground Truth koordináták az ábrán szinte teljesen egybe esnek, de ez a mérték miatt csalóka lehet, igenis van

hiba, ha majd az odometriát leveszem az ábráról ez jobban fog látszódni. Pontossági adatokkal is később, az optimalizálásnál fogom bemutatni.



24. ábra A Kalman-szűrt, a Ground Truth és az odometriai adatok műholdképen.

A 24. ábra és a 8. számú mellékletek alapján látható, hogy az aszfaltos úton és a betonos foci pályán az odometria hasonló jellegű, eltérések leginkább a kanyarokban vannak, az egyenes szakaszok tartják a jellegüket. Ennek oka, hogy ebben az esetben a kerékcúsúszások nem jelentkeznek olyan mértékben a talaj keménysége miatt. Fűben, szántóföldön és homokon a kerék sokszor egyhelyben forgott sokat a megcsúszások miatt, így az odometria jellegben sem hasonlít a valós mozgásra, míg macskakő folyamatosan dobálta a robotot, melynek következtében sokszor ellenkormányozni kellett, mely látszik is az odometrián.

4.4 PSO-alapú optimalizált Kalman-szűrő beállítás

Ebben a fejezetben először bemutatásra kerül az optimalizálási algoritmus (részcseke-raj algoritmus), majd a különböző talajtípusokra vett ideális zajmátrix beállítások kerülnek bemutatásra indoklással. Ezt követően táblázatosan bemutatásra fog kerülni a Kalman-szűrők performanciája.

A részecske-raj algoritmussal megvalósított optimalizáció (Particle Swarm Optimization, röviden PSO) a szakirodalmi áttekintés során nem került bemutatásra, mivel nem saját implementációt készítettem belőle, viszont röviden itt bemutatásra kerül.

4.4.1 A PSO algoritmus általánosan

A PSO egy optimalizálási algoritmus, amely a részecskék rajának viselkedését utánozza. Minden részecske egy lehetséges megoldást képvisel, és a térben mozog az optimális megoldás keresése során. Az algoritmus során minden iterációban frissül az összes részecske pozíciója és sebessége három paraméter figyelembevételével. Az első az adott részecske előző iterációiból vett mozgásaira számított inercia értéken alapszik. A második a részecske saját

legjobb korábbi pozíciója alapján kerül meghatározásra, míg a harmadik a teljes raj legjobb pozíciója alapján adódik. Ezen három paraméter eredőjéből képezi az algoritmus a következő iterációban a részecske sebességét. A sokszor használt legjobb kifejezés az algoritmus esetében egy célfüggvény segítségével adódik, mely valamilyen összehasonlításon alapszik az elvárt és kapott kimenet között, például valamilyen hibafüggvény használatos erre a célra.

A dolgozatban alkalmazott rendszerben az algoritmus a Kalman-szűrő zajmátrixainak optimalizálására került felhasználásra. A célfüggvény pedig az átlagos négyzetes hiba négyzetgyökének – talán az angol terminológia ismertebb, Root Mean Square Error, azaz RMSE – kiszámításával került meghatározásra. Az RMSE meghatározása jelen esetben az alábbi képlet segítségével történt:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N [(x_{kal,i} - x_{gt,i})^2 + (y_{kal,i} - y_{gt,i})^2]} \quad (19)$$

ahol N az adatpontok száma, $x_{kal,i}$, $y_{kal,i}$ az i . iterációban vett Kalman-szűrt pozícióértékek méter mértékegységben, a részecskére jellemző zajmátrixokkal $x_{gt,i}$, $y_{gt,i}$ az i . iterációban vett Ground Truth pozícióértékek méter mértékegységben.

4.4.2 A futtatott PSO algoritmus

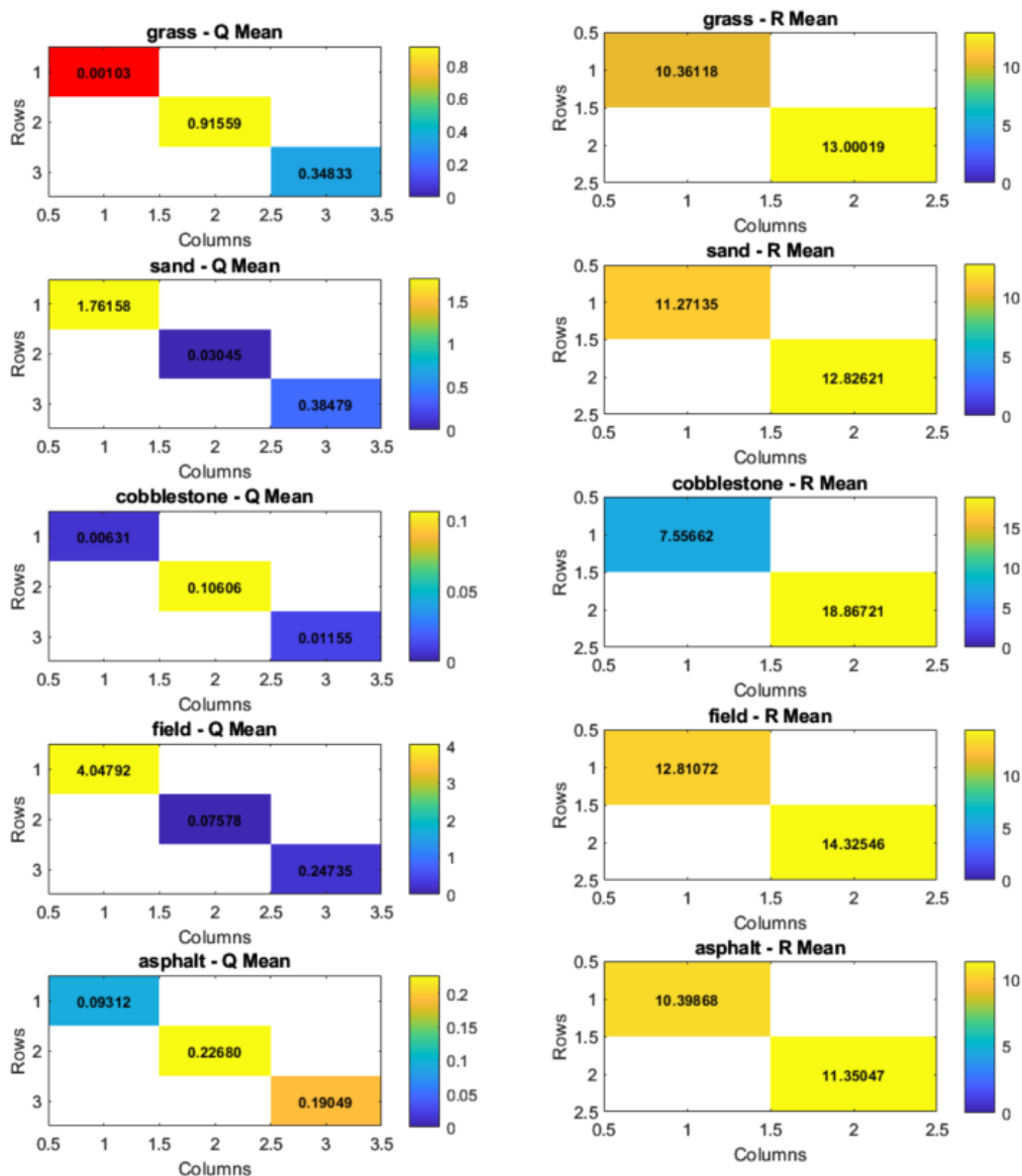
Az algoritmust 50-es részecskeszámmal, 100 iterációban futtatam, egy R és Q mátrix tartományon. Futás után megnéztem, hogy mekkora mértékben javult az RMSE érték, valamint az átlagos abszolút hiba értéke (Mean Absolute Error- MAE). A MAE kiszámítási képlete:

$$MAE = \frac{1}{N} \sum_{i=1}^N \sqrt{(x_{kal,i} - x_{gt,i})^2 + (y_{kal,i} - y_{gt,i})^2} \quad (20)$$

ahol mivel kétdimenziós pontra Euklideszi távolságot számolunk, nem szükséges kirakni az abszolútérték jelet, hiszen a távolság, amely a gyökfüggvény kimeneteként adódik biztosan pozitív lesz.

A célfüggvényben futtatásra kerül a Kalman-szűrő a megadott R és Q mátrixokkal és performanciája meghatározásra kerül. Az összes iteráció végén elmentem a legjobb R és Q mátrixpárost az adott mérési scenárióra. A mátrix optimalizáló Matlab fájl, valamint a célfüggvény programkódja megtalálható a 9. számú mellékletben.

Az optimális zajmátrixokat a talajtípusok alapján átlagoltam, az eredményeket egy szemléletes Matlabbal készített mátrixdiagramon ábrázoltam (25. ábra).



25. ábra Az optimalizált Q értékek talajtípusonként

A kép bal oldali oszlopán a Q mátrix, a jobb oldalon az R mátrix átlagos értékei láthatók. Mindkettő mátrix diagonális, így az ábrán csak a főátló értékeit tüntettem fel, a többi – nulla – érték fehér háttérrel szerepel. Soronként az egyes talajtípusokat láthatjuk, rendre fű, homok, macskakő, szántóföld és aszfalt kategóriákban. Az eredményekből az alábbi következtetéseket vonhatjuk le.

Az R mátrixokat tekintve az összes mérési scenáriót tekintve nagyjából hasonló lett az optimális eredmény, amely logikus, a GPS által mért koordinátákat nem befolyásolja a talaj típusa. Egyedül a macskakövön kaptam egy kicsivel eltérő értéket, az első érték alacsonyabb, a második érték magasabb az átlagosnál, melynek okát én két tényezőben tudom meghatározni. Egyik a kövön való intenzív rázkódás mértéke, mely képes zavarokat okozni. Másik indok

pedig lehet, hogy a mérés a városban történt, és a környező házak vagy fák jobban zavarhatták a GPS-t, mint a többi mérés esetében, ahol az alföldi terepviszonyokat és lehetőségeimet kihasználva mezőgazdasági területeken mértem (a homokot kivéve, amelynél egy szintén nyitott játszótéren voltam). Viszont látható, hogy az olcsó GPS összességében pontatlanabb, mint amire először gondoltam az általános beállítások esetében.

Füves talaj esetében mindhárom Q érték 1 alatt van, alacsonynak számít az első és utolsó érték, míg a középső magasabb. Ennek oka az lehet, hogy mikor a robot a nagyobb fűben elakadt, akkor egyenes mozgással pörgött a kereke. 2D mozgásra ezt levetítve, az egyik tengely mentén állandó értéket kell kapnunk, hiszen abban a síkban nem történik beavatkozás, hiszen egyenes a mozgás, míg a másik irányban gyorsan elpörög az érték, miközben a robot egyhelyben áll, így kialakul erre a tengelyre (jelen esetben 'y') egy közepes mértékű hiba. A homok esetében a Q mátrix első értéke az elég nagy, amely szerintem elvárt eredménynek gondolható, hiszen itt jelentkezett a legnagyobb mértékben a drift jelensége. A szántóföldön tapasztalt jelenségek hasonlítanak a homokos és füves eredményekre, nagyjából hasonló jelleget mutat, amely szintén borítékolható volt, hiszen vetés után egy puha agyagos jellegű talajon mértem. Egy másik hasonló csoportot alkot a macskakő és aszfalt párosa, amely a többitől igen különböző értékeket kapott, viszont egymásra hasonlítanak. Ennek oka, hogy kerékcsúszás ebben a két csoportban egyáltalán nem jelentkezett egyenes mozgás esetében, így kisebb zajt feltételező Q mátrix kerül beállításra. A futtatás során az azonos talajokon átlagolásra került az RMSE és MAE javulás, százalékos értékben, valamint feltüntetésre kerül az átlagos értéke méter mértékegységben a 2. táblázatba.

2. táblázat az optimalizálás eredménye

	Eredeti RMSE [m]	Optimális RMSE [m]	Átlagos RMSE javulás [%]	Eredeti MAE [m]	Optimális MAE [m]	Átlagos MAE javulás [%]
Fű	2,753	2,442	12,17	2,431	2,157	12,03
Homok	2,087	1,797	14,67	1,818	1,565	14,65
Macskakő	2,769	2,514	9,40	2,395	2,182	8,92
Szántóföld	3,975	3,135	18,47	3,411	2,729	17,69
Aszfalt	3,554	2,853	16,86	3,029	2,460	16,44

A táblázatot elemezve látható, hogy a legtöbb helyen két számjegyű javulást kaptam minden értékre. Mivel az alapbeállítás sem vakon történt, hanem egy jó közelítéssel, a rendszer

jellegét figyelembe véve. Ennek ellenére sikerült az eredményeket tovább javítani. A legkevesebbet a macskaköves mérés javult, amely betudható annak, hogy ott volt a legpontatlanabb a GPS.

Az optimalizált és az optimalizálás előtti Kalman-szűrős trajektóriákat összevetem Ground Truth adatokkal, ezáltal létrejött a 26. ábra az összes scenárióra.



26. ábra Az optimalizált és a nem optimalizált Kalman-szűrő trajektóriája.

Az ábrán látható módon csökkentek a kilengések és jobb közelítő eredményt kaptam, mintha nem futtattam volna le az optimalizációt. Az átlagos RMSE értéke viszonylag magasnak tűnik, viszont azt figyelembe kell venni, hogy a mérési rendszer erre felhasznált része 10 ezer Forint alatti költséggel rendelkezik, és – ahogy az majd egy későbbi fejezetben látható lesz – megközelíti, akár meg is előz néhány kültéri lokalizációs eszközt. Beltéren természetesen akár egy nagyságrenddel jobb eredményeket is lehetett volna kapni. Valamint jócskán van még fejlődési lehetőség, egy IMU alapú mozgásmodell, és egy RTK-s GPS megfigyelésmodellre való alkalmazásával (amelyet az összeépített mérőkör tartalmaz, tehát akár ugyanezen scenáriókra is lehetne alkalmazni) sokat lehet javítani a pontosságon. Azonban ebben a feladatban az alacsony költségen, azaz low-cost rendszeren volt a hangsúly.

4.5 Talajtípus klasszifikáció megvalósítása neurális hálózattal

A következő lépés egy klasszifikációs neurális hálózat tanítása és alkalmazása, amely megfelelő biztonsággal képes megmondani, hogy egy adatcsoport milyen talajtípusról származik. Külön elméleti összefoglaló ehhez a témakörhöz sem készült mivel itt sem saját neurális hálózati implementációt valósítottam meg, hanem a Matlab-ba beépített függvényeket használtam, természetesen saját beállításokkal.

Ezután vizsgáltam különböző konfigurációkat, például 6 nyers IMU adatból származó bemenet, 8 bemenet, az előbbi kiegészítve a /cmd_vel értékekkel vagy előfeldolgozás után valamilyen jellemzők kinyerése. A kimenetet is változtattam, 5 osztály és 3 osztály között, mígnem sikerült megkapni a legjobb konfigurációt.

4.5.1 A klasszifikációs neurális hálózatról általánosan

A klasszifikációs neurális hálózat tulajdonképpen egy osztályozási feladatra kifejlesztett gépi tanulási modell, mely a felügyelt tanítási módszert használja. Bemenetei az én esetemben sorvektorok, amely az első részben a nyers IMU adatokból, majd a különböző jellemzőkből áll. A kimeneti réteg pedig annyi neuronból áll, ahány kimeneti osztály meg van adva (5, illetve 3). A bemeneti adatok a címkével látom el, és elvárásom, hogy a címke alapján megfelelő osztályba kerüljenek. A rejtett rétegekben történik az információ feldolgozása, mély neurális hálózatoknál akár több száz rejtett réteg is alkalmazott, az én esetemben ez egy, kettő vagy három.

Az alkalmazott konfigurációban a hiperparaméterek megválasztása is fontos szerepet töltött be, ilyen például a neuron szám, az iterációk, vagy az epochok (mely megadja, hogy tanulás közben hányszor menjen végig a hálózat a teljes adathalmazon) száma.

Végül értékelésre kerül a rendszer pontossága, mely annak megfelelően kerül meghatározásra, hogy a beadott bemeneti paraméterek hány százalékát sikerült helyesen osztályba rendelni.

4.5.2 Nyers IMU adatok alkalmazása

Első megközelítés szerint a bemeneti rétegben a 3 tengely menti gyorsulás és a giroszkóp adatokat használtam. Az eredmény nem volt túl jó, körülbelül maximálisan 50%-os pontosságot sikerült elérni. Ráadásul mivel ebben az esetben körülbelül 336 ezer bemeneti 6-os volt, a tanítás is tovább tartott mint 2 nap, hiszen próbálkoztam az 50 és 150 közötti neuronszámmal, 5 neuronos lépésközzel, 1-3 rejtett réteggel, futásonként 10 iterációval.

A következő próbálkozás az volt, hogy az IMU adatok mellett odometria adatokat is szolgáltatam a neurális hálózatnak. A korábbi fejezetekből megismert /cmd_vel topic lineáris sebességének 'x' irányú és szögsebességének 'z' irányú komponensével 8 bemeneti paraméterre egészült ki a korábbi 6 IMU adat. Ekkor is csupán 1-2%-os javulást sikerült elérnem.

Ezt követően elemezve az előző két megközelítés konfúziós mátrixát, arra jutottam, hogy a hálózat 80-90% pontossággal képes a homok és az aszfalt meghatározására, viszont a másik három típust folyamatosan keveri, alig pár százalékos eredményt sikerült elérni. Ezért megpróbáltam a tanítást az első megközelítés szerint, tehát 6 bemenettel és három kimeneti

osztállyal, aszfalt, homok és egyéb talajtípussal. Ekkor a maximális pontosságom 1-2 nap futás után 61,73% volt. Az eredményeket az alábbi táblázatban (3. táblázat) foglaltam össze.

3. táblázat A nyers adatokkal dolgozó neurális hálózat eredményei.

Neurális Hálózatok	Neuronok száma [db]	Epochok száma [db]	Iterációk száma [db]	Teljes tanulási idő [h]	Maximális pontosság [%]
6 bemenet, 5 osztály	50-150 10-es lépésköz	500	10	48	49,65%
8 bemenet, 5 osztály	75-150 5-ös lépésköz	1000	5	30	53,39%
6 bemenet, 3 osztály	70-150 10-es lépésköz	1000	10	20	61,73%

A fenti táblázat alapján látható a folyamatos fejlődés, viszont folyamatosan egyszerűsítettem is a rendszert, valamint a nagyméretű adatbázis miatt hatalmas komplexitás és futási idő jellemzi a rendszert, az általam kitűzött elvárt pontosság (minimum 90%) pedig nagyon messze van. Ezért egy másik megközelítést választottam. A teljes futási idő az egész intervallumon történő összes neurális hálózat tanítására vonatkozik. Az itt legpontosabb neurális hálózat tanulási ideje 2 óra körül volt, viszont mivel minden rendszerre mások az ideális beállítások, ezért a fontos paramétert nem csupán erre az egy hálózatra, hanem a teljes tanulási időszakra vonatkoztattam.

4.5.3 A legpontosabb megközelítés

Ebben a fejezetben bemutatom a legjobb eredményt hozó klasszifikációs algoritmust. Az ötlet a témavezetőim hathatós közreműködésével került a látókörömbe, és épít Sarcevic és mtsai. (2023.), Electronics folyóiratban közzétett munkájára.

Ebben a tanulmányban többek között IMU szenzorokkal valósítottak meg klasszifikációt, kültéri talajtípusokra mobil robot segítségével. Az általam eddig alkalmazott technológiával ellentétben, az IMU adatok előfeldolgozásra kerültek, és különböző jellemzők kerültek belőle kinyerésre. Valamint nem csupán egy adatsor került egyszerre a bemenetre, hanem ablakozást és eltolást is használ, amely pedig jóval csökkenti a bemeneti adatbázis méretét, így a tanítási idő is inkább másodpercekben, vagy esetleg percekben mérhető, a korábbi több napos helyett. (Sarcevic, és mtsai. 2023.)

Tehát az első változás az ablakozás és eltolás művelete (windowing, shifting). Adott az ablak mérete, amely legyen kezdetben 100, ekkor a tanítás során az adatok 100-asával kerülnek csoportosításra és így csoportokban kerülnének be a neurális hálózatok bemenetére. De alkalmazom az eltolás műveletét is, mely kezdetben 50-es értéket kapott. Így minden csoport kialakítása úgy adódik, hogy első csoportnál az első 100 elemet használom, majd a második csoport úgy alakul ki, hogy kiesik az eltolási értéknek megfelelő számú adat, és helyére a következő adatok kerülnek be. Példa szerint: 1. csoport adatai az első 100 adat, a 2. csoport adatai az 50. adattól a 150. adatig tartó intervallum elemei, és ennek az elvnek megfelelően megyünk tovább.

A másik változás pedig, hogy az így kapott (kezdetben 100 adat hosszú) adatsorból különböző jellemzőket számítok ki. Első esetben az átlagos abszolút értéket (Mean Absolute Value – MAV), négyzetes középérték gyökét (Root Mean Square -RMS), a maximális értéket (MAX), a minimális értéket (MIN) és maximális és minimális érték közötti különbséget (Peak-to-Peak -PTP) értékeket használtam bemenetként a rendszer tesztelésére. A képleteik a következők:

$$MAV = \frac{1}{N} \sum_{i=1}^N |x_i| \quad (21)$$

$$RMS = \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2} \quad (22)$$

$$PTP = MAX - MIN \quad (23)$$

Ahol N az ablak mérete, x pedig az IMU adat egyik értéke, ezek a képletek mind a 6 bemenetre számolandók.

Ekkor tehát száz bemeneti adat helyett használtam összesen öt, könnyen megkapható jellemzőt (mind a hat bemenetre). Az eredmény ebben az esetben is már jóval jobb, 85,47%-os pontosságú.

Miután továbbfejlesztettem a szakirodalomban található mind a 9 jellemzőt felhasználtam már sikerült elérnem a 90% feletti eredményt, annak ellenére, hogy a magnetométerek adatait nem használtam fel. A kimaradt képletek és részletes levezetésük megtalálható a felhasznált folyóiratcikkben. (Sarcevic, és mtsai. 2023.)

Az ablak méretét alap esetben 100 adatot fedett le, míg a mintavételezési frekvenciám 80 Hz volt, azonban az eltolási értéket 50 adat hosszúnak határoztam meg, így minden

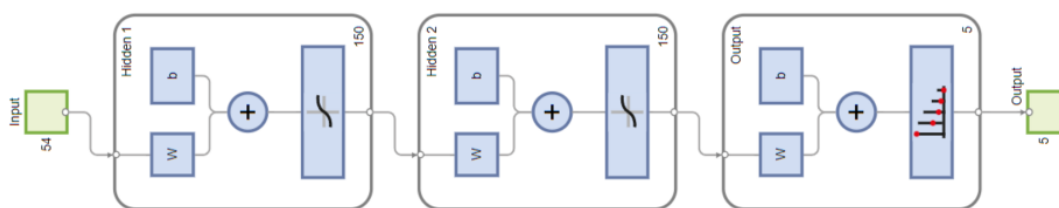
iterációban csak 50 adat esik ki a bemenetből. Egy egyszerű osztással meghatározható, hogy ekkor az ablak mérete az időt tekintve 0,625 másodperc volt. Tehát alaphelyzetben a rendszer ennyi idő alatt több mint 90%-os biztonsággal képes megmondani, hogy milyen talajtípuson mozog. Vizsgáltam ennek a felét és kétszeresét (tehát 50-es és 200-as ablakmérettel) is, az eredményeket táblázatba foglaltam (4. táblázat). A tanítás minden esetben 1000 epoch-számmal, a neuronszám az 50-150 intervallumon 10-es lépésközzel valósult meg, 5 iterációban.

4. táblázat A jellemző alapú neurális hálózatok tanulási eredményei.

Neurális hálózatok	Egy adatsor gyűjtéséhez szükséges idő [s]	Teljes tanulási idő [h]	Elért maximális pontosság [%]
Ablak méret 50, eltolási érték 25	0,3125	0,6	85,30
Ablak méret 100, eltolási érték 50	0,625	0,5	87,90
Ablak méret 200, eltolási érték 100	1,25	0,4	90,47

Látható, hogy a pontosság javult, és sikerült elérni a kitűzött 90%-os határt is, amikor 200-as ablakméretet állítottam be. Ekkor a hálózat 1,25 s alatt képes megmondani, 90,47%-os biztonsággal, hogy milyen talajtípuson történt a mozgás, a betanítottak közül. A tanulási idő is sokat javult, hiszen már az előző néhány százezer bemenet helyett körülbelül tízezer számíthatos volt a legrosszabb esetben is az adatbázis mérete.

A legjobb neurális hálózatnak tehát az előző bekezdésben leírtat veszem. Ebben a hálózatban két rejtett réteg van, rétegenként 150 neuronnal, a kimeneti réteg pedig 5 neuron az osztályok számának megfelelően. A neurális hálózat felépítése a 27. ábrán látható.



27. ábra. A legjobb neurális hálózat felépítése.

További javulást akkor tudnék elérni, ha belevenném az IMU magnetométerének adatait is, vagy ha egyszerre több IMU adatát használnám fel, de mivel a célom gyakorlatilag ezzel is

elértem, a további pontosítás majd egy következő munka során kerülhet elő, ahol növelem majd a bemenet komplexitását.

A legjobb neurális hálózat esetében a következő (28. ábra) konfúziós mátrixot kaptam, mely bemutatja az osztályozások pontosságát. Láthatóan aszfaltra szinte 100%-os pontosságot sikerült elérni, hasonlóan kiemelkedő a homok és a macskakő. Viszont a legtöbb hibás klasszifikálás továbbra is a szántóföld és a fű között található, azonban itt már említettem, hogy maga a talajszerkezet igencsak hasonló volt, a füves területen is voltak fű nélküli részek, illetve a szántóföldön is voltak füves részek, így ez a hiba teljes mértékben indokolt, hiszen valós mérésről beszélünk. Jobban meglepődtem volna, ha ez nem így adódik.

Konfúziós Mátrix a Legjobb Hálózattal (Teljes Adatbázis)						
Output Class	1	2	3	4	5	
	955 43.3%	4 0.2%	0 0.0%	0 0.0%	1 0.0%	99.5% 0.5%
	2 0.1%	329 14.9%	21 1.0%	7 0.3%	6 0.3%	90.1% 9.9%
	0 0.0%	24 1.1%	148 6.7%	52 2.4%	6 0.3%	64.3% 35.7%
	2 0.1%	6 0.3%	61 2.8%	203 9.2%	7 0.3%	72.8% 27.2%
	0 0.0%	2 0.1%	5 0.2%	4 0.2%	358 16.3%	97.0% 3.0%

28. ábra A neurális hálózat konfúziós mátrixa.

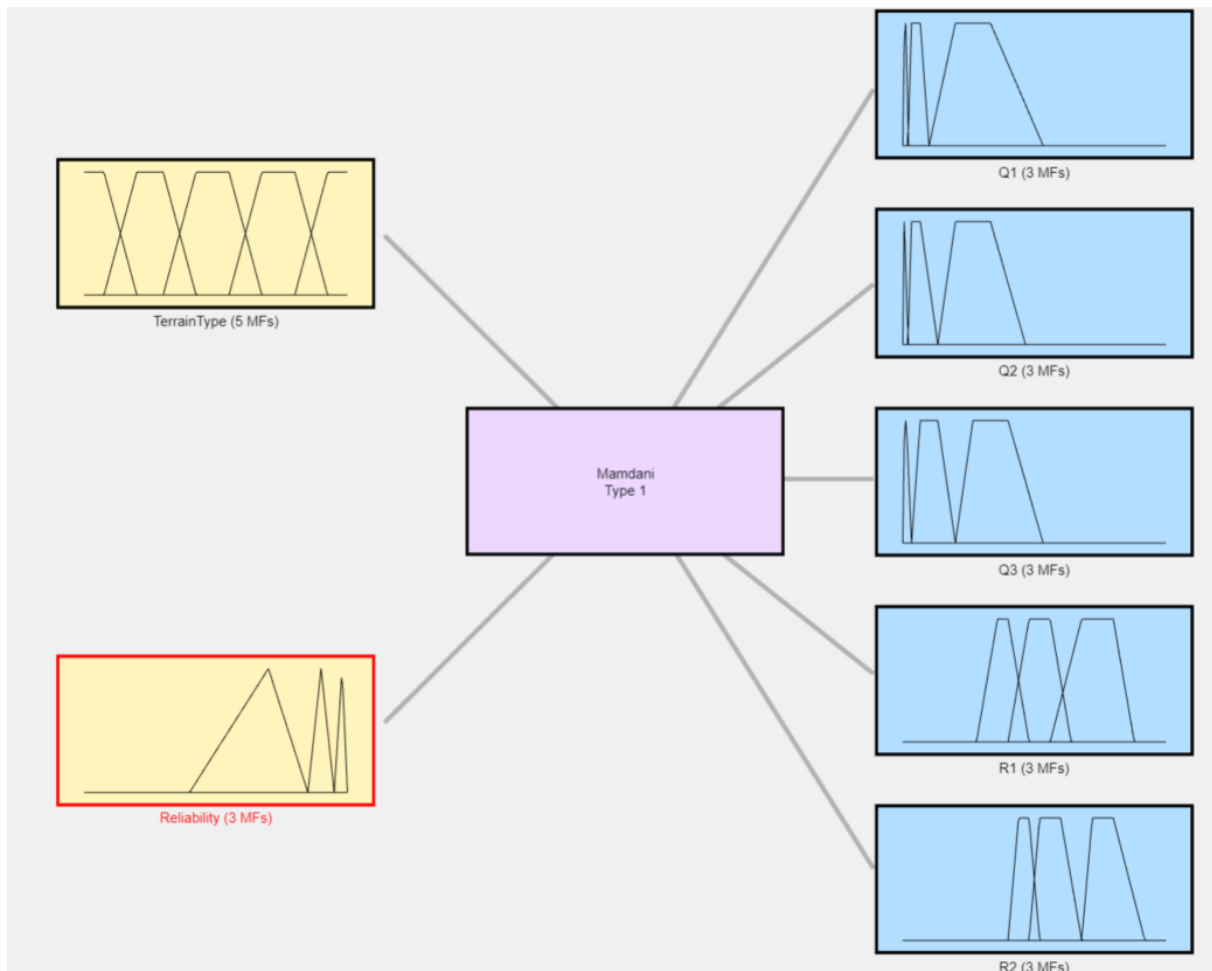
Viszont a következő fejezetben bemutatott rendszer kompenzálni képes az itt kapott bizonytalanságot.

4.6 Fuzzy rendszer alkalmazása a szűrő és a neurális hálózat összekötésére

A Fuzzy következtetési rendszer szintén egy széles körben alkalmazott terület a témakörben, azonban a szakirodalmi résznél sajnos nem túl sok szó eshetett róla, helyhiány miatt. Azért hagytam ki végül ezt a részt is, mivel alkalmazom, viszont nem implementálom a kezdettől fogva.

4.6.1 A megvalósított Fuzzy következtetési rendszer

A rendszerről készítettem egy átfogó ábrát (29. ábra) melyet követően részletesen bemutatásra kerülnek a rendszer komponensei.



29. ábra A Fuzzy következtetési rendszer felépítése

A rendszernek két bemenete van (bal oldal), az egyik a talajtípusnak megfelelő számérték, ami egyben a klasszifikációs neurális hálózat kimenete is. Az így kapott számérték 1-5 között vehet fel értékeket diszkrétan, a megfelelő talajtípus (aszfalt, homok, macskakő, szántóföld vagy fű) szerint. A tagsági függvényt trapéz alakú függvénynek vettem fel, mivel szerettem volna egy kis átfedést, ha esetlegesen hibás a klasszifikáció. Ennek megfelelően vannak egymás mellett a hasonlóbba talajtípusok. A másik bemenet pedig a megbízhatósági érték, melyet jelenleg konstansnak vettem és háromszög alakú görbékkel ábrázoltam, alacsony, közepes és magas tagsági függvényként. Mivel szerencsére az alacsony is 60% körül van, ezért annak a bal oldala teljesen üres.

A jobb oldalon helyezkednek el a kimeneti változók, melyből 5 darab van, darabonként 3 tagsági függvénynel. Az első három változó a Q mátrix 3 elemei (főátló), míg a két utolsó változó az R mátrix elemei.

Középen helyezkedik el a Mamdani-féle következtetési rendszer, melynek egyik legfontosabb általam beállított eleme a szabálybázis, amely az 5. táblázat értékei szerint lett meghatározva.

5. táblázat A Fuzzy szabályrendszer

Szabály	Talajtípus	Megbízhatóság	Q1	Q2	Q3	R1	R2
1	Aszfalt	M	A	A	A	A	A
2	Aszfalt	K	K	K	K	K	K
3	Aszfalt	A	M	M	M	M	M
4	Homok	M	K	K	K	K	K
5	Homok	K	K	K	K	K	K
6	Homok	A	M	M	M	M	M
7	Macskakő	M	A	A	A	A	A
8	Macskakő	K	K	K	K	K	K
9	Macskakő	A	M	M	M	M	M
10	Szántó	M	M	M	M	M	M
11	Szántó	K	K	K	K	K	K
12	Szántó	A	A	A	A	A	A
13	Fű	M	K	K	K	K	K
14	Fű	K	K	K	K	K	K
15	Fű	A	M	M	M	M	M

A táblázatban rövidítve jelöltem az alacsony (A), közepes (K) és magas (M) értékeket, helyszükében, de szerintem így is átlátható. Ezeket a szabályokat pedig a 25. és a 28. ábra hosszas áttekintése után hoztam létre.

Ezzel a teljes Fuzzy következtetési rendszer ismertetésre került, következik a performáciájának vizsgálata.

4.6.2 Eredmények

A Fuzzy rendszer pontosságának értékelésére elkészítettem ugyanazt a táblázatot (6. táblázat), amelyet már használtam a PSO után is, viszont most a Fuzzy rendszerrel megállapított R és Q mátrix alapján futtattam a Kalman-szűrőt. Az értékek minden trajektória esetében javulást mutatnak, tehát létrejött egy olyan adaptív Kalman-szűrő, mely használja klasszifikációs neurális hálózatot és a PSO optimalizálást. Az összekötést Fuzzy rendszer biztosítja.

6. táblázat Az adaptív és az eredeti szűrő összehasonlítása.

	Eredeti RMSE [m]	Fuzzy RMSE [m]	Átlagos RMSE javulás [%]	Eredeti MAE [m]	Fuzzy MAE [m]	Átlagos MAE javulás [%]
Fű	2,753	2,579	6,32	2,431	2,210	9,09
Homok	2,087	1,864	10,69	1,818	1,606	11,66
Macskakő	2,769	2,645	4,48	2,395	2,228	6,97
Szántófield	3,975	3,359	15,50	3,411	2,950	13,52
Aszfalt	3,554	3,204	9,85	3,029	2,749	9,24

Az elvárásom ezzel kapcsolatban mindenképpen teljesült, hiszen valahol az eredménynek valahol az optimális és az eredeti között kell elhelyezkedni, ha jól dolgoztam. Hiszen az eredetinél mindenképpen jobb értékeket kell kapnom, ha ez nincs meg akkor feleslegesen dolgoztam volna az algoritmus kifejlesztésén, az optimális pedig a jelenlegi információk alapján a legjobb megoldás, annál jobbat szintén nem reális kapni erre az algoritmusra.

4.7 Adaptív Kalman-szűrő neurális hálózattal és Fuzzy rendszerrel

Ez a fejezet a teljes feldolgozási folyamat összefoglalása, melyben folyamatábrán kerül bemutatásra az általam megvalósított adaptív Kalman-szűrős algoritmus. Emellett bemutatásra kerül az, hogy mit sikerült elérnem a munkám során.

A megvalósított algoritmus folyamatábrája a 30. ábrán látható. Az adaptív kiterjesztett Kalman-szűrő (EKF) fejlesztése során egy olyan algoritmus került megvalósításra, amely képes kezelni a talajviszonyokból eredő bizonytalanságokat, ezzel biztosítva a robot állapotának pontosabb és robusztusabb becslését.



30. ábra A megvalósított rendszer folyamatábrája

Az algoritmus egy IMU által biztosított gyorsulás- és szögsebesség-adatokból jellemzőket nyer ki, amelyeket egy neurális hálózat használ a talajtípus osztályozására és a megbízhatósági szint meghatározására. Az így szerzett információkat egy Fuzzy következtető rendszer alkalmazza

az R és Q mátrixok dinamikus optimalizálására, ami az EKF teljesítményét adaptív módon szabályozza. Ennek köszönhetően a rendszer képes a talajviszonyoktól függetlenül javítani a mozgásbecslési teljesítményt, minimalizálva az ebből adódó hibákat és növelve a rendszer általános pontosságát.

4.8 Az eredmények értékelése és összehasonlítása hasonló rendszerekkel

Ebben a fejezetben néhány hasonló rendszerrel fogom összehasonlítani az általam megvalósított rendszert.

4.8.1 Lokalizáció

A saját rendszerem performanciájaként a 2. táblázatban kiszámított optimális RMSE értékek átlagait fogom használni. Az RMSE érték így 2,55 méternek adódik.

A lokalizációs eredményeket két ugyancsak kültéri lokalizációs munkával hasonlítom össze. A két rendszer más szenzorokat, algoritmusokat használ viszont a performanciáját mindenképpen leírja az RMSE és AME érték, tehát érdemes összevetni az eredményeket. Ráadásul az én rendszerem low-cost alapon jött létre, míg a következő rendszerek sokkal komplexebbnek és drágábbnak is tűnnek.

Az első rendszer, amellyel összehasonlítom az eredményeimet az Zhang és mtsai. (2016.) által kifejlesztett vezeték nélküli lokalizációs rendszer, mely mély neurális hálózatot (DNN) alkalmaz előfeldolgozott vezeték nélküli szenzorjelekre. Beltéren egész jó eredményeket értek el, kültéren csökkent a teljesítményük, ha az RMSE értékeket nézzük. Egy közepes értéket veszek figyelembe, az eredményeket összefoglaló táblázatukból, mely a DNN(RBM) sor 60N mintamérethez tartozó RMSE érték, azaz 3,29 méter.

A második rendszer, amellyel összehasonlítom az a Yadav és mtsai. (2023) által kifejlesztett Optimalizált Lokalizáció Tanulási Algoritmus (OLLA), ahol a kültéri lokalizáció performanciáját a 8. ábrán láthatjuk. Gyakorlatilag az ott feltüntetett 40 RMSE értékből 39-nél jobb az általam fejlesztett algoritmus.

Vannak természetesen sokkal jobb rendszerek az általam fejlesztettnél, de az előző összehasonlításokkal azt szerettem volna megmutatni, hogy igenis megállja a helyét a munkám.

4.8.2 Klasszifikáció

A klasszifikáció teljesítményét is érdemesnek találom összehasonlításra, hiszen a rendszerem legjobb pontossága 90,47%, amely gyakorlatilag csak azért nem 95-98% mert a talajtípusok (a szántóföld és a fű) nem volt megfelelő mértékben differenciált. Az eredmények szemléltetésére elkészítettem a 7. táblázatot, de előbb bemutatom az ott szereplő rendszereket.

Az első tanulmány az Ahmadi és mtsai. (2021.) által készített félig felügyelt kültéri klasszifikációs neurális hálózatot alkalmazó rendszere. Mivel ők már a tanulást is felügyelték,

és büntetéseket alkalmaztak az esetleges hibás klasszifikációkért, így mindenképpen magasabb szintű rendszernek tekinthető az enyémnél. A 7. táblázatban az az érték fog szerepelni, mikor hozzám hasonlóan 1 szenzort használtak a becslésre. A legjobb pontosságuk sok szenzor és sok szenzorfúzió elérésével $97,64 \pm 1,03\%$ volt. A táblázatban az első tanulmány oszlopában szerepel az eredményük.

A második tanulmány, amely szerepel a táblázatban Wang, Ye és Sun (2021.) által megalkotott adaptív online klasszifikációs algoritmus, mely rezgési jelekre alapszik. A 2. táblázatuk első átlagos pontossági értékét veszem alapul az összehasonlításban. Nekik csak négy csoportjuk volt, és mesterségesen kreált talajon próbálkoztak, így a legjobb pontosságuk elérte a közel 100%-ot, amely nem túl reális cél valós mérés esetében, de laborkörülmények közt elérhető, ahogy nekik is sikerült. A táblázatban a második tanulmány oszlopában szerepel az eredményük.

7. táblázat A klasszifikációs eredmények összehasonlítása

A rendszerek pontossága	Saját rendszer	Első tanulmány	Második tanulmány
Pontosság [%]	90,47	87,74	90,57

A 7. táblázat is mutatja, hogy az általam fejlesztett rendszer megállja a helyét, hiszen a nemzetközi példákhoz hasonló eredményeket kaptam.

5. ÖSSZEFOGLALÁS

A munkám során először összefoglaltam a mobil robot lokalizációhoz használt legfontosabb szenzortípusokat különböző magyar és angol nyelvű szakirodalmak és tanulmányok segítségével. Szintén az irodalomkutatás részhez kapcsolódóan tárgyaltam a különféle szenzorfüziós algoritmusokat, kiemelten a Kalman-szűrős algoritmusokat.

Ezt követően bemutattam a megvalósított mérőrendszer felépítését, melynek alapja egy Ubiquity Robotics Magni mobilis robot. A robotra felszerelésre került két GPS és két IMU modul, melyek pontos típusai megtalálhatók a második fejezetben. Szintén ebben a részben fejtettem ki a megvalósított mérések menetét is.

A harmadik fejezetben bemutatásra került a mérési adatok feldolgozására kifejlesztett algoritmusok értékelése. Először implementáltam, majd optimalizáltam egy Kalman-szűrős algoritmust, az optimalizálást PSO algoritmussal valósítottam meg. Az elért legkisebb RMSE érték 2,55 m volt, amely tekintve, hogy csak a low-cost szenzorok méréseit használtam fel, jónak számít. Az adaptív módszeremmel körülbelül 10 százalékos javulást értem el a lokalizációs teljesítményben. Ezt követően létrehoztam egy talajtípust klasszifikáló neurális hálózatot, mely először nyers IMU adatokkal, később ezekből jellemzők kinyerésével próbálta kitalálni az adott bemeneti értékekhez tartozó talajtípust. Végül 90,47%-os eredményével teljesítette az általam kitűzött minimum 90%-os célt.

A fejezet utolsó részében kifejlesztettem egy adaptív Kalman-szűrőt, melyben a Kalman-szűrő és a klasszifikátor Fuzzy következtetési rendszerrel került összekötésre. Az eredmény egy olyan rendszer, amely a robot alatti talaj bizonytalanságait képes figyelembe venni és kompenzálni a talajtól függetlenül, ezáltal pontosabb állapotbecslést elérve. Emellett sikerült egy nagyobb adatbázist kiépíteni, melyben még számos merítési lehetőség van – például az XSENS adatok, az /odom topic vagy a 9 IMU-ból álló tömb –, melyet fel tudok használni továbbtanulásom során. A jövőbeni tervek között szerepel a pozíció alapú mozgásmodell kiértékelése és összehasonlítása, a 3x3-as Q mátrix helyett a bemeneteket jellemző 2x2-es Q mátrix alkalmazása, valamint a kerékcsúszás detektálása és ennek beépítése a szűrésbe, amellyel tovább javítható a performancia.

IRODALOMJEGYZÉK

- Ahmad, Norhafizan, Raja Ariffin Raja Ghazilla, M. Nazirah Khairi, és Vijayabaskar Kasi. „Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications.” *International Journal of Signal Processing Systems*, 2013.: 256-261.
- Ahmadi, Ahmadreza, Tonnes Nygaard, Navinda Kottege, David Howard, és Nicolas Hudson. „Semi-Supervised Gated Recurrent Neural Networks for Robotic Terrain Classification.” *IEEE ROBOTICS AND AUTOMATION LETTERS*, 2021: 1848-1855.
- Aslam, Muhammad Saad, Muhammad Irfan Aziz, Kanwal Naveed, és Uzair Khaleeq uz Zaman. „An RPLiDAR based SLAM equipped with IMU for Autonomous Navigation of Wheeled Mobile Robot.” *IEEE 23rd International Multitopic Conference*, 2020: 1-4.
- Baharom, Abdul Khaiyum, Rafidah Jamali, Shuzlina Abdul-Rahman, és Sofianita Mutalib. „Towards Modelling Autonomous Mobile Robot Localization by Using Sensor Fusion Algorithms.” *IEEE 10th International Conference on System Engineering and Technology (ICSET)*, 2020.: 185-190.
- Bánlaki, Pál, és Antal Lovas. *SZENZORIKA ÉS ANYAGAI*. Budapest: Typotex Kiadó, 2012.
- Belkin, Ilya, Alexander Abramenko, és Dmitry Yudin. „Real-Time Lidar-based Localization of Mobile Ground Robot.” *14th International Symposium “Intelligent Systems”*, 2021.: 441-448.
- Bräunl, Thomas. *Embedded Robotics*. Perth, Ausztrália: Springer Kiadó, 2006.
- Fazekas, Máté, és Péter Gáspár. „Wheel odometry model calibration with neural network-based weighting.” *Engineering Applications of Artificial Intelligence*, 2024.: 1-10.
- Guo, Fanghong Guo, Hao Yang, Xiang Wu, Hui Dong, Qi Wu, és Zhengguo Li. „Model-Based Deep Learning for Low-Cost IMU Dead Reckoning of Wheeled Mobile Robot.” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 2024.: 7531-7533.
- Han, Soonshin, ByoungSuk Choi, és JangMyung Lee. „A precise curved motion planning for a differential driving mobile robot.” *Mechatronics*, 2008.: 486-494.
- Harindranath, Aparna, és Manish Arora. „A systematic review of user - conducted calibration methods for MEMS-based IMUs.” *Measurement*, 2024: 1-7.
- Huggi, Arpita S, Anushree C Nilavar, Jyoti Bali, Arunkumar Giriyapur, és G K Ashwini. „Implementation of Sensor Fusion for a Mobile Robot Application.” *6th International Conference on Recent Trends on Electronics, Information, Communication & Technology*, 2021.: 15-20.

- InvenSense. *InvenSense ICM-20948 Datasheet*. 2021.
https://product.tdk.com/system/files/dam/doc/product/sensor/motion-inertial/imu/data_sheet/ds-000189-icm-20948-v1.5.pdf (hozzáférés dátuma: 2024.. október 07.).
- Ishihara, Yu, és Masaki Takahashi. „Empirical study of future image prediction for image-based mobile robot navigation.” *Robotics and Autonomous Systems*, 2022.: 1-11.
- Kaplan, Elliott D., és Christopher J. Hegarty. *Understanding GPS Principles and Applications Second Edition*. Norwood, MA: ARTECH HOUSE INC., 2006.
- Kheirandish , Mahsa , Ehsan Azadi Yazdi , Hossein Mohammadi, és Mohsen Mohammadi. „A fault-tolerant sensor fusion in mobile robots using multiple model Kalman filters.” *Robotics and Autonomous Systems*, 2022: 1-22.
- Kurpath, Mohammed Irfan , Adwai Pradeep Kumar, Janardhan Bodireddy, Chandrasekaran K, és Navin Sam K. „An IMUs and potentiometer-based controller for robotic arm-hand teleoperation .” *Sensors and Actuators: A. Physical* , 2024.: 1-2.
- Lee, Junwoo, és Bummo Ahn. „Real-Time Human Action Recognition with a Low-Cost RGB Camera and Mobile Robot Platform.” *Sensors*, 2020.: 1-12.
- Li, Kai, Le Bao, Changsoo Han, Kyoosik Shin, és Wansoo Kim. „IMU-based Online Kinematic Model Parameter Estimation for Four-Wheeled Skid-Steering Mobile Robot Desired Motion Tracking on Different Terrains.” *The 22nd International Conference on Control, Automation and Systems*, 2022.: 1-3.
- Li, Peng, és Xiangpeng Liu. „Common Sensors in Industrial Robots: A Review.” *Journal of Physics: Conference Series*, 2019.: 1-7.
- Li, Zhaodong, Zhibao Su, és Tingting Yang. „Design of Intelligent Mobile Robot Positioning Algorithm Based on IMU/Odometer/Lidar.” *International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, 2019.: 627-629.
- Liu, Yanjie, Chao Wang, Heng Wu, Yanlong Wei, Meixuan Ren, és Changsen Zhao. „Improved LiDAR Localization Method for Mobile Robots Based on Multi-Sensing.” *Remote Sensing*, 2022.: 1-27.
- Lobazev, Nikita, Ekaterina Maslennikova, és Andrey Klovov. „Correction of the Odometry System of a Robotic Wheeled Platform Using GPS RTK Technology.” *Conference of Young Researchers in Electrical and Electronic Engineering (ElCon)*, 2024.: 219-225.
- Madhloom, Ali A. , Firas A. Raheem, és Azad R. Kareem. „A Modified Kalman Filter-Based Mobile Robot Position Measurement using an Accelerometer and Wheels Encoder.” *Engineering and Technology Journal* , 2022.: 267-273.

- Magrin, Carlos Eduardo, Robison Cris Brito, és Eduardo Todt. „A Systematic Mapping Study on Multi-sensor Fusion in Wheeled Mobile Robot Self-localization.” *2019 Latin American Robotics Symposium, , 2019 Brazilian Symposium on Robotics, 2019 Workshop on Robotics in Education*, 2019.: 132-137.
- Meng, Jie, és mtsai. „Efficient and Reliable LiDAR-Based Global Localization of Mobile Robots Using Multiscale/Resolution Maps.” *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 2021.: 1-15.
- Moeller , Ryan , Taher Deemyad, és Anish Sebastian. „Autonomous Navigation of an Agricultural Robot Using RTK GPS and Pixhawk.” *Intermountain Engineering, Technology and Computing (IETC)*, 2020.: 1-6.
- Movella. *MTi Family Reference Manual*. 2020. <https://mtidocs.movella.com/design-2> (hozzáférés dátuma: 2024.. október 05.).
- Mu, Lili, Pantao Yao, Yuchen Zheng, Kai Chen, Fangfang Wang, és Nana Qi. „Research on SLAM Algorithm of Mobile Robot Based on the Fusion of 2D LiDAR and Depth Camera.” *IEEE ACCESS*, 2020.: 157628-157642.
- Panigrahi, Prabin Kumar , és Sukant Kishoro Bisoy. „Localization strategies for autonomous mobile robots: A review.” *Journal of King Saud University –Computer and Information Sciences*, 2022.: 6019-6039.
- Patel, Ujjval N. , és Imraan A. Faruque. „Sensor Fusion To Improve State Estimate Accuracy Using Multiple Inertial Measurement Units.” *IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*, 2021: 1-3.
- Pérez, Luis, Íñigo Rodríguez, Nuria Rodríguez, Rubén Usamentiaga, és Daniel F. García. „Robot Guidance Using Machine Vision Techniques in Industrial Environments: A Comparative Review.” *Sensors*, 2016.: 1-26.
- Robotics, Ubiquity. *Ubiquity Robotics*. 2021. <https://www.ubiquityrobotics.com/products-magni/> (hozzáférés dátuma: 2024.. szeptember 30.).
- Rustam, Ilham , Nooritawati Md Tahir, Ahmad Ihsan Mohd Yassin, Nurbaiti Wahid, és Abdul Hafiz Kassim. „Linear Differential Driven Wheel Mobile Robot Based on MPU9250 and Optical Encoder.” *TEM Journal*, 2022.: 30-36.
- Sarcevic, Peter, és mtsai. „Online Outdoor Terrain Classification Algorithm for Wheeled Mobile Robots Equipped with Inertial and Magnetic Sensors.” *Electronics*, 2023.: 1-16.
- Siegwart, Roland, és Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Cambridge, London: The MIT Press, 2004.

- Skoczen, Magda, és mtsai. „Obstacle Detection System for Agricultural Mobile Robot Application Using RGB-D Cameras.” *Sensors*, 2021.: 1-17.
- Stangebye, Theo , Timothy Mohr, Anna Vallenti, Matt Grauff, és Scott Koziol. „Custom Real-Time-Kinematics Positioning System Testbed for Mobile Robot Localization.” *IEEE 14th Dallas Circuits and Systems Conference (DCAS)*, 2020.: 1-4.
- Techfun. *Techfun.hu*. 2024. <https://techfun.hu/termek/neo-6m-gps-modul> (hozzáférés dátuma: 2024.. november 04.).
- Thrun, Sebastian, Wolfram Burgard, és Dieter Fox. *Probabilistic Robotics*. Cambridge, Massachusetts: MIT Press, 2005.
- Ubiquity Robotics. *Ubiquity Robotics Magni Documentation*. 2022. <https://learn.ubiquityrobotics.com/> (hozzáférés dátuma: 2024.. október 02.).
- U-BLOX AG. *C94-M8P Application Board Setup Guide*. 2017. https://content.u-blox.com/sites/default/files/C94-M8P-Appboard-Setup_QuickStart_%28UBX-16009722%29.pdf (hozzáférés dátuma: 2024.10.03).
- . *Product summary C94-M8P application kit*. 2021. https://content.u-blox.com/sites/default/files/C94-M8P-AppBoard_ProductSummary_UBX-15024894.pdf (hozzáférés dátuma: 2024.. október 03.).
- Wang, Mingming, Liming Ye, és Xiaoyun Sun. „Adaptive online terrain classification method for mobile robot based on vibration signals.” *International Journal of Advanced Robotic Systems*, 2021.: 1-14.
- Xsens Technologies B. V. *XSENS*. 2020.. Augusztus. <https://www.xsens.com/hubfs/Downloads/Leaflets/MTi%20600-series%20Datasheet.pdf> (hozzáférés dátuma: 2024.. október 05.).
- Yadav, Preeti, S. C. Sharma, Omkar Singh, és Vinay Rishiwal. „Optimized Localization Learning Algorithm for Indoor and Outdoor Localization System in WSNs.” *Wireless Personal Communications*, 2023.: 651-672.
- Yu, Hong, és Liuyang Kong. „Autonomous Mobile Robot Based on Differential Global Positioning System.” *IEEE International Conference on Mechatronics and Automation*, 2018: 392-395.
- Yu, Shibing, és Zhen Jiang. „Design of the navigation system through the fusion of IMU and wheeled encoders.” *Computer Communications*, 2020.: 730-737.
- Zhang, Wei, Kan Liu, Weidong Zhang, Youmei Zhang, és Jason Gu. „Deep Neural Networks for wireless localization in indoor and outdoor environments.” *Neurocomputing*, 2016.: 279-287.