



SZEGEDI TUDOMÁNYEGYETEM
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
MÉRNÖKI KAR

***IMU/WIRELESS TECHNOLOGIÁKON ALAPULÓ BELTÉRI
LOKALIZÁCIÓS ALGORITMUS KIFEJLESZTÉSE***

TDK DOLGOZAT

Magony Gergely

II. évfolyamos okleveles mechatronikai mérnök

hallgató

Témavezetők:

Dr. Odry Ákos

Főiskolai docens

Dr. Sarcevic Péter

Főiskolai docens

SZEGED

2022

TARTALOMJEGYZÉK

TARTALOMJEGYZÉK.....	1
1. BEVEZETÉS.....	3
1.1. A munka előzményei	4
1.2. Megoldásra vezető út	6
1.3. Alkalmazott módszerek	7
2. SZAKIRODALMI ÁTTEKINTÉS.....	7
2.1. Pozíció és orientáció mérés magneto-inerciális érzékelőkkel	7
2.2. Kvaternió alapú orientáció reprezentáció	9
2.3. Orientáció mérése MARG szenzorokkal	10
2.3.1. Giroszkóp modell	11
2.3.2. Gyorsulásérzékelő és magnetométer modellek.....	11
2.4. Térbeli orientáció becslése Kiterjesztett Kalman-szűrővel	13
2.4.1. Pozíció és orientáció mérése MARG segítségével	14
3. MÉRŐRENDSZER FELÉPÍTÉSE	15
3.1. Az inerciális mozgások meghatározása.....	16
3.2. A ground truth kinyerése.....	16
3.3. Az abszolút pozíció meghatározása	17
3.4. Az effektor felépítése	18
4. MÉRÉSADATGYŰJTŐ KERETRENDSZER	20
4.2. ROS alapfogalmak.....	20
4.3. A ROS Node-ok megírása	21
4.3.1. Az IMU Node implementálása	22
4.3.2. Az IMU kiolvasása	23
4.3.3. ROS Node beépítése, és a publikálás	23
4.3.4. A Node tesztelése	25
4.3.5. Az UWB modul programjának megírása, és a node kifejlesztése.....	26
4.3.6. Az "Anchor" modulok implementálása	26
4.3.7. A "Tag" modul implementálása	27
4.4. Az UR driver és a koordináták meghatározása	30
5. MÉRÉSEK	32
5.1. Magnetométer kalibráció	33
5.1.1. Keményvas hatás	33
5.1.2. Lágyvas hatás	34

5.1.3.	Nem stacionárius keményvas hatások (mozgó vas).....	34
5.1.4.	Magnetométer kalibráló mérés	35
5.2.	Fingerprint felvétel és az abszolút pozíció becslésére alkalmas rendszer betanítása	36
5.2.1.	A fingerprint trajektória felvétele	37
5.2.2.	Pozícióbecslés mesterséges neurális hálókka.....	38
5.3.	Az UR5 trajektóriák megalkotása.....	40
6.	UWB FINGERPRINT-MARG SZENZORFÚZIÓ MEGVALÓSÍTÁSA.....	42
6.1.	Kiterjesztett Kalman szűrő predikciós fázisa.....	43
6.2.	Kiterjesztett Kalman szűrő korrekciós fázisa.....	46
7.	EREDMÉNYEK	49
7.1.	Első szcenárió	50
7.2.	Második szcenárió	52
9.	KONKLÚZIÓ.....	53
	IRODALOMJEGYZÉK.....	55
	KÖSZÖNETNYÍLVÁNÍTÁS.....	59

1. BEVEZETÉS

Manapság egyre elterjedtebb a robotok használata az iparban, ezen belül pedig a mobilrobotok használata különböző feladatokra. Az autonóm rendszerek rengeteg szegmensben megkönnyítik, segítik az emberi munkát, ezért fejlesztésük nagy léptékekkel halad előre. Az ipari robotokkal szemben a mobil robotoknál egy kardinális téma a lokalizáció. A pontos lokalizáció kulcsfontosságú a megbízható szabályozások és irányítások megvalósításához. Ha bizonytalan a lokalizációs performancia (azaz a becsült állapotokat jelentős zaj és ofsztet jellemzi), akkor az a visszacsatolt rendszert alkalmatlan beavatkozások fogják meghajtani a szabályozó algoritmusokon keresztül. Az alkalmatlan beavatkozások hibás működést, vagy akár kárt is okozhatnak a rendszerben és környezetében (Thanh és Hong, 2018).

Külföldön már léteznek kifejlesztett ipari lokalizációs rendszerek, melyek használata már kiforrottnak tekinthető (ilyenek például a GNSS/INS modulok az Xsense-től, www.xsense.com). Ezek a modulok általában GPS segítségével biztosítják az abszolút pozíció méréseket, míg IMU, magnetométer, nyomás- és hőmérsékletérzékelők segítségével adnak rövid távú információkat egy a modulba implementált állapotbecslő algoritmusnak (pl. Kalman szűrő vagy annak kiterjesztett változata).

Viszont beltérben a sikeres lokalizációt nehezebb kifejleszteni, hiszen a GPS, mint abszolút információkat szolgáltató modul helyett alternatív beltéri megoldások becsatolása szükséges a problémába. Ipari kifinomult drága megoldások kamerarendszereket telepítenek (ilyenek például a szinkronizált és kalibrált multi-kamera rendszerek a Vicon-tól, www.vicon.com). Ezek a rendszerek megbízható abszolút pozíció méréseket biztosítanak beltérben a mobil robot pozíció és orientáció becsléséhez, azonban az áruk ezeknek a rendszereknek nagyon magas. A másik elterjedt megoldás a mobil robotra erősített LiDAR (Light Detection and Ranging – lézer alapú távolságérzékelés) érzékelő használata (pl. a SICK gyártótól, www.sick.com). Azonban ez a technológia a korábbi becslésre építkezik az abszolút pozíció meghatározásánál (hiszen az érzékelő koordináta rendszerében állnak rendelkezésre a távolság értékek), így nem tekinthető teljesen független abszolút pozíciót szolgáltató információ forrásnak (Hashim, 2021).

A low-cost megoldásokat tekintve a beltéri lokalizációs probléma egy kiforratlan terület, fejlődő tudományág. Rengeteg szenzor áll már a rendelkezésünkre, ami segíti a helyes relatív vagy abszolút pozíció mérést. A relatív érzékelők esetében az enkóderek (kerekes robotok esetében), kamerás elsodródásszenzorok (Optical Flow) (melyek képi információ

differentiálásával adják meg a kamera sebességét) és a szögsebességet szolgáltató giroszkópok vannak elterjedten használva mobil robotoknál. Az abszolút érzékelők csoportját a gyorsulásmérő, magnetométer, LiDAR, és rádiós modulok, mint Wifi RSSI, UWB alkotják. Ezeknél az érzékelőknél a mérések felhasználását (pl. egy Kalman szűrő algoritmusban) a nyers adatok szűrése és kalibrálása előzi meg, ezzel biztosítjuk a mérési hibák és bizonytalanságok eliminálását a lehető legnagyobb mértékben, ugyanakkor, ezek járulnak hozzá a különböző algoritmusok hatékony alkalmazásához, melyek a mobil robot pózt (pose – a robot konfigurációs terének változóit tartalmazó több dimenziós vektor) állítják elő szenzorfüzítő alkalmazásával (Kok, Hol, Schön, Gustafsson és Luinge, 2012), (Papafotis és Sotiriadis, 2019).

A lokalizáció probléma megoldása egy fontos mérföldkő a visszacsatolt rendszer irányításában, azonban a rendszer megfelelő irányításért a szabályozó algoritmusok felelnek. Az alkalmas visszacsatolást a lokalizációs algoritmus biztosítja. Az algoritmus beállítását a hasonló fontossággal bíró szabályzó tervezés követi, mely feladata a megfelelő aktuátor működés biztosítása; ez a rendszert a kívánt referencia jelek környezetében tartja.

Fontos nyomatékosítani, hogy a lokalizációs probléma egy komoly mérföldkő és ugyanakkor kritikus pont is a visszacsatolt rendszerek (főleg instabil mobilis rendszerek) irányításának megtervezésében. Ha a pozíció és orientáció mérés (azaz a becsült robot pose) bizonytalan, zajos és jelentős hibával írja le a valós állapotokat, akkor az váratlan, instabil helyzeteket idéz elő, hiszen a szabályozó algoritmusokat nem a valós rendszer állapotai alapján üzemeltetjük. A megbízható irányítás első lépése tehát a megbízható lokalizáció biztosítása, hiszen a visszacsatolt rendszerben az előbbi az utóbbi eredményeit használja fel.

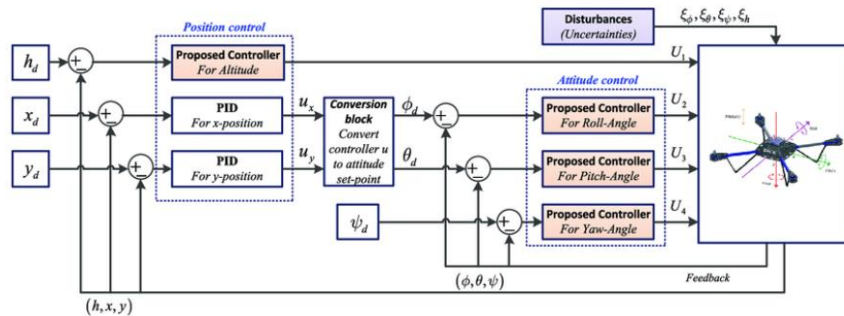
1.1. A munka előzményei

Az alapképzésben egy quadcopter vezérlésének implementálásával foglalkoztam (1. ábra). Az inerciális mérések feldolgozása, az orientáció meghatározása, majd a PID szabályzók beállítása során előtérbe került egy komoly megoldatlan probléma, ez a pozíció és orientáció megbízható mérése volt. Dokumentáltam, hogy a bemutatott rendszert nem tudtam tökéletesen stabilizálni, hiszen a pozíció és orientáció mérést komoly bizonytalanságok jellemezték. A propellerek vibrációi és külső zavarások jelentős zajt indukáltak a mérési eredményekbe, mely azt eredményezte, hogy a bonyolult visszacsatolt rendszert nehéz volt stabilizálni. A további problémát az jelentette, hogy nem lehetett következtetni a beállított Kalman szűrő performanciájára, hiszen a referencia (ground truth) mérések nem álltak rendelkezésre.



1. ábra - Az alapképzés alatt kifejlesztett quadcopter (Magony, 2020)

Felmerült több probléma is a quadcopter irányításával kapcsolatban, hiszen a négy rotoros mobil robotok irányítása nagyon komplex rendszert alkot (lásd 2. ábra) (Thanh és Hong, 2018). A megbízható lokalizációhoz kifinomult érzékelők telepítésére lett volna szükség, ilyenek például az Xsens modulok (www.xsens.com), és Vicon (www.vicon.com) kamera rendszerek. A fenti szenzorok nagyon költségesek, az előbbi 500 dollár környékén érhető el, az utóbbi 10000 dolláros tételben mérhető.



2. ábra - A quadcopter irányítás hatásvázlata (Thanh és Hong, 2018)

Mivel a rendszer instabil, valamint jelentős zaj és zavarás jelentkezik a low-cost érzékelők mérési eredményeiben, ezért az erre az eredményekre alapozott PID szabályzók beállítása nehéz. Sőt a sok esetben használhatatlan mérések alapján kiszámított PID beavatkozások kritikus manővereket eredményeztek, mely veszélyessé tette beltérben a mérést. Megfigyeltem, hogy ilyen esetekben kis hiba is nagy kárt okoz.

Megismertem a fenti bonyolult rendszer nehézségeit, majd a beltéri lokalizáció területén kezdtem el informálódni, kutakodni az újabb megoldások iránt. Több szenzortípust is vizsgáltam a kutatásaim alatt. A fő érdeklődésemet a low-cost technológiák integrálása irányába tereltem, melyben kutatásával lokalizációs módszereket fejleszthetek és értékelhetek ki. Ehhez egy általános mérőrendszert céloztam meg kifejleszteni, mely egy szenzorszigetből áll, és

lehetővé teszi, hogy a különböző low-cost technológiák méréseiből pozíció- és orientáció (azaz póz/pose) becselő algoritmusokat fejlesszek. Céлом egy biztonságosabb beltéri lokalizációt megvalósító rendszer kiépítése, melyben low-cost technológiákat alkalmazhatok. Nevezetesen, egy olyan rendszerben gondolkodtam, mely lehetővé teszi a lokalizációs algoritmusok beállítását és performanciáinak kiértékelését azelőtt, hogy azokat a visszacsatolt rendszerben alkalmazzam.

1.2. Megoldásra vezető út

A megoldás a probléma izolálása volt a szabályozó algoritmusoktól, azaz egy olyan környezet kiépítése, melyben csak a lokalizáció minőségének mérésére tudtam koncentrálni. Céлом az volt, hogy a rendszerben low-cost szenzorokat alkalmazzak, azok kalibrációját, szűrését és szenzorfüzióját megvalósítsam, majd a szenzorfüzió minőségét kiértékeljem. Ez az analízis lehetővé teszi azt, hogy a kifejlesztett algoritmust direkt visszahelyezzük a visszacsatolt rendszerbe, hiszen annak performanciáját ellenőriztük.

Szükség volt egy multiszenzoros panelre, mely több szenzonnal felvértezve az dinamikus mozgások mérését teszi lehetővé. A szenzorszigetet felvérteztem inerciális mérőmodulokkal (Inertial Measurement Unit – IMU), melyek tartalmaznak giroszkópot, gyorsulásmérőt, valamint magnetométert is. Az abszolút pozíciók biztosításához vezeték nélküli technológiákat választottam, nevezetesen, RSSI-alapú (Received signal strength indication – Fogadott jel erősség jelzés) méréseket, valamint ezt az irányvonalat tovább vizsgálva az UWB (Ultra Wideband – ultra szélessávú) technológiákat alkalmaztam. A szenzorfüzió kiértékeléséhez szükség volt még egy eszközre, mely a valós (ground truth) abszolút pozíciót és orientációt biztosítja, azaz a kifejlesztett szenzorszigetnek a világkoordinátában elfoglalt pose értékeit biztosítja a különböző mozgások alatt. Ezen ground truth értékeket felhasználva az UWB modulok mérései is validálhatók válhatnak. További fontos szempont volt még a mérések biztonságossá tétele és megismételhetősége. Emiatt került bevonásra egy UR5-ös ipari robot, melynek a csuklókoordinátaiból direkt geometriával számolható az effektor pozíciója. A kifejlesztett szenzorsziget az UR5 end-effektor szegmensre lett telepítve, így alkotva a teljes mérőrendszert.

1.3. Alkalmazott módszerek

Első lépésként az UR5 robothoz csatoltam a kifejlesztett szenzorszigetet. Ezt követően az UR5 konfigurációs terében pásztázó mozgásokat végeztem. A pásztázó mozgások alatt az UWB modulok méréseit és a pillanatnyi end effektor pozíciót és orientációt (pózt) regisztráltam. Az end-effektor póz alkotta a ground truth adatokat, hiszen az UR5 kalibrált ipari robot, melynek bázis koordináta rendszerében pontos end-effektor póz számítható a pillanatnyi csuklókoordináták alapján a direkt geometria alkalmazásával. Az így felépített adatbázist használtam fel egy neurális hálózat kifejlesztésére, mely abszolút pozíció mérést valósít meg. Ezt követően a szenzorszigetre telepített IMU és magnetométer szenzorok illesztését végeztem el. A magnetométer zavarásokat kompenzáltam egy kalibrációs eljárással. Végül kifejlesztettem az UWB/IMU/Magnetométer fúziót egy kiterjesztett Kalman szűrő keretrendszerben a pozíció és orientáció javított minőségű becslésére. A ground truth ismeretében kiértékeltem a kifejlesztett algoritmus hatékonyságát és meghatároztam, hogy milyen mértékű javulást eredményez a javasolt algoritmus a neurális hálózathoz képest. A dolgozatomat a továbbfejlesztési lehetőségek ismertetésével zárom. A következő fejezetekben ezek alapján mutatom be az elvégzett munkámat.

2. SZAKIRODALMI ÁTTEKINTÉS

2.1. Pozíció és orientáció mérés magneto-inerciális érzékelőkkel

A lokalizációs probléma célja a robot pillanatnyi pozíciójának és orientációjának meghatározása, melyre röviden robot pózként (robot pose) hivatkozik a szakirodalom. A probléma körültekintést igényel, hiszen mind az útvonaltervező, mind a vezérlő algoritmusok a kapott eredményekre támaszkodnak (ez látható a quadcopter esetében is a 2. ábrán). A megbízhatatlan és pontatlan pose-ok kimozdíthatják a stabilitásból a rendszert és ezáltal akár kárt is okozhat a környezetében. A legújabb fejlesztések a problémát mind új technológiákkal és új szenzorfüziós eljárásokkal próbálják megoldani. Az inerciális mérőfelület gyorsulásérzékelőt és giroszkópot tartalmaz; ennek magnetométerrel kibővített változatát MARG (Magnetic, Angular Rate, and Gravity) modulnak hívják. Az IMU és MARG egységek alapvető szenzoroknak számítanak a mai mobilrobotokban. Ezek alkalmazása nagy figyelmet kapott a lokalizációban egyszerű interfészük, alacsony költségük, és a nagy felbontású méréseik miatt. A MARG alapú pose mérés egy nehéz feladat, mert a zaj és a változó ofszet megjelenik a mérésekben, továbbá az orientáció becslési pontosság is befolyásolja a kettős integráció alapú pozíció számításokat. A külső hatások (vagyis a rendszerdinamika és mágneses zavarok)

tovább rontják az orientáció becslés megbízhatóságát. Az anomáliákat általában referencia jelekkel javítjuk, például beltérben tereptárgyak észlelésével, kültéren pedig globális helymeghatározó rendszeri (GPS) mérésekkel (Zhao, Zhang, Wang, Nogueira és Scherer, 2021), (Cioffi és Scaramuzza, 2020), (Li, Wang, Zhuang és Yan, 2019). Azonban, ha a frissítő jelek szórnak, valamint ritkán elérhetőek el a környezeti feltételek vagy a rendszer korlátoltsága miatt (Szrek, Trybala, Goralczyk, Michalak, Ziketek és Zimroz, 2021), (Hashim, 2021), akkor a MARG-alapú pozíció mérést hosszabb ideig kell végrehajtani (Hashim, Abouheaf és Abido, 2021), (Hashim és Eltoukhy, 2020).

A mobilrobotok pozíciójának és orientációjának MARG-alapú mérése egy kiterjedten kutatott téma. Általában a megközelítések újszerű matematikai modelleket, egyedi kényszereket, és/vagy rekurzív Bayes-i keretrendszerbeli virtuális méréseket (Akai, Morales és Murase, 2018), (Silva, do Monte Lima, Uchiyama és Taniguchi, 2019) használnak. Az irodalom gyakran hivatkozik Unscented Kalman Filter (UKF) alkalmazására, melyben a dinamikát leíró hatások approximációjával több hatás is hatékonyan modellezhető; például ilyen a motor sebesség és nyomaték egyenletei egy egyedi folyamatmodellben (Svacha, Paulos, Loianno és Kumar, 2020). Hasonlóképpen, multitörörök komplex dinamikai modelljét is használták már ilyen keretrendszerben (Macdonald, Leishman, Beard és McLain, 2014), ami megbízható állapotbecsléseket szolgáltatott, amikor ritka referencia jelek álltak rendelkezésre. A kifinomult modellek nehéz paraméter beállítási eljárásokat eredményeznek. Ezek a beállítási eljárások általában mind a mérést, mind a zaj azonosítását magába foglalják, és kijelölt laboratóriumokban valósíthatók meg.

A közelmúltbeli fejlesztések adatvezérelt megközelítéseket javasolnak és fuzionálják az állapotbecslőket a gépi tanulási eszközökkel, például: konvolúciós (Convolutional Neural Network - CNN) és mély neurális hálózatok (Deep Neural Network - DNN) (Esfahani, Wang, Wu és Yuan, 2019). Sikeres megközelítések lettek kidolgozva a gyalogos holtpontszámítási problémákra (Chen, Zhou, Bao, Liu, Gu, Li, Zhao, Zhu és Lia 2021), (Wang, Cheng, Wang és Meng, 2021), azonban ezeket az alkalmazásokat lassabb dinamika jellemzi, mint a mobilrobotokat. A felügyelt tanulás visszafejtette a sebességeket az IMU jelekből, így kompenzálva a gyorsulásmérő bias jelét, ami pontos kettős integrációhoz vezet (Yan, Shan és Furukawa, 2018). A neurális hálózatok (Neural Network - NN) és a Kiterjesztett Kalman-szűrő (Extended Kalman filter - EKF) fúziója is kielégítő IMU-alapú pose számítást biztosít limitált csúszási (drift) hibával (Liu, Caruso, Ilg, Dong, Mourikis, Daniilidis, Kumar és Engel, 2020). A Kalman-szűrő (Kalman Filter - KF) és mély neurális hálózat (DNN) sikeres alkalmazását

javasolták a kerekes mobilrobotok IMU-alapú póz számításához (Brossard, Barrau és Bonnabel, 2020). A munka egy virtuális megfigyelés-alapú állapotter-modelt használt a valós idejű kovariancia adaptációhoz, hogy hatékonyan kompenzálja mind a zajt, mind a bias összetevőket. Egy hasonló DNN topológia sikeres giroszkópzaj kompenzációt valósított meg, ami egy megbízható nyílt-hurkú integráción alapuló orientáció meghatározásra vezetett (Brossard, Bonnabel és Barrau, 2020), (Kim, Seo és Seo, 2021). MARG-alapú pose becsléshez EKF és CNN topológiát is javasoltak, majd a megvalósított dinamikus kovariancia optimalizálás hosszú távú stabil navigációs eredményeket eredményezett (Roux, Changey, Weber és Lauffenburger, 2021).

A gépi tanulás alapú módszerek kiváló becslési teljesítményt nyújtanak, azonban alkalmazhatóságuk meglehetősen korlátozott, mivel a legtöbb mobilrobotot szabványos beágyazott rendszerek működtetik, amik nem tudnak megbirkózni ekkora számítási kapacitásokkal.

2.2. Kvaternió alapú orientáció reprezentáció

Az irodalmi hivatkozásokban E és S keretek (vagy frame-ek) segítségével írják le az orientációt. E és S a földi és szenzor frame-ek, más néven az inercia és mozgó koordináta rendszerek. A kereteket a hagyományos North-East-Down (NED) konfigurációval adjuk meg, amit gyakran használnak robotos alkalmazásoknál (Roh és Kang, 2018), (Liu, Lyu, Lai, Yuan és Wang, 2019), (Zhang és Liao, 2017). Az E és S közötti leképezést egy rotációs mátrix adja:

$${}^E x = {}^E_S R {}^S x \quad (1)$$

ahol ${}^E x$ és ${}^S x$ jelöli a 3x1-es megfigyelés-vektorokat a földi és szenzor keretben.

A kvaternió reprezentáció hatékony módot biztosít mind a fentebb említett rotációs mátrix meghatározására, mind a koordináta-rendszerek helyzetének háromdimenziós térben történő leírására (Kuipers, 1999). Ez az előnyös struktúra gyors számítást tesz lehetővé és teljesen elkerüli az Euler-szögek jól ismert szingularitás problémáját is (más néven a „gimbal lock” probléma) (Diebel, 2006). A négydimenziós vektor ${}^E_S q \in \mathbb{R}^4, \| {}^E_S q \| = 1$ írja le az E frame térbeli helyzetét az S framehez képest úgy, mint egy μ szöggel történő elforgatást az $e = (e_x, e_y, e_z)^T$ egységvektor körül, ami a forgástengelyt jelöli S-ben. Ezt a forgási quaterniont úgy értelmezzük, mint ${}^E_S q = (\cos \frac{\mu}{2}, e^T \sin \frac{\mu}{2})^T = (q_0, \boldsymbol{q})^T$, ahol q_0 és $\boldsymbol{q} = (q_1, q_2, q_3)^T$ jelölik a skaláris és a vektor rész tagokat. A koordináta-transzformációt a nem-kommutatív kvaternió szorzat hajtja végre, amit a $\otimes$ jelöl:

$${}^E x = {}^E_S q \otimes {}^S x \otimes {}^E_S q^* \quad (2)$$

A (2)-es egyenletben a ${}^E_S q^* = (q_0, -q)^T$ jelöli a konjugált kvaterniót (az inverz rotáció, ami az ${}^E_S q^* = {}^E_S q$ -ként van meghatározva). Továbbá az ${}^E x$ és ${}^S x$ jelölik a vektormegfigyelésekhez kapcsolódó quaternionokat a zéró skalár résszel ($q_0 = 0$, mint $x = (0, x^T)^T$).

Jelölje ${}^S \omega = (0, \omega_x, \omega_y, \omega_z)^T$ a szögsebességek által alkotott négydimenziós kvaterniót az S frameben az x, y, z tengelyek körül. A kvaternió időderiváltja:

(3)

$${}^E_S \dot{q} = \frac{1}{2} {}^E_S q \otimes {}^S \omega = \frac{1}{2} Q(q) {}^S \omega, \quad Q(q) = \begin{bmatrix} q_0 & -q^T \\ q & q_0 I_3 + [q \times] \end{bmatrix}, \quad [q \times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}.$$

ahol a mátrix-vektor szorzatot a $Q(q)$ jelöli. Az E frame S-hez képesti térbeli helyzetét a quaternion derivált ${}^E_S \dot{q}$ integrálásával kapjuk. A továbbiakban az egyszerűség kedvéért az alsó- és felső indexeket elhagyom.

Az Euler-szögeket használjuk a térbeli helyzet becslésének minőségi értékelésére, mivel az értelmezésük egyértelmű, könnyen feldolgozható. Az Euler-szögeket (yaw, pitch és roll) ψ , θ , és ϕ jelölik a z, y, és x-tengelyek körül.

$$\phi = \arctan2(2q_2q_3 - 2q_0q_1, 2q_0^2 + 2q_3^2 - 1), \quad (4)$$

$$\theta = -\tan^{-1}\left(\frac{2q_0q_2 + 2q_1q_3}{\sqrt{1 - (2q_0q_2 + 2q_1q_3)^2}}\right),$$

$$\psi = \arctan2(2q_1q_2 - 2q_0q_3, 2q_0^2 + 2q_1^2 - 1).$$

2.3. Orientáció mérése MARG szenzorokkal

A MEMS-alapú MARG egységek mindegyik szenzora hasznos információt nyújt a pillanatnyi térbeli orientációról; azonban egyik szenzor sem képes önmagában megbízható térbeli helyzetbeli eredményeket nyújtani. A giroszkópok szögsebességet mérnek; ezért giroszkóp-alapú helyzet megvalósulást kapunk numerikus integrálással, de a mérések tartalmazznak mind hőmérsékletfüggő offsetet, mind zajt, amik kumulatív hibákat okoznak. Egy gyorsulásmérő a gravitációs és a külső gyorsulások összegét adja vissza. Mozdulatlan helyzetekben, hosszú távú stabil térbeli orientáció mérés érhető el az érzékelt gravitációs vektor felbontásával, de mivel a külső gyorsulások nőnek a dinamikus mozgás eredményeként, ezért az orientáció mérés minősége drasztikusan romlik. Magnetométerek a geomágneses teret

mérik, amiket az irány meghatározására használhatunk. Azonban a környezet ferromágneses tárgyak zavarása által okozott mágneses fluktuációja nagymértékben zavarja a magnetométer kimenetét.

2.3.1. Giroszkóp modell

Jelölje Ω_k egy háromtengelyű MEMS giroszkóp nyers mérési vektorát a k -adik mintavételi időben. Ez a mérési vektor az x , y , z -tengelyek körüli valós szögsebességek ω_k (3×1 -es vektor), egy $\omega_{o,k}$ jelű vektorból, ami a nem-állandó bias értékeket tartalmazza, és egy additív mérési zajokat tartalmazó vektorból áll. A gyártás tökéletlenségének eredménye, hogy a szenzormodell kiterjesztik a nem-ortogonalitást és érzékenységet jellemző hibákkal, amiket a 3×3 -as M_Ω és ΔS_Ω mátrixok képviselnek. Továbbá, a szenzor hőmérséklet-érzékenysége a lassan változó bias vektort egy „random walk” folyamatként írja le, amit egy $\eta_k^{\omega_0}$ zaj karakterizál (Aggarwal, 2010)

$$\Omega_k = (I + \Delta S_\Omega) M_\Omega \omega_k + \omega_{o,k} + \eta_k^\Omega \quad (5)$$

$$\omega_{o,k} = \omega_{o,k-1} + \eta_k^{\omega_0}$$

A fenti modellben feltételezik, hogy a zajvektorok nulla átlagú fehér Gauss-változók. Giroszkóp alapú térbeli orientáció mérést a valós szögsebességvektor ω_k numerikus integrálásával valósítjuk meg az (5) egyenlet felhasználásával. Tökéletes gyártási folyamatot feltételezünk, ez $M_\Omega = I$ és $\Delta S_\Omega = 0$ változókat eredményez (Markley és Crassidis, 2014), (Nowicki, Wietrzykowski és Skrzypczynski, 2015), (Höflinger, Müller, Zhang, Reindl és Burgard, 2013).

$$q_{k+1} = q_k + \frac{T_s}{2} Q(q_k) \begin{bmatrix} 0 \\ \Omega_k - \omega_{o,k} \end{bmatrix}, \quad (6)$$

A (6) összefüggésben $T_s = 1/f_s$ a mintavételi idő. Ez a módszer csak rövid távú pontosságot ad. A bias és mérési zajok jelenléte miatt csúszás jelentkezik az így meghatározott orientációban.

2.3.2. Gyorsulásérzékelő és magnetométer modellek

A gyorsulásmérő és magnetométer szenzorok abszolút referencia megfigyeléseket biztosítanak, ezért a méréseik kombinálhatók a szenzorok teljes térbeli orientációjának meghatározásához. A háromtengelyű MEMS gyorsulásmérő nyers kimenete A_k négy fő alkotóelemből áll: a szenzor frameben (S) mért gravitációs és külső gyorsulási vektorok g_k és α_k , a bias összetevők vektora a_0 , és az additív mérési zajok vektora. Továbbá a háromtengelyű

magnetométer modell nyers mérési vektora az S-ben érzékelt valódi lokális mágneses mezőből h_k , a szenzor bias vektorából h_0 , és a mérési zajvektorból áll:

$$\begin{aligned} A_k &= (I + \Delta S_A) M_A (\alpha_k + g_k) + a_0 + \eta_k^A, \\ H_k &= (I + \Delta S_H) M_H (B_{si} h_k + b_{hi}) + h_0 + \eta_k^H. \end{aligned} \quad (7)$$

Hasonlóan a giroszkóp modellhez, a Gauss-zajokat figyelembe veszik a fent említett modellekben; A gyártási hibákon (ΔS_A , ΔS_H , M_A , és M_H) kívül a magnetométer méréseit zavarja a lokális környezet mágneses lágyvas és keményvas okozta hibák is, amit a 3×3 -as B_{si} mátrix és a 3×1 -es B_h mátrix képviselnek. Ezeket a modellhibákat kalibrációs eljárásokkal kompenzálják a mérések eloszlásának feltérképezésével egy ellipszoidon (Kok, Hol, Schön, Gustafsson és Luinge, 2012), (Papafotis és Sotiriadis, 2019), (Sarcevic, Kincses és Pletl, 2019). Ha a kompenzáció megtörténik (tehát $h_k := B_{si}^{-1} h_k - b_{hi}$), és a gyártás tökéletes, akkor a fenti modellek jelentősen leegyszerűsödnek.

Ha nem gyorsul a rendszer, azaz $\alpha_k \approx 0$, és ezenfelül, ha a magnetométert nem zavarják ferromágneses tárgyak, akkor a referencia vektorok és mérések közötti összefüggést a rotációs mátrix írja le:

$${}^S A_k = {}^S R(q_k) {}^E g, \quad {}^S H_k = {}^S R(q_k) {}^E h. \quad (8)$$

A fent említett konfigurációban a gravitációs vektor ${}^E g$ -ként van megadva, míg a mágneses tér vektora ${}^E h$ SI egységekben adott.

Az inerciarendszer komponensei S-ben és E-ben kifejezhetők ortonormális egységvektorokkal:

$$\hat{s}_1 = \frac{{}^E g}{\|{}^E g\|}, \quad \hat{s}_2 = \frac{{}^E g \times {}^E h}{\|{}^E g \times {}^E h\|}, \quad \hat{s}_3 = \hat{s}_1 \times \hat{s}_2. \quad (9)$$

$$\hat{r}_1 = \frac{{}^S A_k}{\|{}^S A_k\|}, \quad \hat{r}_2 = \frac{{}^S A_k \times {}^S H_k}{\|{}^S A_k \times {}^S H_k\|}, \quad \hat{r}_3 = \hat{r}_1 \times \hat{r}_2. \quad (10)$$

A (8)-(10) egyenletek alapján először a mérési és referencia mátrixok alkothatók meg, mely segítségével a rotációs mátrix számítható ki:

$$M_{mea} = [\hat{r}_1 \hat{r}_2 \hat{r}_3], \quad M_{ref} = [\hat{s}_1 \hat{s}_2 \hat{s}_3], \quad {}^S R(q_k) = M_{mea} M_{ref}^T. \quad (11)$$

A fentiekben meghatározott rotációs mátrix ${}^S R(q_k) = (r_{ij})$ lehetővé teszi a magnetométer és gyorsulásérzékelő jelekhez köthető kvaternió megfigyelés kiszámítását:

$$q_0 = \frac{1}{2} \sqrt{1 + r_{11} + r_{22} + r_{33}}, \quad q_1 = \frac{r_{23} - r_{32}}{4q_0}, \quad q_2 = \frac{r_{31} - r_{13}}{4q_0}, \quad q_3 = \frac{r_{12} - r_{21}}{4q_0}. \quad (12)$$

A fent említett algoritmus neve TRIAD (Markley és Crassidis, 2014), (Shuster és Oh, 1981), ami a gyorsulásmérőn és magnetométer méréseken alapuló nyers kvaternió realizációt állítja elő. Ezt az algoritmust egy egyszerű és egyértelmű implementáció jellemzi, és ezért népszerű választás a nyers térbeli helyzet meghatározásához (Wen, Liu, Huang, Su, Guo, Zuo és Qu, 2019), (Roh és Kang, 2018). Azonban hátránya, hogy nagy hibákat ad, amikor dinamikus hatások vagy mágneses zavarások jelentkeznek

2.4. Térbeli orientáció becslése Kiterjesztett Kalman-szűrővel

A (6). és (12). egyenletben leírt MARG szenzor alapú térbeli orientáció megvalósítások egy szenzorfüziós algoritmusban hasznosíthatók, ami egyaránt összegzi az egyes érzékelők egyedi előnyeit és jellemzőit, valamint nagyobb megbízhatósággal és pontossággal nyújt térbeli orientáció eredményeket. Először is, ez a szenzorfüziós algoritmus a giroszkóp-alapú realizációt használja a térbeli orientáció számításában, majd ezeket az eredményeket frissíti a gyorsulásmérőből és magnetométerből kiolvasott legfrissebb kvaternió realizációval. A szenzormodellek fúziója egy EKF-fel van végrehajtva. Az EKF hatékonyan kombinálja a zajos méréseket és a dinamikus modell-alapú predikciókat (Markley és Crassidis, 2014). A szűrő linearizálja a nemlineáris dinamikus modellt az utolsó becsült állapotvektor körül a Jacobi-mátrix segítségével, és a linearizált dinamikához a lineáris Kalman szűrő van felhasználva, ami egy optimális állapotbecslő azáltal, hogy $E[x_k - \hat{x}_k] = 0$ és $E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \rightarrow \inf$.

A MARG érzékelők matematikai modelljei direkt alkalmazhatók a problémában. A folyamatmodell leírja a kvaternió számítást a diszkrét-időben integrált szögsebességgel ((6). egyenlet). Ezért a dinamikus modell egy 7×1 -es állapotvektorral $x_k = (q_k, \omega_k)^T$ van megadva és a 3×1 -es bemeneti vektor $u_k = \Omega_k$ segítségével számítható az új állapotok értéke $x_{k+1} = f(x_k, u_k)$ szerint.

A (12). egyenlet szerint a mérési modell lineáris, hiszen a kvaternió megjelenik az állapotvektorban. Ezért a mérési modellben a 4×1 -es $z_k = q_{k,TRIAD}$ kimeneti vektort használhatjuk, ami biztosítja a kvaternió frissítést:

$$z_k = Hx_k, \quad q_{k,TRIAD} = [I_4 \quad 0_{4 \times 3}] \begin{bmatrix} q_k \\ \omega_k \end{bmatrix}. \quad (13)$$

Az EKF algoritmus szuboptimális állapotbecslést $\hat{x}_k$ ad minimális becslési hiba kovarianciával. Az algoritmus első fázisban az u_k bemeneti változót és az előző becslést $\hat{x}_{k-1}$ használja az ún.

a priori becslés kiszámításához az $f(\hat{x}_{k-1}, u_k)$ függvény kiértékelésével. Az a priori állapotbecslés ($\hat{x}_k^-$) és a hozzá tartozó bizonytalanság (P_k^-) kiszámítása:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k), \quad P_k^- = \phi P_{k-1} \phi^T + Q, \quad \phi = \left. \frac{\partial f}{\partial x} \right|_{\hat{x}_{k-1}}. \quad (14)$$

A (14). egyenletben a Jacobi mátrix ϕ -t alkalmazzuk a priori kovariancia mátrix kiszámításában. Továbbá, a Q kovariancia mátrix a dinamikus modellt leíró állapotok bizonytalanságát jellemzi.

Az algoritmus második fázisa a mérések beépítése. Ekkor a megfigyelési vektort z_k (gyorsulásmérő- és magnetométer-alapú kvaternió), és a mérési zaj kovarianciát R építi be, hogy korrigálja korábbiakban előállított a priori állapotbecslést. Először a Kalman-erősítési mátrixot G_k számítja ki, majd az állapotbecslést $\hat{x}_k$ és annak hiba kovarianciáját (P_k^-) korrigálja. A posteriori becslési eredményeket a következő lépésekben kapjuk meg.

$$\begin{aligned} G_k &= P_k^- H^T (H P_k^- H^T + R)^{-1}, \\ \hat{x}_k &= \hat{x}_k^- + G_k (z_k - H \hat{x}_k^-), \\ P_k &= (I - G_k H) P_k^-. \end{aligned} \quad (15)$$

Az EKF becslési teljesítményét leginkább a Q és R zaj kovariancia mátrixok határozzák meg. Általában a Q és R paramétereket a mérnöki intuíció alapján hangolják; azonban ez a módszer csak kompromisszumos megoldást biztosít a becslési pontosság és a szűrődinamika között (Odry, Fullér, Rudas és Odry, 2018).

2.4.1. Pozíció és orientáció mérése MARG segítségével

A MARG-alapú navigációk célja megbecsülni az orientációt kvaternió reprezentációban, ezenfelül a mobilrobot irányításához szükséges még ismerni a pillanatnyi sebesség és pozíció vektorokat is.

A korábbi fejezetekben bemutatott modelleket és eredményeket felhasználva egy alkalmas kalibrálás lehetővé teszi a gyorsulásmérő- és giroszkóp-alapú kinematikai modell levezetését a pozíció és orientáció számításához:

$$\begin{aligned} p_{k+1} &= p_k + T_s v_k + \frac{T_s^2}{2} ({}^E A_k - {}^E g) \\ v_{k+1} &= v_k + T_s ({}^E A_k - {}^E g) \\ q_{k+1} &= q_k + \frac{T_s}{2} \varrho(q_k) \begin{pmatrix} 0 \\ \Omega_k - \omega_{o,k} \end{pmatrix} \end{aligned} \quad (16)$$

$$\begin{pmatrix} 0 \\ {}^E A_k \end{pmatrix} = q_k \otimes \begin{pmatrix} 0 \\ A_k - a_{o,k} \end{pmatrix} \otimes q_k^*$$

A (16)-ban az ${}^E A_k$ a bias kompenzált gyorsulásmérő méréseket jelöli az E frame-ben, ${}^E g = (0, 0, 9.81)^T$ az SI egységekben kifejezett állandó gravitációs vektor, és a különbségük a robot gyorsulását adja, azaz ${}^E a_k = {}^E A_k - {}^E g$ (Odry, Kecskes, Sarcevic, Vizvari, Toth és Odry, 2020), (Odry, 2020).

A (16) kinematikai egyenlet segítségével következtetni lehet a mozgásban lévő robot pillanatnyi pozíciójára és orientációjára. Azonban nem szabad megfeledkeznünk arról, hogy a korábbiakban említett hibaforrások miatt csak rövid távon tud a fenti modell használható eredményeket szolgáltatni, hiszen a zaj, parazita gyorsulások, zavarások és a szenzorok tökéletlensége jelentős bizonytalanságot indukál a rendszerbe.

A dolgozatban a fenti modellt alkalmazom pillanatnyi pozíció és orientáció mérésre egy kiterjesztett Kalman szűrőben. Tudva azt, hogy a modell csak rövid távon ad megbízható adatokat, a modellel számított értékeket abszolút mérésekkel szándékozom korrigálni. Ehhez rádió technológiákon alapuló pozíciómérést fejleszték és csatolok be az EKF keretrendszerbe. Célom egy olyan fúzió kifejlesztése, mellyel a rádió-alapú mérések jelentősen javíthatók. Ehhez mérőrendszert fejleszték, mérésadatgyűjtést végzek, kalibrálok és algoritmust implementálok. Végül az algoritmus hatékonyságát értékelem ki. Megmutatom, hogy a bemutatott módszerrel jelentősen javul a pozíciómérés pontossága. Az így kifejlesztett algoritmus kimenete javított minőségű jeleket állít elő, így alkalmasabbá teszi azokat a felsőbb irányítások számára, mellyel remélhetőleg jobb szabályozás valósítható meg.

Először a mérőrendszert mutatom be a 3. fejezetben. A 4. fejezetben a mérésadatgyűjtőt részletezem. 5. fejezetben a méréseken megyek végig a magnetométer kalibrálásától a trajektóriák elvégzéséig. A 6. fejezetben megalkotom a szenzorfúziós algoritmust, majd a 7. fejezetben kiértékelem az eredményeket.

3. MÉRŐRENDSZER FELÉPÍTÉSE

A megvalósításhoz négy fő elemre volt szükségem

- egy mérőrendszerre, ami tud szolgáltatni információt a relatív, inerciális mozgásokról a predikcióhoz (EKF mozgásmodelljét meghajtó mérési eredmények),
- egy másik mérőrendszerre, ami az abszolút pozíciót szolgáltatja a korrekcióhoz (EKF megfigyelésmodelljét megvalósító mérési eredmények),

- egy keretrendszerre, mely a mért adatokat a megfelelő struktúrában és a megfelelő mintavételi idővel tárolni tudja, időbélyeggel együtt,
- egy alkalmas algoritmus implementációjára, ami a beérkezett adatokat feldolgozza és a pozíció becslését megvalósítja.

3.1. Az inerciális mozgások meghatározása

A relatív mozgások mintavételezéséhez egy giroszkópra, egy gyorsulásmérőre, valamint egy magnetométerre volt szükségem. Az IMU, amivel a mérőrendszert felvérteztem egy MPU9250 modul. Ez egy 9 szabadságfokú MEMS mozgáskövető eszköz, ami magában foglalja az MPU-6500 6 tengelyes szenzorját, valamint az AK8963 3 tengelyes digitális iránytűjét. A MEMS (Microelectromechanical Systems - Mikro elektro-mechanikai rendszer) technológiának köszönhetően rendkívül kicsi tokozásban lévő mikromechanikai eszközök. A giroszkóp mechanikai szerkezete alig 100-200 μm méretű és a szerkezet elmozdulásaiból számolható a megfelelő szögsebesség. Az MPU-9250 tokozásának befoglaló méretei 3 mm x 3 mm x 1 mm. Az MPU-6500-as modul tartalmaz egy giroszkópot, ami $\pm 250^\circ/\text{s}$ -tól egészen $\pm 2000^\circ/\text{s}$ tartományban tudja szolgáltatni a szögsebesség adatokat, valamint egy gyorsulásmérőt, aminek mérési tartománya $\pm 2g$ -tól $\pm 16g$ -ig terjed ($g=9,81 \text{ m/s}^2$). Az AK8963 tartalmaz egy magnetométert, ami $\pm 4800 \mu\text{T}$ mérési tartománnyal rendelkezik. A magnetométerrel az IMU egy belső I2C protokollon keresztül kommunikál, az adatgyűjtő felé pedig I2C-n vagy SPI kommunikáción keresztül tudja eljuttatni a megfelelő adatokat. Az SPI protokollal nagyobb sebesség érhető el, így azt választottam az adatok továbbítására (MPU-9250 Product Specification, 2016). Az adatok feldolgozására egy ESP32-es vezérlőt használtam.

3.2.A ground truth kinyerése

A ground truth értékek meghatározása a pose becselő kutatások egyik legnagyobb kihívása. A trajektóriák megvalósításához egy Universal Robots UR5 kooperatív robot került kiválasztásra, ami több szempontból is előnyös volt a mérés megvalósításában.

- Egyszerű kezelői interfész lehetővé teszi a trajektóriák definiálását, akár a teach pendant-ról irányítva, vagy a mozgások megírásával a különböző keretrendszerekben.
- Mivel kooperatív robot, így az emberrel való munkát nagyon meg tudja könnyíteni, valamint a balesetveszélyt minimalizálja. Intuitív programozása, könnyű illeszthetősége a mérőrendszerhez ideálissá tette a vele való munkát.

- Driverekkel az adatgyűjtő rendszerekhez illeszthető, melyen keresztül az aktuális csuklókoordináták lekérhetők. Ezeket a direkt geometria alkalmazásával kiegészítve egyszerűen TCP koordinátává, valamint IMU koordinátává alakíthatók rövid programrészek implementálásával.

Az UR5 a mérőrendszer része, ennek segítségével validálható (a ground truth felhasználásával) az UWB modulokból származtatható abszolút koordináták, továbbá a későbbiekben bemutatásra kerülő szenzorfüzió teljesítménye.

3.3. Az abszolút pozíció meghatározása

Az UWB a wifihez és a Bluetooth-hoz hasonló, rövid hatótávolságú vezeték nélküli protokoll. A többi technológiától abban különbözik, hogy nagyon magas frekvencián, több GHz szélességű frekvenciaspektrumot használ. Az UWB adó broadcast impulzusokat küld szét széles frekvenciatartományban, a megfelelő vevő pedig adatokká alakítja az impulzusokat, ha az adó által küldött ismerős impulzus sorozatokat észlel. Az impulzusokat két nanoszekundumonként küldi az adó, így ezzel az UWB valós helymeghatározást tesz lehetővé (Wang, Waadt, Burnic, Xu, Kocks, Bruck és Jung, 2010). Az energiafogyasztása rendkívül kicsi, a sáv szélesség viszont nagy, így rengeteg adat továbbítására alkalmas. A helymeghatározás/távolságmérés úgynevezett Time of Flight (ToF) mérésekkel történik, azaz az eszközök közötti jelcsomagok utazási idejéből történik a távolság számítása.

Több gyártó is fejleszt UWB kommunikációt, erre egy külön konzorcium a FiRa (Fine Ranging, pontos helymeghatározás) is épült, ami az UWB-ökoszisztéma kiépítésén dolgozik (ez a meglévő IEEE 802.15.4/4x vezeték nélküli kommunikációs szabványra épül). A FiRa konzorcium szerint két egymásra rálátó eszköz relatív pozícióját képes az UWB meghatározni 200 méteres távolságon belül (Wang, Waadt, Burnic, Xu, Kocks, Bruck és Jung, 2010). Egy általános helymeghatározási módszer a fix helyekre elhelyezett úgynevezett UWB-horgonyokhoz (anchor-ok) képest mozgó UWB vevő helymeghatározása. A megvalósításhoz kialakított mérőrendszerben én is ezt a metodikát használtam. Kijelöltem egy területet, melynek a 4 sarkán elhelyeztem 4 anchor-t. A vevő (tag) az UR5 effektorán helyezkedik el és a 4 távolság kiolvasásából becsüli a pontos pozíciót. Az UWB-re a Makerfabs ESP32-es vezérlővel ellátott UWB modulját használtam a 4 anchor-nak és a tag-nek egyaránt.

3.4. Az effektor felépítése

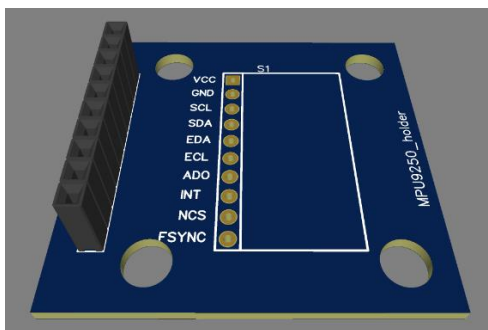
A kiválasztott eszközöket (IMU, ESP32 vezérlő, UWB) el kellett helyezni az effektoron. A célom egy univerzális mérőegység kifejlesztése volt, melyen többféle vezérlő, és több különböző szenzor is csatlakoztatható, a későbbi fejlesztések tükrében.

A fentiekben tárgyalt szenzorokon kívül még a következő eszközökre lett felkészítve az effektor:

- ADNS3080 Optical Flow Sensor (Optikai elsodródás érzékelő). Az eszköz a pillanatnyi kameraképet elemzi, és differenciáláson keresztül és a kamerakép az elsodródásának sebességét szolgáltatja X és Y irányban. Ezek a mérések a síkbeli mozgást jellemzik, és integrálása későbbi kutatások során valósul meg.
- SR04 Ultrahangos (UH) távolságmérő szenzor. A szenzor ultrahang csomagok alkalmazásával méri Time of Flight (ToF) módszerrel az ultrahang impulzus közegben töltött idejét. A hang terjedési sebességéből számítható a távolság. Az orientáció és az UH adatokból Z irányú abszolút pozícióra lehet következtetni, melynek becsatolása későbbi kutatások során valósul meg.

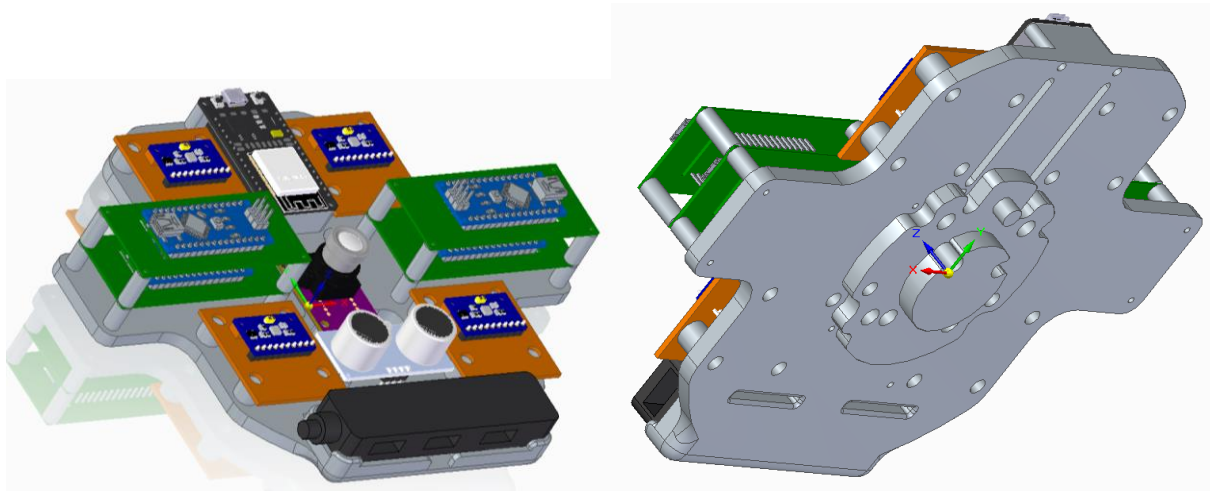
A fentiek mellett felkerült az eszközre az aktuális MPU-9250 szenzor mellé még 3 db MPU-9250 a későbbi javított szenzorfüzió megvalósítása, valamint a szenzorok hibáinak javított kiküszöbölése érdekében. Egy USB HUB végzi a soros vonali adatok összegyűjtését.

Az MPU-9250-hez készítettem egy adapter nyomtatott áramkört (3. ábra) a könnyebb rögzíthetőség érdekében, hogy az effektoron a megfelelő pozícióban és orientációban tudjam rögzíteni a modult, valamint a bekötését segítő tűkesort is elhelyeztem az áramkörön.

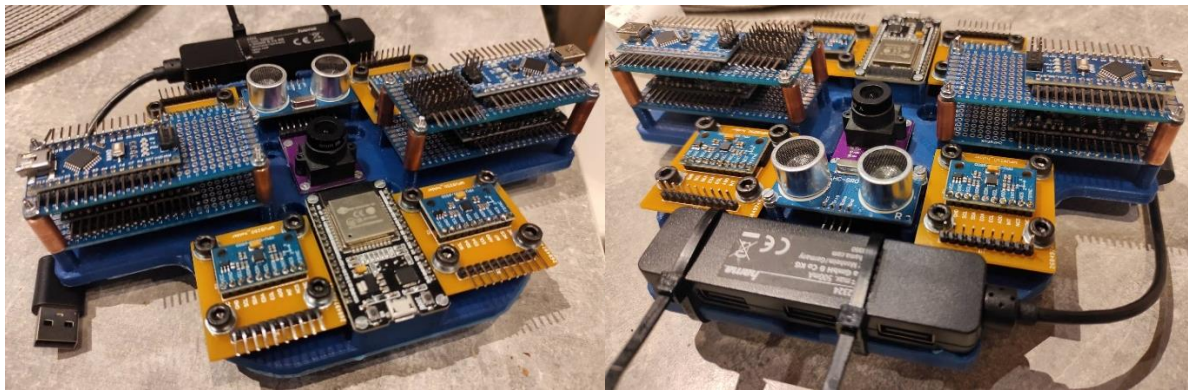


3. ábra - Az MPU-9250-hez készített adapter lap

Az effektort (mely a szenzorszigetet tartalmazza) az Invertor programban terveztem meg, majd 3D nyomtatóval legyártásra került, modellezve a vezérlőket és a felhelyezésre kerülő modulokat (lásd 4-5. ábra).



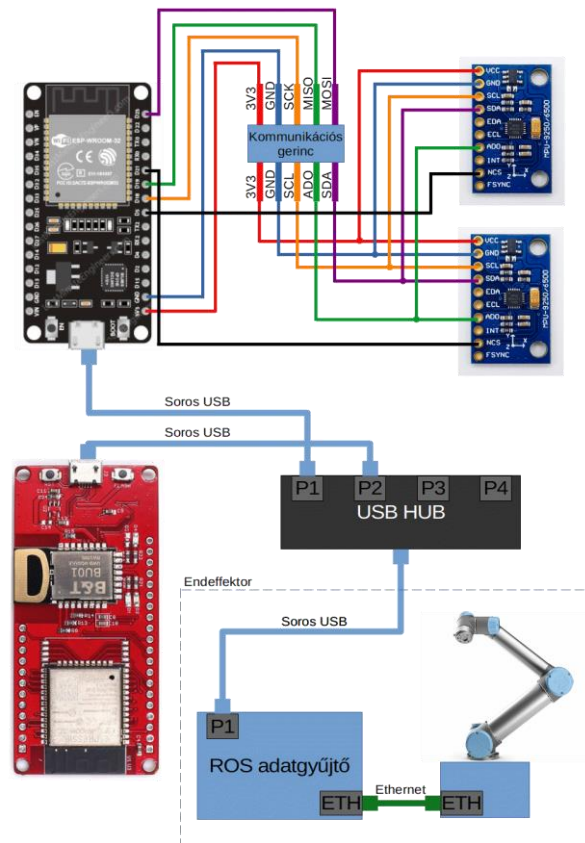
4. ábra - Az endeffektor 3D modellje



5. ábra - Az elkészült endeffektor

A rendszer struktúrája a következőképpen áll össze (6. ábra):

- Az IMU-k egy kommunikációs gerincre (Bus-ra) csatlakoznak, a későbbi fejlesztési lehetőségek érdekében; A chip választó (Chip Select – CS) pont pedig közvetlen kerül bekötésre az ESP32 modulba.
- Az ESP32, valamint az UWB modul soros kommunikációi egy USB HUB-ra csatlakoznak.
- Az USB HUB szintén soros USB vonalon csatlakozik az adatgyűjtő számítógéphez.
- Az adatgyűjtő gép réz-ethernet kommunikációval csatlakozik az UR5-ös egységhez.



6. ábra - A mérőrendszer struktúrája

4. MÉRÉSADATGYŰJTŐ KERETRENDSZER

Az effektor hardverének fizikai összeállítását a mérőrendszer keretrendszerének felépítése követte. A szoftveres környezetnek a Robot Operációs Rendszert (Robot Operating System, ROS) választottam. Ez egy robotikában széles körben alkalmazott nyílt forráskódú rendszer. Az előnye, hogy rengeteg beépített funkciót és csomagok formájában elérhető támogatást tartalmaz, ilyenek:

- különböző modellezett robotkörnyezetek, robot modellek, és drónok támogatása
- algoritmusok arcfelismeréshez, mozgáskövetéshez, sztereó látáshoz, útvonaltervezéshez (path planning) és a szimultán lokalizáció és térképezés (Simultaneous localization and mapping, SLAM) megvalósításához

Én a ROS melodic-ot választottam a feladat kivitelezésre. Ehhez szükség volt egy Ubuntu 18.04-re, amit külön a mérőrendszernek dedikáltan készítettem el.

4.2. ROS alapfogalmak

Node: Futtatható ROS alkalmazás, mely képes más ROS alkalmazásokkal (node-okkal, komponensekkel) való kommunikációra, tud publikálni és olvasni a topic-ból és adott néven él

a rendszerben. Az én esetemben az ESP32 vezérlőkre írtam meg a Node-okat Arduino IDE környezetben.

Message: A Node-ok üzenetek segítségével kommunikálnak, ez az üzenet tartalmazza a küldendő információt. Ezek az üzenetek kerülnek továbbításra a topic-okon keresztül. A ROS különböző üzenet típusokat tartalmaz a különböző alkalmazásokhoz.

Topic: A Node-ok ide publikálják az üzeneteiket, valamint a Node-ok ide tudnak feliratkozni, hogy más Node-ok üzeneteit fogadni tudják. A Topic tartalmazza az üzenetek struktúráját.

Roscore: A központi mag, elindítása után a Master (ROS szervíz szolgáltatása), a Rosout (ROS kimenetei), valamint a ROS paraméter szervere indul el, amik ezek keresztül a ROS működését biztosítják.

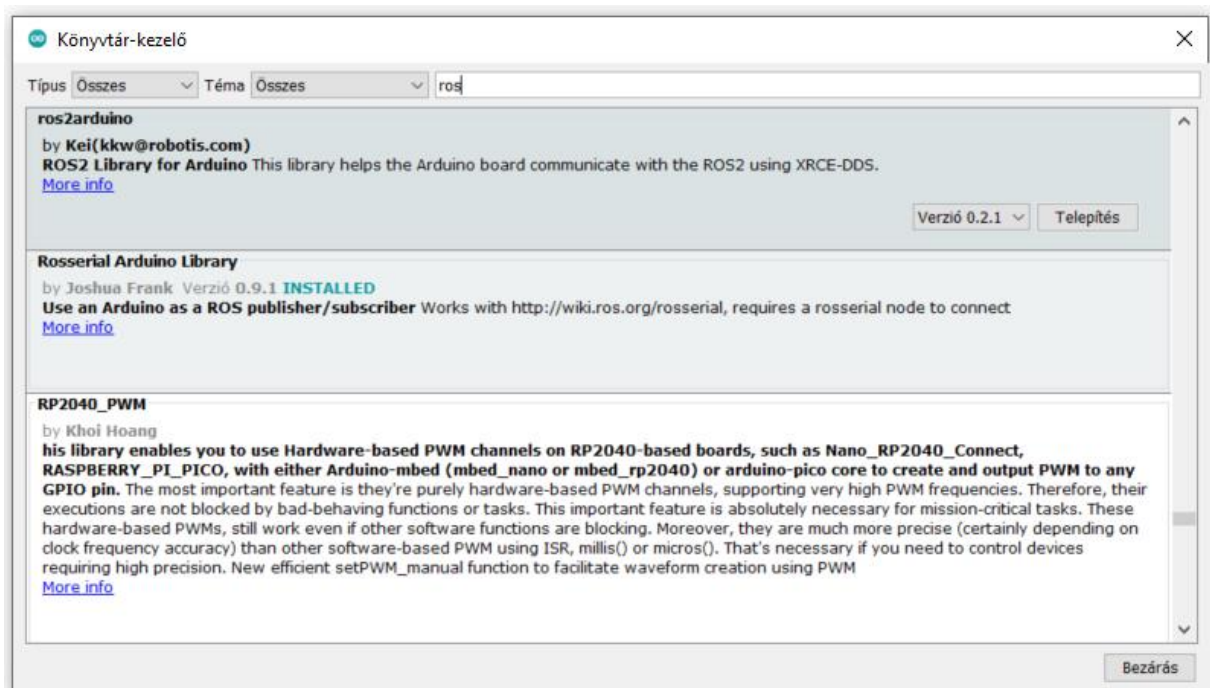
4.3. A ROS Node-ok megírása

A következő feladat a ROS Node-ok megírása volt a különböző ESP32-es vezérlőkre. A mérőrendszer első ütemét úgy állítottam fel, hogy 1 darab IMU-t (MPU-9250) olvasok 100 Hz-es mintavétellel (szögsebesség adatok, gyorsulás adatok és magnetométer adatok), valamint 4 db fix ponton elhelyezett UWB anchor jeleit regisztrálom az abszolút pozícióhoz. Az ötödik UWB modul az effektoron helyezkedik el, mint Tag, ezen lett megírva a ROS Node-is, mely négy mérést publikálja.

A ROS keretrendszer sok beépített szenzorstruktúrát tartalmaz, ahol tudtam ott ezeket a struktúrákat alkalmaztam. A Node-okat Arduino IDE környezetben írtam meg, viszont szükség volt a keretrendszerbe egy Drivert beépítésére, hogy az Arduino IDE-ben megírt ESP32 kódok által publikált topikokat a ROS fogadni tudja. Ehhez a következő kódrészletek kellettek:

```
sudo apt-get install ros-melodic-rosserial-arduino
sudo apt-get install ros-melodic-rosserial
```

Minden driver frissítés/telepítés után szükséges a workspace-t is frissíteni. A ROS így már képes volt fogadni a virtuális soros vonalon (rosserial csomag alkalmazásával) az Arduino-ban megírt és publikált topic-okat. A következő pont az Arduino IDE frissítése volt, ehhez a Rosserial Arduino Library-t telepíteni kellett (7. ábra).



7. ábra - A Rosserial könyvtár beillesztése az Arduino IDE fejlesztői környezetbe

4.3.1. Az IMU Node implementálása

Az MPU9250-et (pontosabban az ebben foglalt MPU-6500-as modult) SPI protokollon keresztül olvasom ki. Az MPU-6500-as belső I2C vonalon kommunikál a magnetométerrel (AK8963 modul), valamint továbbadja az SPI vonalon ennek a szenzornak a méréseit. A koncepció az volt, hogy minden adatból nyers (szűz mérési adat, raw adat) adatot regisztrálok, és a későbbi adatfeldolgozások során fogom a kalibrálásokat elvégezni. A kódot univerzálisra készítettem el így az SPI vonalon több IMU is kiolvasható a későbbi fejlesztési igényeket megalapozva. Az első lépés az IMU kiolvasása volt. Ehhez a következő kódrészletek voltak szükségesek:

Az IMU kiolvasásához szükséges header file-ok és az SPI vonal deklarálásához szükséges adatok a következők.

```
#include <SPI.h>
#include <MPU9250.h>

#define SPI_CLOCK 8000000 // 8MHz clock works.
#define SS1_PIN 5
#define SS2_PIN 21

MPU9250 mpu1(SPI_CLOCK, SS1_PIN);
MPU9250 mpu2(SPI_CLOCK, SS2_PIN);
```

A Setup rutin fontosabb összetevői, az IMU-k indítása, alapbeállítása:

```
SPI.begin();
mpu1.init(true);
mpu2.init(true);
```

A Loop rutin összetevői:

```
mpu1.read_all();
```

4.3.2. Az IMU kiolvasása

A fentiek után egy adatstruktúra állt rendelkezésre a következő adatokkal.

- mpu1.gyro_data[3]: háromelemű tömb a szögsebesség adatokkal a X, Y és Z irányoknak megfelelően
- mpu1.accel_data[3]: háromelemű tömb a gyorsulás adatokkal a X, Y és Z irányoknak megfelelően
- mpu1.mag_data[3]: háromelemű tömb a magnetométer adatokkal a X, Y és Z irányoknak megfelelően

4.3.3. ROS Node beépítése, és a publikálás

Az összetevők deklarálását a következők ismertetik.

```
#include <sensor_msgs/Imu.h>
#include <sensor_msgs/MagneticField.h>
ros::NodeHandle nh;
sensor_msgs::Imu imu1_msg;
sensor_msgs::MagneticField magneticfield1_msg;
ros::Publisher pub_imu1("imu1", &imu1_msg);
ros::Publisher pub_magneticfield1("magneticfield1",
&magneticfield1_msg);
```

Az *IMU.h* és a *MagneticField.h* a különböző szenzorok struktúráihoz tartozó ROS könyvtárreszek. A *NodeHandle* inicializálja a ROS Node-ot, a *sensor_msgs* struktúrát felhasználva inicializálásra kerülnek az IMU, valamint a magnetométer ROS topikokban használt üzenetek. A *ros::Publisher* parancsokkal a publikáláshoz szükséges osztályok definiálása valósul meg.

A Setup Rutin elemei, a Baud rate beállítása:

```
nh.getHardware()->setBaud(500000);
nh.initNode();
nh.advertise(pub_imu1);
```

```
nh.advertise(pub_magneticfield1);
imu1_msg.header.frame_id = "imu1_frame";
magneticfield1_msg.header.frame_id = "magneticfield1_frame";
```

A *setbaud* paranccsal a ROS soros vonal sebességét (bit/s) változtatom. Erre azért volt szükség, mert a gyári érték (57600 bit/s) lassúnak bizonyult. A ROS struktúrák nagy adatsomagok és annak érdekében, hogy tudjak tartani egy megfelelő mintavételi sebességet (EKF tekintetében a relatív pozíció mintavételi sebessége kardinális szempont) növelnem kellett az adatátviteli sebességet. A ROS Serial maximális támogatott sebessége 500 000 bit/s. Ezzel elértem azt, hogy 100Hz-es mintavételi frekvenciát tudok tartani az IMU adatok ROS felé történő továbbításában (giroszkóp, gyorsulásmérő, és magnetométer együttes adatai).

Az *nh.initNode* paranccsal inicializáltam a node-ot, az *nh.advertise* paranccsal pedig a publikálásokat. Az üzenetek header-jeiben elneveztem a frame-eket (ezek az *imu1_frame*, és *magneticfield1_frame* voltak).

A Loop rutin elemei:

```
if (millis() > publisher_timer) {
    publisher_timer = millis () + 5;
    mpul.read_all();
    imu1_msg.angular_velocity.x = mpul.gyro_data[0];
    imu1_msg.angular_velocity.y = mpul.gyro_data[1];
    imu1_msg.angular_velocity.z = mpul.gyro_data[2];
    imu1_msg.linear_acceleration.x = mpul.accel_data[0];
    imu1_msg.linear_acceleration.y = mpul.accel_data[1];
    imu1_msg.linear_acceleration.z = mpul.accel_data[2];
    imu1_msg.header.stamp = nh.now();
    magneticfield1_msg.magnetic_field.x = mpul.mag_data[0];
    magneticfield1_msg.magnetic_field.y = mpul.mag_data[1];
    magneticfield1_msg.magnetic_field.z = mpul.mag_data[2];
    magneticfield1_msg.header.stamp = nh.now();
    pub_imu1.publish(&imu1_msg);
    pub_magneticfield1.publish(&magneticfield1_msg);
}
```

Végül a topikokat feltöltöm a beérkező nyers adatokkal és publikálom őket, viszont a publikálás előtt az időbélyeg paramétereknél alkalmazom az *nh.now()* parancsot. A későbbi adatok feldolgozásánál ez nagyon fontos szempont, hiszen az időbélyegek segítségével lehet a mérési eredményeket időrendi sorrendben feldolgozni.

4.3.4. A Node tesztelése

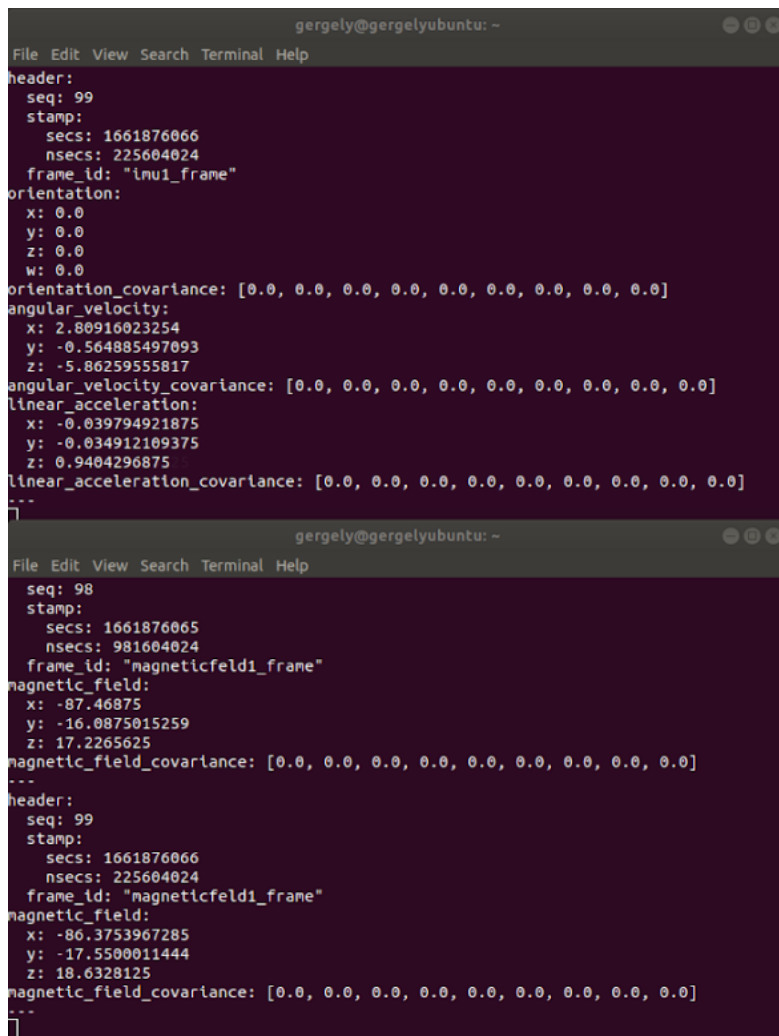
Az adatgyűjtő számítógépen először el kell indítani a ROS keretrendszert a következő paranccsal: *roscore*. A ROS topikokhoz tartozó fontosabb parancsok:

`roslaunch roscpp_serial_node serial_node.py /dev/ttyUSB0 _baud:=500000` - A parancs a ttyUSB0 porton nyitja meg a roscpp vonalat 500000-es baudrate-el.

`rostopic list` - kilistázza az aktuális topikokat, melyekre fel lehet iratkozni.

`rostopic echo` - feliratkozik a topikra és kiírja a terminálablakba publikálásonként az üzeneteket.

Az IMU-tól érkező adatokról egy képernyőképet az alábbi ábra mutatja (8. ábra):



```

gergely@gergelyubuntu: ~
File Edit View Search Terminal Help
header:
  seq: 99
  stamp:
    secs: 1661876066
    nsecs: 225604024
  frame_id: "imu1_frame"
orientation:
  x: 0.0
  y: 0.0
  z: 0.0
  w: 0.0
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 2.80916023254
  y: -0.564885497093
  z: -5.86259555817
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: -0.039794921875
  y: -0.034912109375
  z: 0.9404296875
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
gergely@gergelyubuntu: ~
File Edit View Search Terminal Help
seq: 98
stamp:
  secs: 1661876065
  nsecs: 981604024
frame_id: "magneticfeld1_frame"
magnetic_field:
  x: -87.46875
  y: -16.0875015259
  z: 17.2265625
magnetic_field_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
header:
  seq: 99
  stamp:
    secs: 1661876066
    nsecs: 225604024
  frame_id: "magneticfeld1_frame"
magnetic_field:
  x: -86.3753967285
  y: -17.5500011444
  z: 18.6328125
magnetic_field_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---

```

8. ábra - A Rosserial-on keresztül érkező IMU adatok

Az 1-es index a topikoknál a későbbi fejlesztés esetén a több IMU-val működő szenzorfúzió lehetőségeire utal.

4.3.5. Az UWB modul programjának megírása, és a node kifejlesztése

A rendszerben a 4 db anchort terveztem elhelyezni melyeket egy tag olvassa az effektoron elhelyezve. Ezekből az értékekből lehet a későbbiekben a neuronhálók segítségével egy leképezést kifejleszteni a 4 távolság, valamint az effektor pozíciója vonatkoztatásában.

4.3.6. Az "Anchor" modulok implementálása

Az Anchor kódja az összetevők deklarálásával indul:

```

#include <SPI.h>
#include "DW1000Ranging.h"

```

```

#define ANCHOR_ADD "03:00:5B:D5:A9:9A:E2:01"

#define SPI_SCK 18
#define SPI_MISO 19
#define SPI_MOSI 23
#define DW_CS 4

const uint8_t PIN_RST = 27;
const uint8_t PIN_IRQ = 34;
const uint8_t PIN_SS = 4;

```

Deklarálásra került az SPI kommunikációs vonal, ami az UWB modul (DW1000) és az ESP32 vezérlő között teremt kapcsolatot. Ezen kívül a legfontosabb elem az anchor címe. Ebből a címből fogom meghatározni a tag-ben, hogy melyik anchortól származik az üzenet. A címek struktúrája a következőképpen alakult:

```

Anchor-1: "01:00:5B:D5:A9:9A:E2:01"
Anchor-2: "02:00:5B:D5:A9:9A:E2:01"
Anchor-3: "03:00:5B:D5:A9:9A:E2:01"
Anchor-4: "04:00:5B:D5:A9:9A:E2:01"

```

A setup rutin elemeit a következő sorok ismertetik:

```

SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
DW1000Ranging.initCommunication(PIN_RST, PIN_SS, PIN_IRQ);
DW1000Ranging.attachNewRange(newRange);
DW1000Ranging.attachBlinkDevice(newBlink);
DW1000Ranging.attachInactiveDevice(inactiveDevice);

DW1000Ranging.startAsAnchor(ANCHOR_ADD,
DW1000.MODE_LONGDATA_RANGE_LOWPOWER, false);

```

A fenti programrész inicializálja a SPI vonalat, majd elindítja a vezérlőt anchor módban.

```
DW1000Ranging.loop();
```

A fenti programrész valósítja meg a távolságokat, melyeket a tag fog kiolvasni.

4.3.7.A "Tag" modul implementálása

A következő a tag megvalósítása, valamint a tag adatainak integrálása egy ROS Node-ba. Első lépés az UWB adatok kiolvasása a 4 anchor-ból. Ehhez az összetevőket deklaráltam a következő sorokkal.

```

#include <SPI.h>
#include "DW1000Ranging.h"

#define SPI_SCK 18
#define SPI_MISO 19

```

```

#define SPI_MOSI 23
#define DW_CS 4
#define ANCHOR1_ADD "86:17:5B:D5:A9:9A:E2:99"
const uint8_t PIN_RST = 27;
const uint8_t PIN_IRQ = 34;
const uint8_t PIN_SS = 4;
float d_array[12] = {0,0,0,0,0,0,0,0,0,0,0,0};
int address =0;
boolean d_puffer[4] = {0,0,0,0};

```

A deklarálásnál szintén az SPI vonal meghatározása volt az első feladat. Ezt követte a cím adása az anchor-nak. A következő pont egy float tömb deklarálása volt, melybe az anchor-okból kiolvasott adatok kerülnek be. Az *address* segédváltozót és *d_puffer* 4 elemű tömböt is itt deklaráltam, melyek a későbbi tömb írásában fognak szerepet vállalni. A setup rutin elemeit a következő sorok ismertetik:

```

SPI.begin(SPI_SCK, SPI_MISO, SPI_MOSI);
DW1000Ranging.initCommunication(PIN_RST, PIN_SS, PIN_IRQ);
DW1000Ranging.attachNewRange(newRange);
DW1000Ranging.attachNewDevice(newDevice);
DW1000Ranging.attachInactiveDevice(inactiveDevice);
DW1000Ranging.startAsTag(ANCHOR1_ADD,
DW1000.MODE_LONGDATA_RANGE_LOWPOWER);

```

A fenti programrészletben az SPI kommunikáció inicializálása és indítása történik. Ezek után a DW1000 modullal való kommunikáció épül fel, a modul inicializálása, majd a modul elindítása tag üzemmódban. Az anchorok kiolvasását a következő programrészben oldottam meg:

```

void newRange() {
address=(DW1000Ranging.getDistantDevice()->getShortAddress());
d_array[(address-1)*3]=(DW1000Ranging.getDistantDevice()-
>getShortAddress());
d_array[(address-1)*3+1]=(DW1000Ranging.getDistantDevice()-
>getRange());
d_array[(address-1)*3+2]=(DW1000Ranging.getDistantDevice()-
>getRXPower());
d_puffer[address]=1;
}

```

A fenti programrészletben először kiolvasom az eszköz rövid címét. A rövid cím a cím első négy karakteréből áll össze a következő módon:

```
cím: ab:cd:ef:gh:ij:kl:mn:op
rövid cím: cdab
```

Az anchorok rövid címei a következőképpen alakultak:

```
Anchor-1 cím: "01:00:5B:D5:A9:9A:E2:01", Anchor-1 rövid cím: 0001
Anchor-2 cím: "02:00:5B:D5:A9:9A:E2:01", Anchor-1 rövid cím: 0002
Anchor-3 cím: "03:00:5B:D5:A9:9A:E2:01", Anchor-1 rövid cím: 0003
Anchor-4 cím: "04:00:5B:D5:A9:9A:E2:01", Anchor-1 rövid cím: 0004
```

A fenti címeket felhasználva kerülnek be a *d_array* tömbbe az értékek. Három érték kerül kiolvasásra. A rövid cím, a távolság, ami az UWB-ből kerül kiolvasásra, valamint az RSSI érték. Az RSSI a fogadott jel erőssége. Ezt az értékét később szintén feldolgozom a távolságértékek meghatározásához. A kiolvasott tömb struktúrája a következő:

```
- d_array[0]: Anchor-1 rövid cím
- d_array[1]: Anchor-1 távolság méterben
- d_array[2]: Anchor-1 RSSI
- d_array[3]: Anchor-2 rövid cím
- d_array[4]: Anchor-2 távolság méterben
- d_array[5]: Anchor-2 RSSI
- d_array[6]: Anchor-3 rövid cím
- d_array[7]: Anchor-3 távolság méterben
- d_array[8]: Anchor-3 RSSI
- d_array[9]: Anchor-4 rövid cím
- d_array[10]: Anchor-4 távolság méterben
- d_array[11]: Anchor-4 RSSI
```

Ezek után felépítettem a ROS node-ot. Mivel ez egy speciális alkalmazás így nincs rá előre megírt ROS struktúra. A ROS képes Float32 tömböt is küldeni node-ok között, de ezzel az volt a probléma, hogy az időbélyeget nehezen lehetett belehelyezni. Másik megoldás egy saját üzenet struktúra implementálása lett volna. Azonban effektívebbnek bizonyult az, hogy egy IMU struktúrát definiáltam, melybe belehelyeztem az összes értéket az időbélyeggel együtt. Az IMU struktúra két 3 elemű tömb. Ez még nem volt elég így felhasználtam a kovariancia értékek helyeit is. A cél az volt, hogy az időbélyeggel megfelelően kerüljön az adatgyűjtőbe az összes adat.

```
sensor_msgs::Imu uwb_msg;
ros::Publisher pub_uwb("uwb", &uwb_msg);
```

A ROS struktúra kiépítése az IMU szenzorral megegyezően történt, annyi változtatással, hogy az topik elnevezése uwb lett. A topik megvalósítása a következőképpen alakult:

```

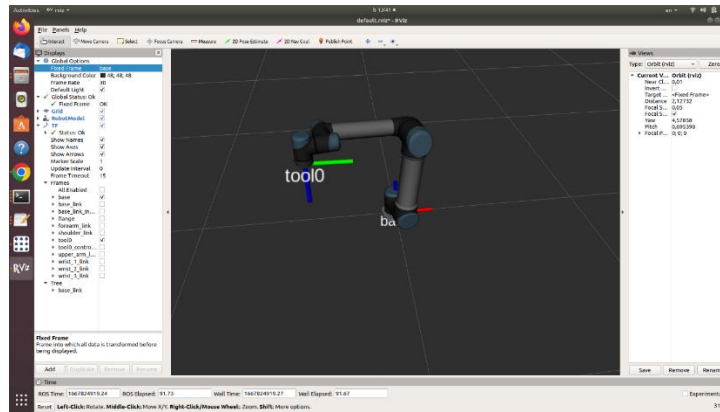
if (millis() > publisher_timer) {
publisher_timer = millis () + 1;
DW1000Ranging.loop();
if(d_puffer[1]and d_puffer[2]and d_puffer[3]and d_puffer[4]){
uwb_msg.angular_velocity.x = d_array[0];
uwb_msg.angular_velocity.y = d_array[1];
uwb_msg.angular_velocity.z = d_array[2];
uwb_msg.linear_acceleration.x = d_array[3];
uwb_msg.linear_acceleration.y = d_array[4];
uwb_msg.linear_acceleration.z = d_array[5];
uwb_msg.angular_velocity_covariance[0] = d_array[6];
uwb_msg.angular_velocity_covariance[1] = d_array[7];
uwb_msg.angular_velocity_covariance[2] = d_array[8];
uwb_msg.linear_acceleration_covariance[0] = d_array[9];
uwb_msg.linear_acceleration_covariance[1] = d_array[10];
uwb_msg.linear_acceleration_covariance[2] = d_array[11];
uwb_msg.header.stamp = nh.now();
pub_uwb.publish(&uwb_msg);
d_puffer[1]=0;
d_puffer[2]=0;
d_puffer[3]=0;
d_puffer[4]=0;
}

```

Az értékek regisztrációja akkor valósul meg, ha megérkezik az összes mérési eredmény. Ezt a korábbiakban definiált *d_puffer* tömb biztosítja. Az mérések megérkezése után beírásra kerülnek a struktúra megfelelő helyeire, beírásra kerül az időbélyeg, publikálásra kerül az üzenet, valamint a *d_puffer* értékek kinullázódnak.

4.4. Az UR driver és a koordináták meghatározása

A node-ok kifejlesztését az UR-5 driver hozzáillesztése követte a ROS keretrendszerhez. A driverrel feltelepítettem és frissítettem az RViZ-be beépülő modult, mely a robot megjelenítését segítette.

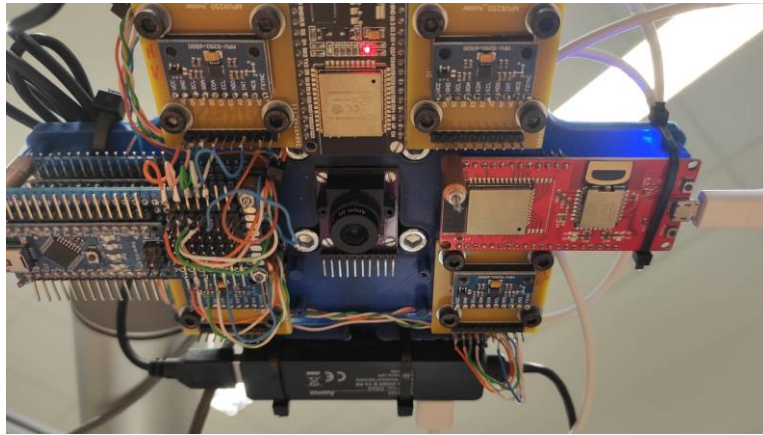


9. ábra - Az RViz keretrendszer és a UR-5 robotkar modellje

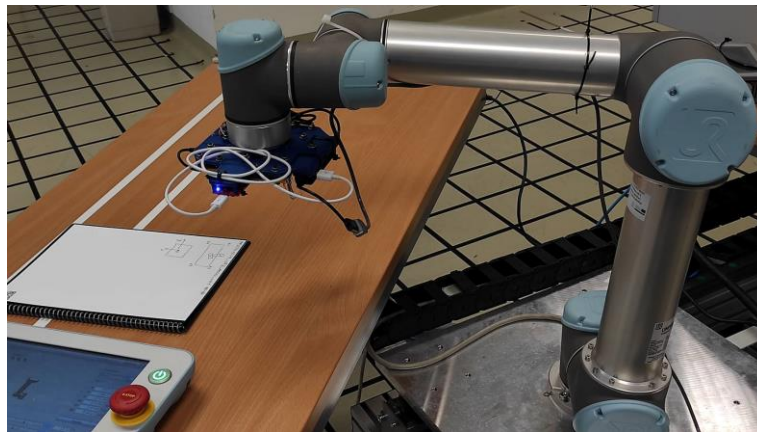
Az UR teach pendant-ján keresztül telepítettem egy externalcontrol-1.0.5.urcap programot a robotra. Az URcap fájlt egy Pendrive-on keresztül raktam a teach-pendant-re. Ezen driveren keresztül kommunikált a ROS driverrel a robot. Létesítettem egy vezetékes Ethernet kapcsolatot az UR és a mérésadatgyűjtő ROS számítógép között és így megjelentek az UR csuklókoordináták topic-jai. Egy további node-ra megírására volt még szükség, mely kiszámolta az effektor koordinátáit a bázis koordinátarendszerhez képest a csuklókoordinátákból. Az end effektor póz meghatározását a MoveIt pályatervezőhöz tartozó osztály metódusai segítségével határoztam meg (Chitta, 2016). A kifejlesztett node legfontosabb programrészét a következő sorok ismertetik:

```
moveit::planning_interface::MoveGroupInterface
move_group_interface(PLANNING_GROUP);
geometry_msgs::PoseStamped ee_link_pose;
ros::Publisher eelink_pub =
n.advertise<geometry_msgs::PoseStamped>("ee_link_pose", 1000);
ros::Rate loop_rate(20);
while (ros::ok()) {
    ee_link_pose = move_group_interface.getCurrentPose();
    eelink_pub.publish(ee_link_pose);
    ros::spinOnce();
    loop_rate.sleep();
}
```

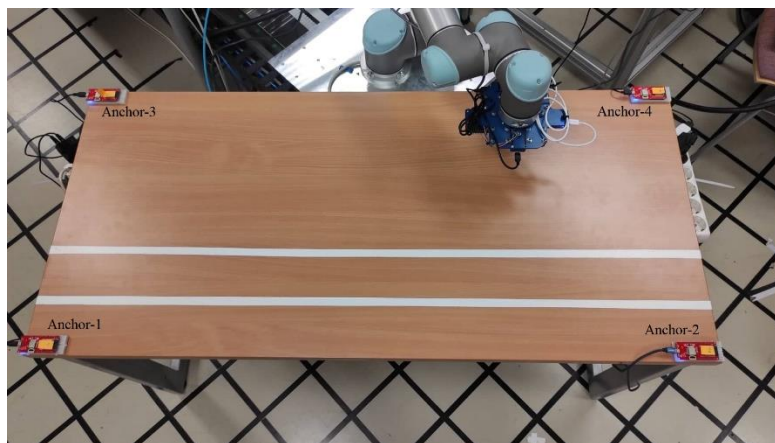
A fentiekben először a pályatervezésért felelős interfészt definiáljuk, ezután létrehozunk egy pózok publikálására alkalmas `ee_link_pose` változót és a hozzá tartozó publisher-t (`eelink_pub`). A ground truth adatok publikálását 50Hz-re állítottam, ezt a `loop_rate` biztosítja. A node folyamatosan publikálja az effektor pózt, a `getCurrentPose` metódus meghívásával.



10. ábra - A telepített endeffektor



11. ábra - Az elkészült mérőrendszer



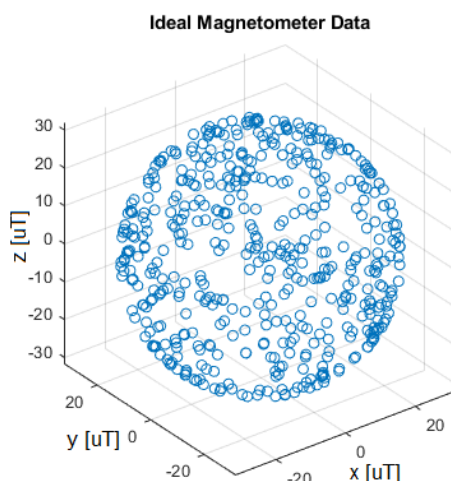
12. ábra - A mérőrendszer és a négy anchor modul

5. MÉRÉSEK

A kialakított mérőrendszer összeállítása után a feladat megvalósításához szükséges méréseket három szakaszra osztottam, ezek: magnetométer kalibráció, fingerprintek felvétele, és trajektóriák felvétele.

5.1. Magnetométer kalibráció

Ideális esetben, ha a magnetométerünk tökéletesen működik és csak a Föld mágneses tere hat rá, akkor, ha minden lehetséges irányba körbe mozgatjuk a magnetométert, és az X, Y valamint a Z tengelyekhez kapcsolódó értékeket ábrázoljuk egy koordináta-rendszerben, akkor a pontoknak egy gömbre kell felfeküdniük (Bonnet, Bassompierre, Godin, Lesecq és Barraud, 2009), melynek sugara a Föld mágneses térerőssége.

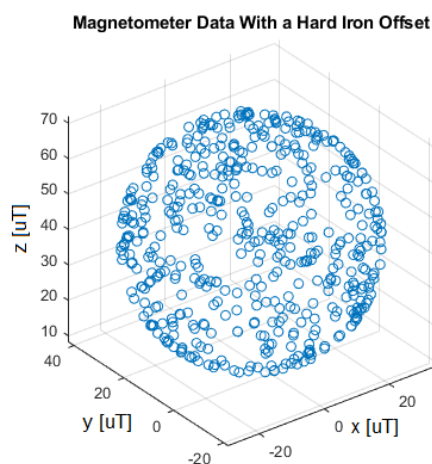


13. ábra - Az ideális magnetométer adatok ábrázolása 3D-ben

Mivel a szenzorunkat több más tényező is befolyásolja (zajforrások, gyártási hibák és külső zavarások), ezért a fenti kép eltorzul. A különböző befolyásoló hatásokat a következő pontokat tárgyalom.

5.1.1. Keményvas hatás

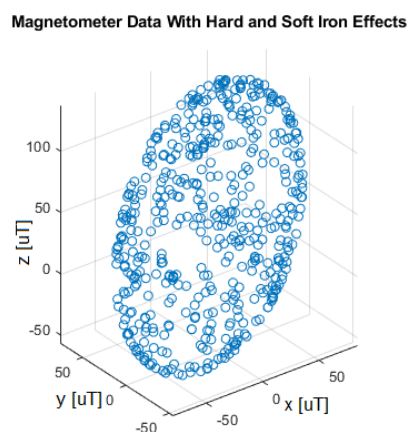
A hibák közül a legszembetűnőbbek a keményvas hatások. Ezek helyhez kötött, stacionárius zajforrások, melyek legtöbbször az áramköri lapon elhelyezett fémtárgyakból eredeztethetőek. A keményvashatások eltolják az ideális gömb origóját.



14. ábra - A keményvas hatás által módosított magnetométer adatok 3D-ben

5.1.2. Lágyvas hatás

A lágyvas hatások kisebb mértékben befolyásolják a méréseket. Az érzékelő közelében lévő tárgyakból származnak, amik torzítják a mágneses teret. Ezek az ideális gömböt nyújtják, valamint billentik. Így a regisztrált értékek egy ellipszoidon fekszenek.



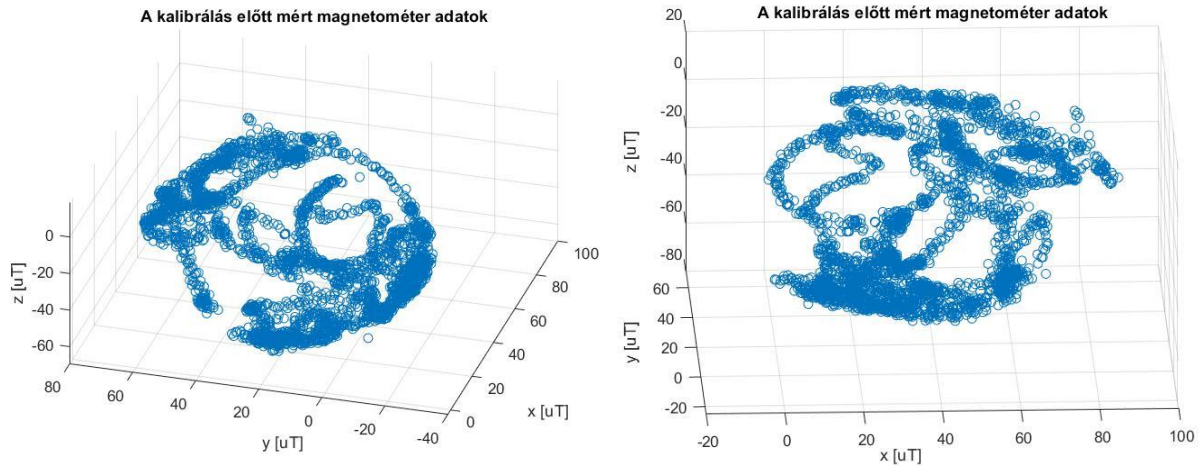
15. ábra - A lágyvas hatás által torzított magnetométer adatok 3D-ben

5.1.3. Nem stacionárius keményvas hatások (mozgó vas)

Az általam vizsgált problémában egy harmadik hatás is beleszól a magnetométeres mérésekbe. A keményvashatásnak ez egy olyan speciális esete, ami abból fakad, hogy a torzítást okozó vas tárgyak nem mozognak együtt a magnetométerrel. Ez jelen esetben az UR5 váza, ami a különböző pozíciókban különböző helyen helyezkedik el a magnetométer koordinátarendszeréhez képest. Ezek a gömb képében kitüremkedéseket, kúpokat hoznak létre, és sajnos nem eliminálhatóak utólag a mérésből.

5.1.4. Magnetométer kalibráló mérés

A kalibrációhoz szükséges mérések elvégzése során igyekeztem a robot Teach Pendant-jának segítségével minél több irányba mozgatni a szenzort egy adott pont körül. A mérési értékeket a 17. ábra szemlélteti.



4. ábra - A kalibrálás előtt mért nyers magnetométer adatok 3D-ben

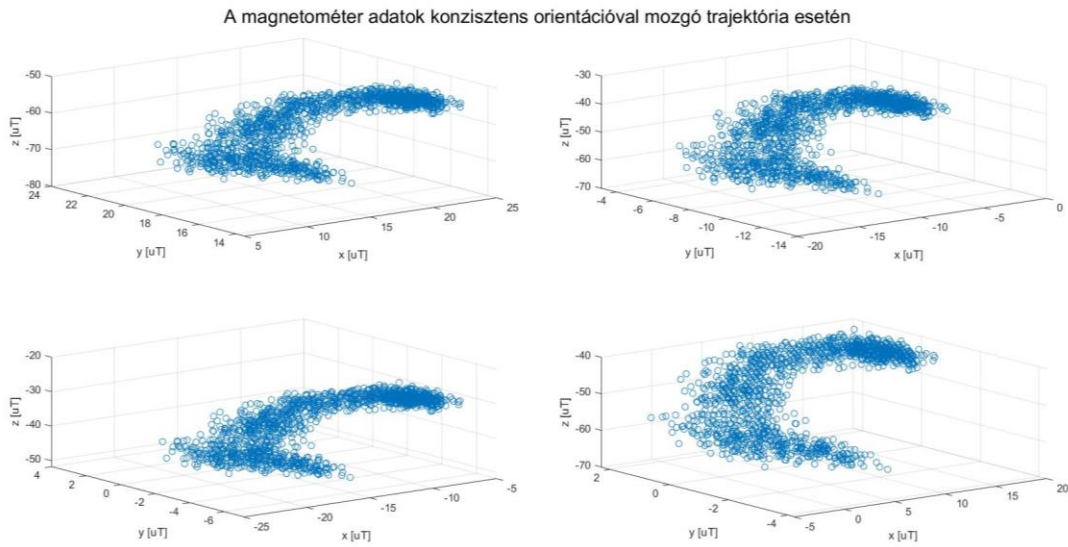
Megvizsgálva a végeredményt látszódik az origó eltolás hatása, valamint a gömb ellipszoiddá válása is. Ezenkívül található 1-2 kicsúcsosodás, amit a közel mozgó robot fémteste okoz, a korábban említett nem állandó keményvas hatás miatt. Ezekbe beleszólhat még a pálya sarkain elhelyezett UWB modulok, valamint az asztal vas szerkezete is.

A kalibrálást MATLAB beépített magnetométer kalibráló függvényével végeztem (magcal). Az eredmény egy 3x1-es vektor (b), mely a középpont eltolást javítja, valamint egy 3x3-as mátrixot (A), ami az elnyújtott ellipszist korrigálja. A kalibrálás a következő (17) képlet szerint megy végbe:

$$X_{out} = (X_{in} - b)A \quad (17)$$

A (17) összefüggésben X_{in} a nyers, míg X_{out} a kalibrált magnetométer adatok.

Kiválasztottam egy olyan trajektóriát, ahol az effektor orientációja nem változik így, ha 3D-ben ábrázolom a trajektória alatt mért magnetométer értékeket akkor minél inkább pontszerű képet kell kapjak. Többféle beépített kalibrálást elvégeztem, melyek eredményét a 18. ábra mutatja be.



17. ábra - A magnetométer adatok ábrázolása 3D-ben a kalibrálások előtt, valamint három különböző kalibrálás után.

Az legjobb kalibrálás adódó eredményei:

$$A = \begin{bmatrix} 1.2781 & 0.0010 & 0.1990 \\ 0.0010 & 0.7003 & 0.1980 \\ 0.1990 & 0.1980 & 0.9303 \end{bmatrix}, b = \begin{bmatrix} 0.0672 \\ 0.0898 \\ 0.0868 \end{bmatrix} \quad (18)$$

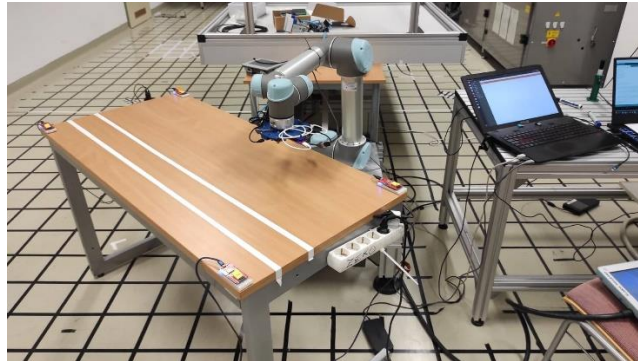
A 18. ábra 4. képén lévő koordinátatengelyeken látszik, hogy közelítette a pontszerű hatást, viszont még mindig rengeteg a zavaró hatás. Ezt a környező keményvas hatások eliminálásával lehet csökkenteni, viszont a robotkar vázából adódó hatások sajnos nem eliminálhatóak a kifejlesztett mérőrendszerben.

5.2. Fingerprint felvétel és az abszolút pozíció becslésére alkalmas rendszer betanítása

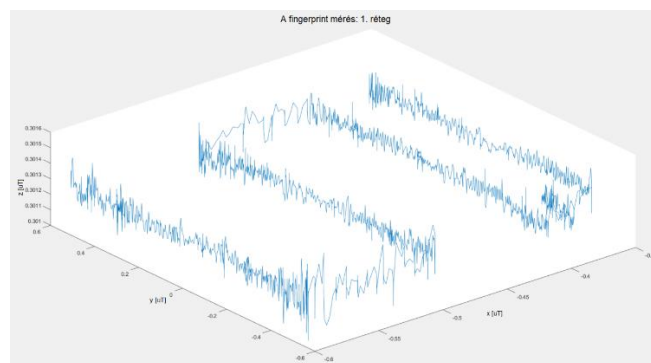
A beltéri lokalizáció egy aktívan kutatott tudományág, mivel a külső technológiától (pl. GPS) elzárt területen más eszközöket szükséges használni. Erre egy jó megoldás a rádiókommunikáció alapú technológiák, ezen belül a jelerősséget felhasználó alkalmazások. A jelerősség alapján két módszercsalád használható a pozíció meghatározásához, a trilateráció és a fingerprint alapú módszerek. Én a fingerprint módszert választottam amire többféle metódust dolgoztak ki, ezek közül a Weighted K-Nearest Neighbor (WKNN), a Random Forest (RF) és az Artificial Neural Network (ANN) a legismertebbek. Kutatások alapján a mesterséges neurális hálózat (ANN) felülmúlja hatékonyságban a többi technikát (Csík, Odry és Sarcevic, 2022), ezért én is ezt a technikát választottam.

5.2.1. A fingerprint trajektória felvétele

A fingerprint mérésekhez végig mértem az UR5 robottal bejárando teret, mely közben rögzítettem az UWB adatait, valamint a csuklókoordinátákból számolt pose-t.

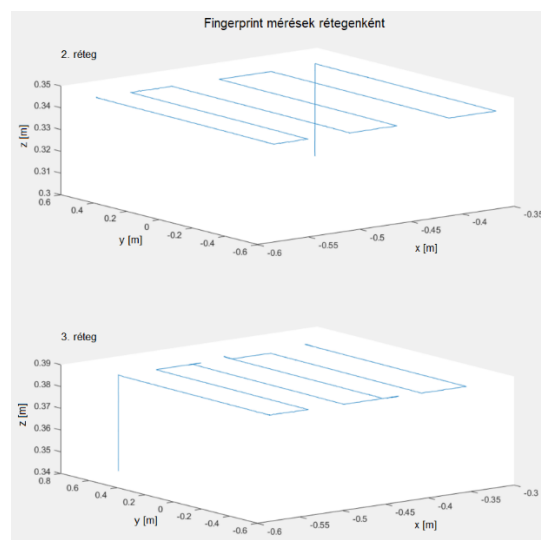


18. ábra - A mérőrendszer a fingerprint mérés közben



5. ábra – A fingerprint mérés, első réteg (részlet)

Rétegeket határoztam meg, melyekkel végig pásztázható a robot konfigurációs tere. Ezekhez a rétegekhez megterveztem a pályákat, majd a robottal megvalósítottam a mozgásokat.



20. ábra - Fingerprint mérések, második és harmadik réteg

5.2.2. Pozícióbecslés mesterséges neurális hálókkal

Az előző alfejezetben bemutatott pásztázó robotmozgások eredményeül két adatbázist építettem. Egyik az UWB által szolgáltatott távolság és RSSI értékeket tartalmazta, míg a másik a ground truth (világkoordinátákhoz tartozó) X, Y és Z értékeket foglalta magában. Ez három rétegű MLP (multilayer perceptron) ANN hálózatot definiáltam, melyben az UWB által szolgáltatott távolság értékeket használtam fel a neurális háló bemeneteként, a ground truth értékeket pedig célként (target) adtam meg.

A hálózat tanítását MATLAB segítségével valósítottam meg a következő paraméterekkel.

- Rejtett rétegek neuronjainak száma: 1-40
- Próbálkozások száma: 4 (ugyanazzal a rejtett réteg számmal négyszer próbálkoztam, mivel a kezdeti súlyok nagy mértékben számítanak a neurális háló végkimenetében. A kezdeti paraméterek pedig randomszámként generálódnak minden tanítás előtt.)
- Iterációk száma: 5000

Az eredményeknél a valós ground truth koordinátákhoz képest számolt átlagos hiba minimalizálására törekedtem.

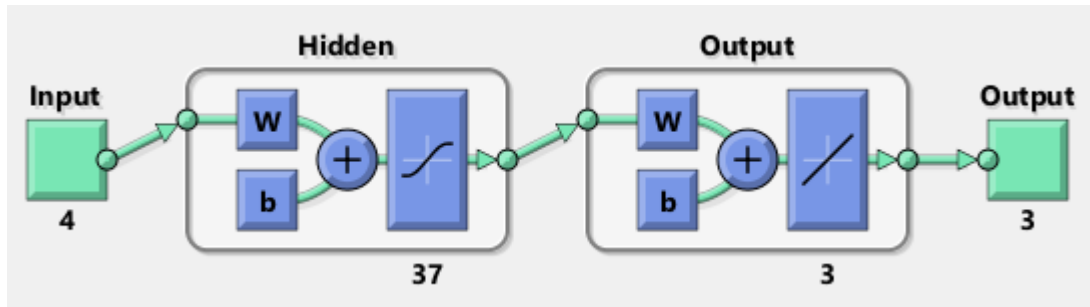
A tanítás eredményeként a következő pozíciómérés pontosságot és hálózat felépítést kaptam.

UWB távolságokkal:

- Meghatározott legkisebb átlagos hiba magnitúdóban (pont-pont távolság): 60,19 mm
- Rejtett rétegek száma: 37 (legjobb topológia)

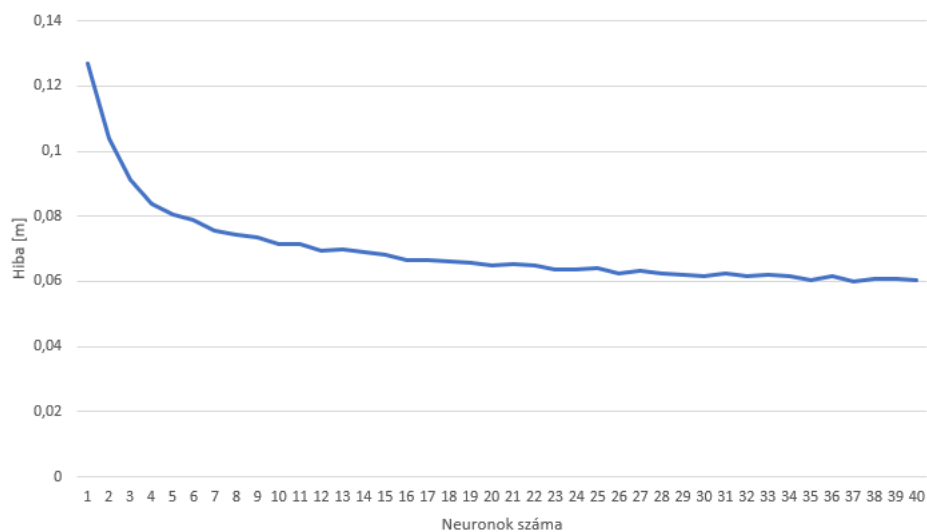
RSSI mérésekkel:

- Meghatározott legkisebb átlagos hiba magnitúdóban (pont-pont távolság): 294,59 mm
- Rejtett rétegek száma: 16 (legjobb topológia)



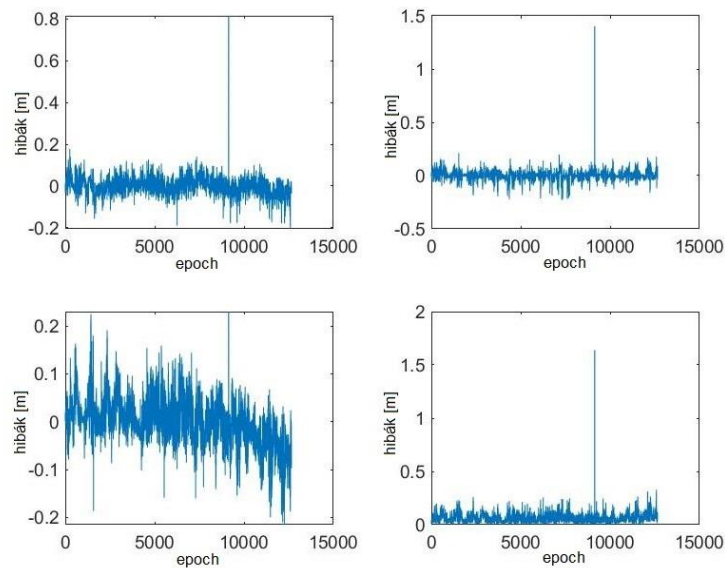
21. ábra - A kiválasztott 37 neuronnal rendelkező MLP hálózat

Az UWB-s mérés 3D pozíciómérés tekintetében meglepően jó performanciát és eredményeket szolgáltat. Látható, hogy már 15 rejtett réteg esetén is 70mm-en alatt van az átlagos hiba (22-23. ábra). A 22. ábra alapján az a következtetés vonható le, hogy a kifejlesztett és UWB-vel meghajtott ANN alkalmas térbeli (3D) pozíciómérést valósít meg, így beltérben a GPS kiváltható a kifejlesztett technológiával.



22. ábra – Az átlagos hiba alakulása a neuronok számának növelésével

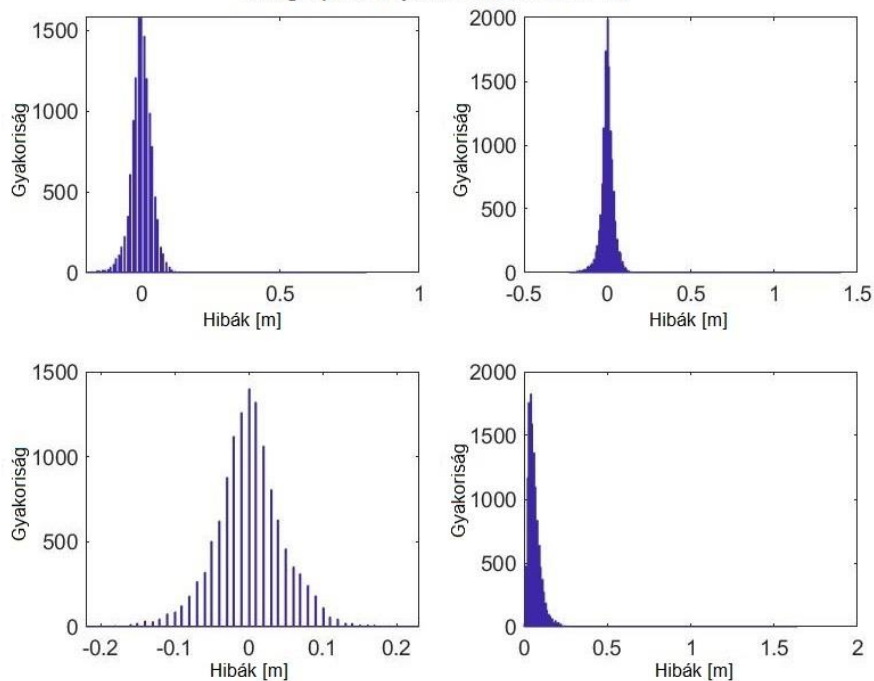
A 37 neuronnal rendelkező neuronháló hibái a fingerprint trajektóriára alkalmazva



23. ábra - A kiválasztott legjobb neuronháló hibái a fingerprint trajektóriára alkalmazva

A 23. ábrán megfigyelhető egy kiugró érték, amit az UWB távolságmérésekből eredő mérési hiba okozott.

A 37 neuronnal rendelkező neuronháló hibáinak hisztogramja a fingerprint trajektóriára alkalmazva



24. ábra - A kiválasztott legjobb neuronháló hibáinak hisztogramja a fingerprint trajektóriára alkalmazva

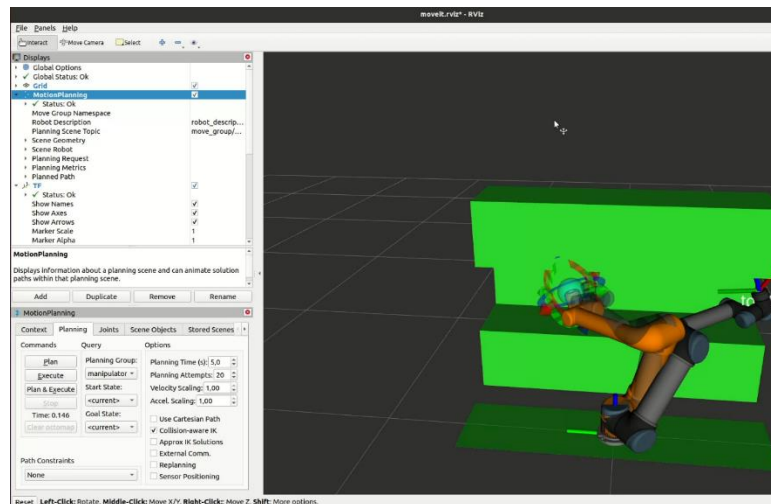
5.3. Az UR5 trajektóriák megalkotása

A mérések előtt az UR5-ös mozgásterét lekorlátoztam a fingerprintek során végigpásztázott területre, hogy a neuronháló a mozgások során alkalmas értékeket adjon az

UWB távolságmérésekből. Erre lehetőséget nyújtott az MoveIt ROS interfész, melynek segítségével korlátozhatóvá vált a robot mozgástere egyszerű térbeli téglatest alakzatok robot konfigurációs terébe történő integrálásával. Az alábbi programrészlet ismerteti, hogy a téglatest dimenziói és térbeli koordinátái definiálásával valósítható meg az akadályok feltöltése a *collision_object* osztályba, melyet a MoveIt *planning_scene_interface* osztálya kapcsol a robotoz.

```
moveit_msgs::CollisionObject collision_object;
collision_object.header.frame_id=move_group_interface.getPlanningFrame();
collision_object.id = "box1";
shape_msgs::SolidPrimitive primitive;
primitive.type = primitive.BOX;
primitive.dimensions.resize(3);
primitive.dimensions[primitive.BOX_X] = 0.1;
primitive.dimensions[primitive.BOX_Y] = 1.5;
primitive.dimensions[primitive.BOX_Z] = 0.5;
geometry_msgs::Pose box_pose;
box_pose.orientation.w = 1.0;
box_pose.position.x = 0.5;
box_pose.position.y = 0.0;
box_pose.position.z = 0.25;
collision_object.primitives.push_back(primitive);
collision_object.primitive_poses.push_back(box_pose);
collision_object.operation = collision_object.ADD;
std::vector<moveit_msgs::CollisionObject> collision_objects;
collision_objects.push_back(collision_object);
planning_scene_interface.addCollisionObjects(collision_objects);
```

A 25. ábrán a zölddel jelölt területrészek a mozgás során tiltott területek (Collision objects) az adott konfigurációban.



25. ábra - A robot mozgásának korlátozása az RViz felületen és a 3D modell

A trajektóriák megalkotásánál többféle módszert alkalmaztam, hogy minél több adat álljon rendelkezésre a későbbi feldolgozások során. A következő trajektória csoportokat valósítottam meg.

- Kézi mozgatás: A robotot a teach pendant-ról irányítottam és így alkottam meg a mozgásokat. A robot állandó konstans sebességgel mozgott.
- MoveIt: A RViz felületen belül MoveIt parancsokkal mozgattam a robotot két pont között a behatárolt mozgási területen. A mozgások gyorsak és dinamikusak voltak.
- Egyenesek menti mozgások: A MoveIt trajektóriákhoz hasonló mozgások, csak abban különbözik, hogy a robot egyenes vonalú pályán jut el a kezdőpontból a végpontig. Kevésbé dinamikus mozgások valósultak meg.
- Free-run: Szabadonfutás üzemmód. A UR egyik funkciója, hogy a Free-run üzemmódot aktiválva kézzel mozgatható a robot és különböző pozícióba/orientációba állítható kis erők hatására is. Ebben az esetben mozgások méréselt dinamikával valósultak meg.

6. UWB FINGERPRINT-MARG SZENZORFÚZIÓ MEGVALÓSÍTÁSA

A mozgó egység állapota egy hat szabadságfokú póz vektorral írható le, mely magában foglalja a térbeli X,Y, Z koordinátákat (pozíció) és a koordináták körüli szögelfordulásokat (azaz az end effektor orientációját). A póz becsléséhez szenzorfúziós keretrendszert kell alkalmazni, hiszen egyik technológia sem alkalmas önmagában a hat szabadságfokú információ meghatározására. Az UWB esetében a következő jellemzőket foglalhatjuk össze.

- Hátrány: 6-7 cm-es pontossággal lehet a térbeli pozíciót mérni, mely további pontosítása indokolt. Nincs információ az orientációra. A mérést alacsonyabb mintavételi

frekvencia jellemzi, mely dinamikus rendszerek esetében megnehezíti az irányítást. A technológia önmagában nem ad elég megbízható és alkalmas visszacsatolást a felsőbb irányítások számára.

- Előny: abszolút pozíciót szolgáltat, a mérésekhez nincs szükség integrálás végrehajtására, így a következtetést nem jellemzi csúszó hatás (drift).

A MARG mérőegység esetében a térbeli orientáció meghatározása hatékonyan lekezelhető, viszont a térbeli pozícióra vonatkozó számításokat a következők jellemzik.

- Hátrány: A térbeli koordináták a relatív gyorsulás adatok integrálásával határozhatók meg. A mérésben megjelenő zaj és külső zavarások jelentős csúszással jellemzik az kiszámított értékeket. A kalibráció hiánya jelentős mértékben befolyásolja a koordinátákra vonatkozó következtetések minőségét. A relatív információk (gyorsulás és szögsebesség) miatt a számítások csak rövid távon tekinthetők megbízhatónak. Hosszabb trajektóriák esetén nem lehet pontos pozíciót meghatározni a módszerrel.
- Előny: magas mintavételi frekvencia; a mérési eredmények lefedik a pozícióra és orientációra vonatkozó következtetéseket; a magas mintavételi frekvencia lehetővé teszi a 6 DOF póz számítását addig, míg nincs referenciajel.

A fenti jellemzők azt nyomatékosítják, hogy a két technológia fúzionálásával hatékony (javított performanciával jellemezhető) póz mérés valósítható meg. A fúzió kiépítésére egy kiterjesztett Kalman szűrőt alkalmaznak, mely a szakirodalmi áttekintőben feldolgozott (Odry, Kecskes, Csik, Hashim és Sarcevic, 2022), (Odry, Kecskes, Csik, Rodríguez-Reséndiz, Carbone és Sarcevic 2022) munkákban bemutatott keretre épül.

6.1. Kiterjesztett Kalman szűrő predikciós fázisa

Az EKF keret egy mozgás- és egy megfigyelés modell definiálását igényli. A korábbiakban bemutatott MARG-alapú kinematikai modell (16) alkalmas a mozgás modell definiálására. Nevezetesen, az $x_{k+1} = f(x_k, \mu_k)$ nemlineáris rendszert a következő állapotvektor jellemezheti:

$$x_k = (p_k, v_k, a_{o,k}, q_k, \omega_{o,k})^T. \quad (19)$$

A fenti (19) egy 16 dimenziós állapotvektor, mely a térbeli pozíció, sebesség és orientációt leíró vektorok mellett, a gyorsulásmérő és giroszkóp kimenetét terhelő időben változó offset jeleket is jellemzi. Minden állapotot bizonytalan, ezért az állapot propagációjának bizonytalanságát Q kovariancia mátrixszal adjuk meg, mely egy 16×16 dimenziós mátrix, ahol a főátlóban az

egyes állapotokhoz tartozó szórásnégyzetek kapnak helyet (σ). Q további elemeit nullák alkotják, mert azt feltételezem, hogy az állapotok függetlenek (azaz nincs korreláció):

$$Q = \text{blkdiag}(\sigma^p I_3, \sigma^v I_3, \sigma^{a_o} I_3, \sigma^q I_4, \sigma^{\omega_o} I_3). \quad (20)$$

A (16) összefüggés eredménye, hogy a diszkrét idejű nemlineáris mozgás modell a következő formában írható le:

$$x_{k+1} = f(x_k, u_k) = \begin{bmatrix} p_k + T_s v_k + \frac{T_s^2}{2} (A_k - g) \\ v_k + T_s (A_k - g) \\ a_{o,k} \\ q_k + \frac{T_s}{2} \varrho(q_k) (0, \Omega_k^T - \omega_{o,k}^T)^T \\ \omega_{o,k} \end{bmatrix} \quad (21)$$

A fenti összefüggésben $A_k - g$ különbség a rendszer gyorsulása az inercia keretben. A gyorsulás integrálása a kinematikai egyenletek értelmében a sebesség és pozíció vektorokat adja meg. Az orientáció reprezentációja kvaternió; a kvaternió a giroszkóp adatok Ω_k integrálásával határozható meg. A Kalman szűrő ezen mozgás modell alapján következtet az állapot vektorra, ezt a szakirodalom $\hat{x}_k^-$ a priori becslésnek nevezi. A priori becslést annak bizonytalanságával jellemezzük (P_k^- kovariancia mátrixs), ez az irodalmi áttekintésben ismertetett (14) összefüggés szerint:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k), \quad (22)$$

$$P_k^- = \phi P_{k-1} \phi^T + Q, \quad \phi = \left. \frac{\partial f}{\partial x} \right|_{\hat{x}_{k-1}}$$

A fenti egyenletek segítségével a Kalman szűrő predikciós fázisa futtathatóvá válik. Az implementációhoz egy osztályt hoztam létre MATLAB-ban, melynek a *predictPose* függvényében implementáltam a fenti összefüggéseket.

```
function predictPose(obj, a, w, dt)
q = obj.Quaternion;
obj.aWorld = obj.quaternProd(q, obj.quaternProd([0 (a-
transpose(obj.X(7:9))))], obj.quaternConj(q)));
obj.aWorld = transpose(obj.aWorld(2:4)) - obj.g;

pos = obj.X(1:3);
```

```

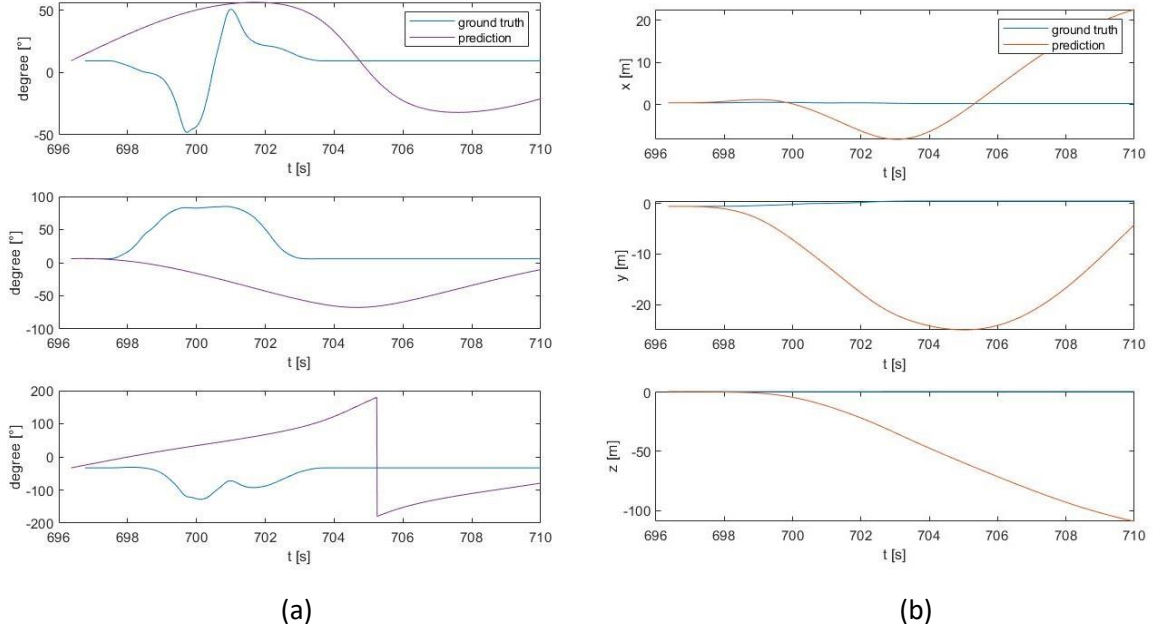
vel = obj.X(4:6);
biasa = obj.X(7:9);
q = transpose(obj.X(10:13));
biasg = obj.X(14:16);

obj.X(1:3,1) = eye(3)*pos + dt*eye(3)*vel +
((dt^2)/(2))*eye(3)*obj.aWorld;
obj.X(4:6,1) = eye(3)*vel + dt*eye(3)*obj.aWorld;
obj.X(7:9,1) = eye(3)*biasa;

qDot = 0.5 * obj.quaternProd(q, [0 w(1)-obj.X(14) w(2)-obj.X(15)
w(3)-obj.X(16)]);
q = q + qDot * dt;
q = q / norm(q);
obj.X(10:13,1) = transpose(q);
obj.X(14:16,1) = eye(3)*biasg;

```

A predikció futtatásához a Q kovariancia mátrixot be kell állítani. Ez heurisztikus úton, trial-and-error alapon valósult meg MATLAB környezetben. A következtetések alapján a következő varianciák lettek beállítva: $\sigma^P = 0.1$, $\sigma^V = 0.05$, $\sigma^{a_o} = 0.0001$, $\sigma^q = 0.001$, $\sigma^{\omega_o} = 0.0001$. A megfontolás az volt, hogy az ofszetet terhelő zaj mértéke a legelenyészőbb, ezért a_o és ω_o állapotokat relatív kis bizonytalanság terheli. Az orientációt a giroszkóp adatok integrálásával határozzuk meg, melynek csúszása az idővel arányosan növekszik, ezért az orientációhoz nagyobb bizonytalanságot rendeltem. Hiszen a sebesség a gyorsulás integrálásával van meghatározva, ezért itt is csúszásra számíthatunk, viszont ennek mértéke nagyobb, mint ami az orientációnál jelentkezik, hiszen a pontatlan orientáció, és parazita gyorsulások ezt nagy mértékben befolyásolják. Ennek következménye egy lényegesebben nagyobb variancia választását eredményezte ($\sigma^V = 0.05$). Végül a pozíció a bizonytalan sebességvektor integrálásával kerül meghatározásra, melynek bizonytalansága várhatóan a legnagyobb, így $\sigma^P = 0.1$ választása tűnik racionálisnak. Az alábbi diagrammok (30. ábra) a predikció futtatását mutatják, ahol megfigyelhető, hogy a csúszás jelentős mértékben jelentkezik az orientáció és pozíció becslésekben.



26. ábra – A predikció eredményeül kapott poz eredmények: (a) Euler szögek (b) 3D pozíciók

6.2. Kiterjesztett Kalman szűrő korrekciós fázisa

A becslés javítását a Kalman szűrő a korrekciós fázisban hajtja végre. Ehhez egy megfigyelés modellt kell használni, mely segítségével összehasonlítható a mérés és az a priori becslésből számítható mérés várható értéke. A leképezés az állapotok teréből a valós mérések terébe viszonylag egyszerű, hiszen a 16 dimenziós állapotvektorból az orientációra és a pozíció vektorra is méréseink állnak rendelkezésre. Az orientációra vonatkozó mérést a gyorsulásérzékelő és magnetométer vektorok segítségével létrehozott rotációs mátrix segítségével határozzuk meg a (11) összefüggés alapján. A rotációs mátrix ismeretében a kvaternió megfigyelés direkt számítható (12) segítségével. A pozícióra vonatkozó megfigyelést a neurális hálózat kimenete adja. Ennek megfelelően a megfigyelésmodell kimenete $z_k = (p_k, q_k)^T$. Az állapotvektor leképezését H matrix segítségével adjuk meg, melynek értéke ennek értelmében:

$$H = \begin{bmatrix} I_3 & 0_{3 \times 10} & 0_{3 \times 3} \\ 0_{4 \times 9} & I_4 & 0_{4 \times 3} \end{bmatrix}. \quad (23)$$

A p_k és q_k megfigyeléseket hasonlóan bizonytalansággal jellemeztem. A pozíció mérésre vonatkozó analízis alapján a bizonytalanságnak $\sigma^P = 0.05$ értéket választottam. Az orientáció mérést a rendszer mozgása és külső mágneses zavarások befolyásolják, ebből kifolyólag $\sigma^Q = 0.07$ bizonytalansággal jellemeztem ezeket a megfigyeléseket. A korrekció bizonytalanságának

jellemzéséhez a Kalman szűrő az R kovariancia mátrixot használja, ez egy 7×7 blokkdiagonális mátrix a következő felépítéssel:

$$R = \text{blkdiag}(\sigma^P I_3, \sigma^Q I_4). \quad (24)$$

A Kalman szűrő korrekciós fázisa három lépésben valósul meg. Az algoritmus az első lépésben kiszámítja az erősítést G_k , mely meghatározza azt, hogy milyen súllyal vegyük figyelembe a mérések és következtések közötti különbséget ($z_k - H\hat{x}_k^-$). Ezt követően az erősítést felhasználva meghatározza a posteriori becslést $\hat{x}_k$ a második lépésben. Végül a harmadik lépésben a posteriori becslés bizonytalanságát P_k számítja ki az algoritmus.

$$\begin{aligned} G_k &= P_k^- H^T (H P_k^- H^T + R)^{-1}, \\ \hat{x}_k &= \hat{x}_k^- + G_k (z_k - H \hat{x}_k^-), \\ P_k &= (I - G_k H) P_k^-. \end{aligned} \quad (24)$$

A fenti összefüggéseket egy *updatePose* függvényben implementáltam (31 ábra).

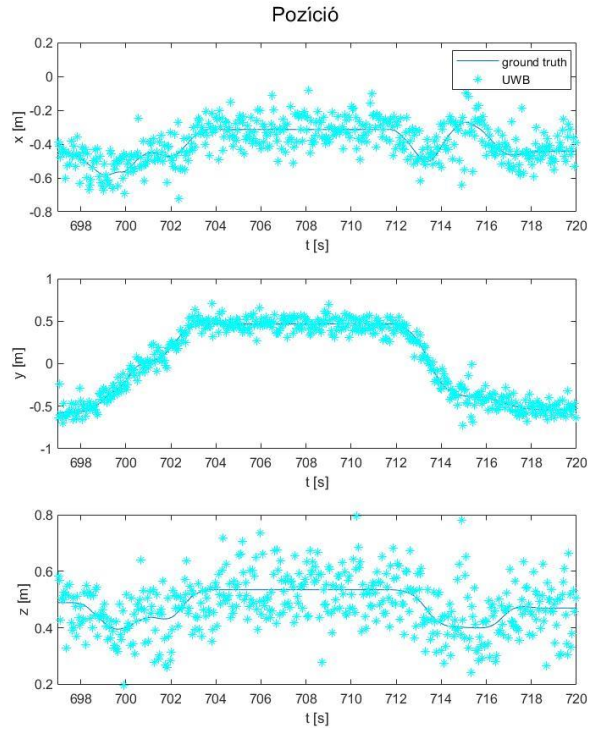
```
function updatePose(obj, z)
v = [transpose(z); obj.quatUpdate] - [obj.X(1:3); obj.X(10:13)];
R = blkdiag(eye(3).*obj.posObsNoise, eye(4).*obj.quatObsNoise);
H = [...
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; ...
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; ...
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; ...
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0; ...
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0; ...
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0; ...
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0];

S = H * obj.P * transpose(H) + R;
K = obj.P * transpose(H) * (S \ eye(size(S)));
obj.X = obj.X + K * v;

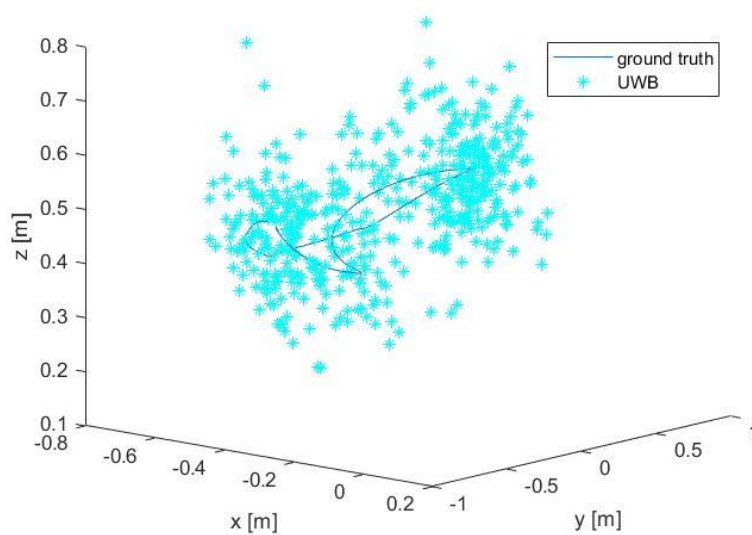
q = obj.X(10:13);
q = q / norm(q);
obj.X(10:13) = q;
obj.Quaternion = transpose(obj.X(10:13));

I = eye(length(obj.X));
obj.P = (I - K * H) * obj.P;
end
```

Az alábbi ábrák a korrekciós jeleket jelentős szórását mutatják a ground truth mellett. Megfigyelhető, hogy az UWB-alapú ANN következtető gép a valós állapotok környezetében szolgáltat abszolút pozíció méréseket, így ezek becsatolása az EKF algoritmusba alkalmasnak mutatkozik az integrálásból eredő drift hatások korrigálására.



27. ábra – A mért UWB pozíciók a ground truth pályához képest



28. ábra - A mért UWB pozíciók a ground truth pályához képest 3D-ben

7. EREDMÉNYEK

A predikció és korrekció lépések rekurzív végrehajtásával a szenzorfüziót sikeresen megvalósítja a Kalman szűrő. Az eredményeket két trajektóriára értékeltem ki. A pozíció és orientáció becslés minőségét az átlagos négyzetes hibával (mean squared error – MSE) jellemzem a következő szerint:

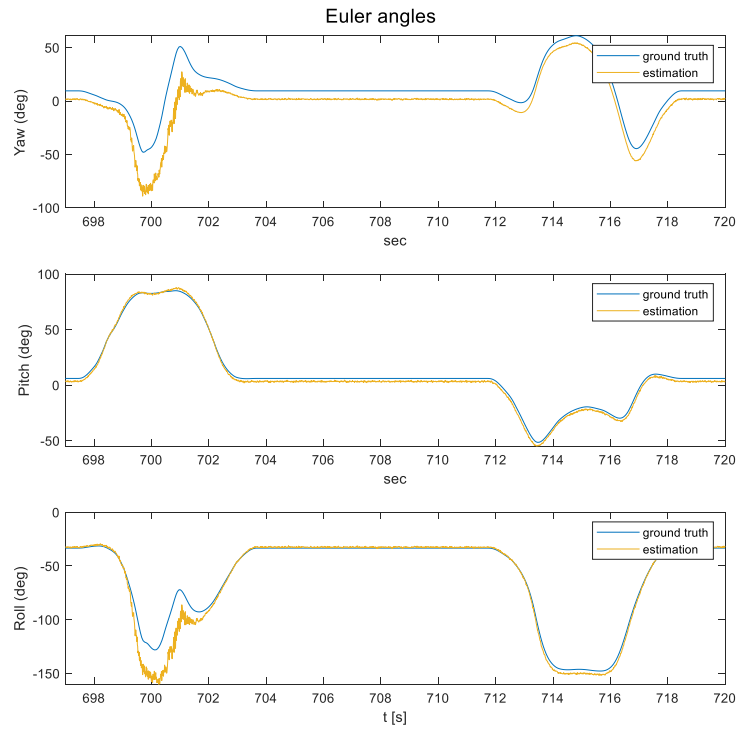
$$\begin{aligned} Err_{XYZ} &= \frac{1}{N} \sum_{i=1}^N (\hat{x}_i - x_i)^2 + \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 + \frac{1}{N} \sum_{i=1}^N (\hat{z}_i - z_i)^2 \\ Err_{RPY} &= \frac{1}{N} \sum_{i=1}^N (\hat{\psi}_i - \psi_i)^2 + \frac{1}{N} \sum_{i=1}^N (\hat{\theta}_i - \theta_i)^2 + \frac{1}{N} \sum_{i=1}^N (\hat{\phi}_i - \phi_i)^2 \end{aligned} \quad (25)$$

ahol N jelöli a minták számát, a differenciák jelölik a becsült és valós értékek közötti különbségeket, Err_{RPY} jelöli az orientáció becslés teljes performanciáját, míg Err_{XYZ} a teljes pozícióbecslés minőségét adja.

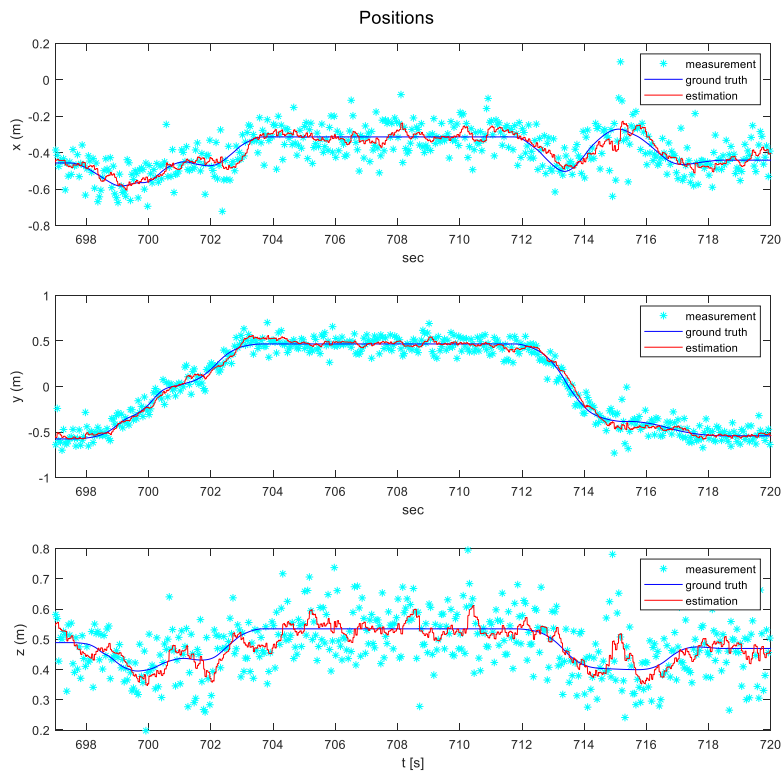
Mindkét scenárióban megfigyelhető, hogy a pozíció mérések szórása jelentős mértékű a háromdimenziós térben, továbbá ritkábban állnak rendelkezésre ezek a mérések. Ugyanakkor, az orientáció mérést jelentős zaj terheli. Ennek oka az, hogy parazita gyorsulások hozzáadódnak a gravitációs gyorsulás vektorához, így a rotációs mátrix minőségét a parazita gyorsulások direkt befolyásolják. Az orientációnál az is megfigyelhető, hogy jelentős ofszettel írják le a valós orientációt. Ennek oka az, hogy az IMU egység nem lett kalibrálva, így a nem ortogonalitás, tengelyek menti ofszet értékek és erősítési hibák konstans eltérést eredményeznek mindhárom tengely mentén.

Ugyanakkor az is megfigyelhető, hogy a kifejlesztett szenzorfüzióval jelentősen csökkent a póz mérés bizonytalansága. Az EKF kimenetét kisebb szórás jellemzi és a drift hatás sikeres kompenzációja valósul meg, hiszen a becsült értékek mindvégig a valós állapotok környezetében tartózkodnak. Az EKF jelentősen javított a pozícióbecslés minőségén. A performancia indexeket az egyes scenáriókhoz tartozó táblázatok foglalják össze.

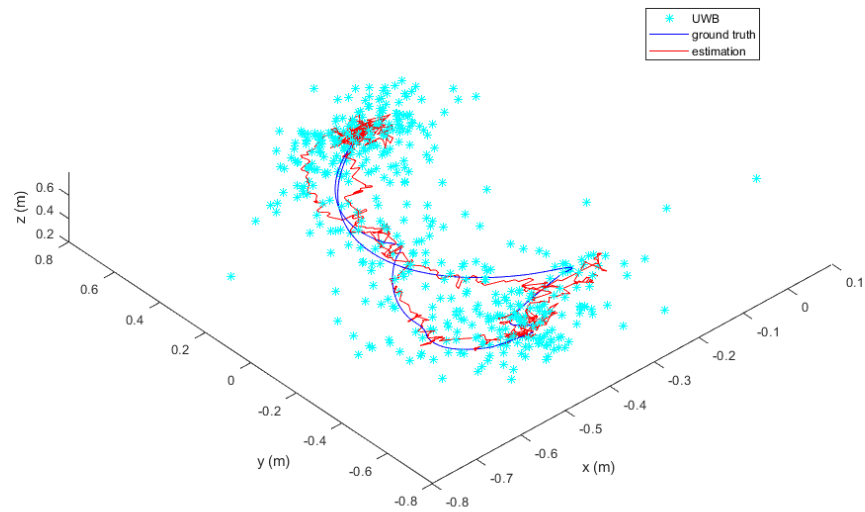
7.1. Első szcenárió



29. ábra – Az orientáció becslés performanciája az első szcenárióban



30. ábra - A mért és becslt pozíció alakulása az első szcenárió alatt



31. ábra - A mért és becsült pozíció alakulása az első szcenárió alatt 3D-ben

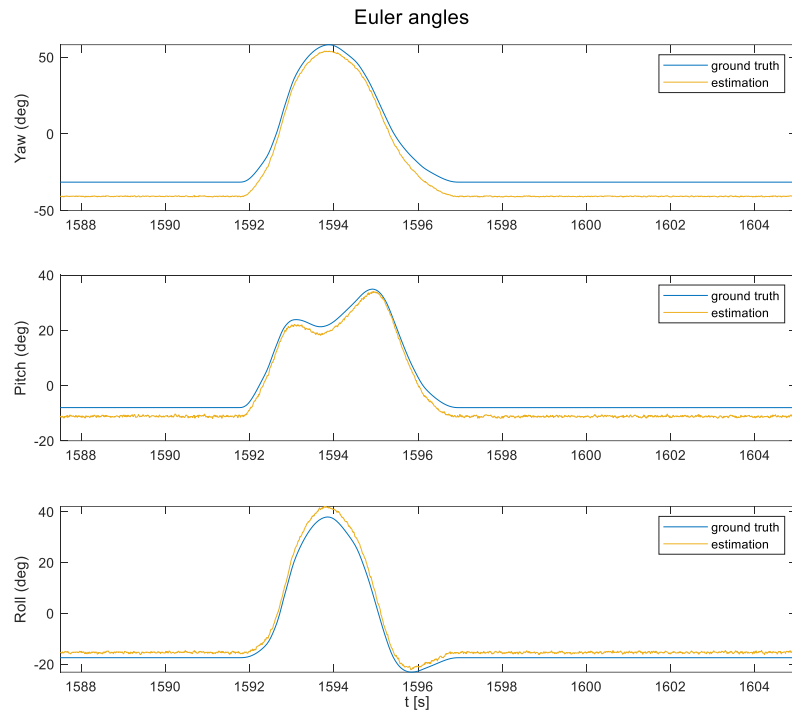
	Yaw [°]	Pitch [°]	Roll [°]	RPY együtt
Szögekre vonatkozó MSE értékek	200.01	6,6253	78,963	285,6

1. Táblázat – orientációbecslés performanciája (1. szcenárió)

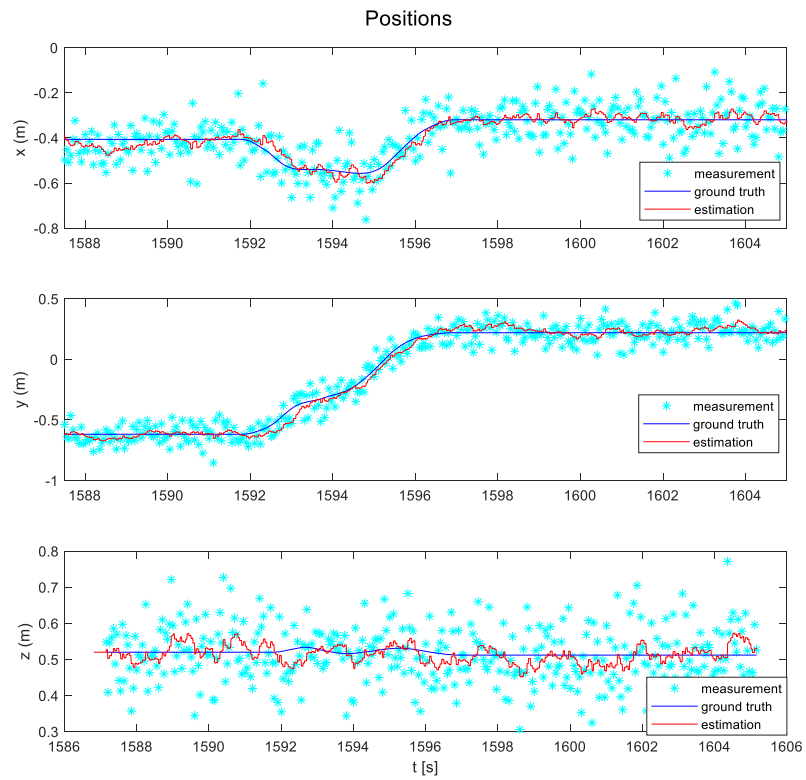
	x [m]	y [m]	z [m]	XYZ együtt
Pozíció becslésre vonatkozó MSE értékek	0,00098538	0,0013676	0,00093037	0,0032834
Pozíció mérésre (NN kimenet) vonatkozó MSE értékek	0,0048384	0,0048921	0,0048702	0,014601
Javulás	79,63%	72,04%	80,9%	77,51%

2. Táblázat – pozícióbecslés performanciája (1. szcenárió)

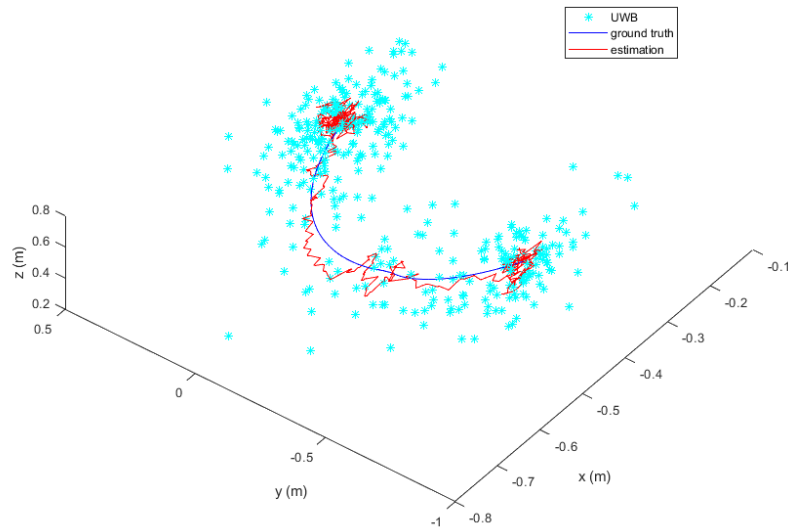
7.2. Második szcenárió



326. ábra - Az orientáció becslés performanciája az első szcenárióban



33. ábra - A mért és becslt pozíció alakulása az második szcenárió alatt



34. ábra - A mért és becsült pozíció alakulása az második szcenárió alatt 3D-ben

	Yaw [°]	Pitch [°]	Roll [°]	RPY együtt
Szögekre vonatkozó MSE értékek	73,503	8,8575	5,7912	88,152

1. Táblázat – orientációbecslés performanciája (2. szcenárió)

	x [m]	y [m]	z [m]	XYZ együtt
Pozíció becslésre vonatkozó MSE értékek	0,00060516	0,0013316	0,00060245	0,0025392
Pozíció mérésre (NN kimenet) vonatkozó MSE értékek:	0,0040852	0,0041201	0,003924	0,012129
Javulás	85,19%	67,68%	84,65%	79,07%

8. 2. Táblázat – pozícióbecslés performanciája (2. szcenárió)

9. KONKLÚZIÓ

Megmutattam, hogy szenzorfüzió alkalmazásával jelentősen javítható pozíció és orientáció becslés a háromdimenziós térben. A szenzorfüzió megvalósításához alkalmas hardveres és szoftveres keretrendszert hoztam létre, melyben a ground truth és nyers mérési eredmények szimultán regisztrálhatók. Egy UR5 ipari robot segítségével végig pásztáztam a konfigurációs teret és adatbázist építettem az UWB és ground truth értékekről. Az így létrehozott adatbázist felhasználva neurális hálózatot fejlesztettem ki, mely segítségével pozíció megfigyeléseket állítottam elő. Ezen pozíció megfigyelések nem adnak elég pontos leírást a rendszer valós állapotára, ezért szenzorfüzióval tovább javítottam a pozíciómérési módszert. Az IMU és magnetométerek adataival egy dinamikus modellt hajtottam meg, melyet alkalmas

bizonytalanság értékekkel jellemeztem. A Kalman szűrő ezt a dinamikus modellt használta arra, hogy a pozíció és orientáció adatokra következtessen. Az abszolút pozíció beépítésével a csúszási hibát kompenzálni tudtam, és a két technológia ötvöztetésével jelentős javulást értem el a pozíció és orientáció becslés minőségében.

Az elért eredmények azonban nem tökéletesek. A bemutatott ábrákon látható, hogy hibák most is jelentkeznek a becslésben, ugyan azok mértéke lényegesen kisebb, mint az neurális hálózat alapú megfigyelések. A hibák sok forrásból származhatnak:

- A gyorsulásérzékelő és giroszkóp az idő hiányában nem lett kalibrálva. Ennek következménye, hogy a nem-ortogonalitás, különböző erősítések és offset értékek szignifikánsan rontanak a következtetések minőségén. Ehhez tartozik az is, hogy a gravitációs vektor így nincs tökéletesen izolálva a rendszer gyorsulásától.
- A magnetométert kifinomultabb folyamatban kellene kalibrálni, hogy a zavaró hatások ne jelentkezzenek minden mozgásban (mozgó vas hatása).
- A szűrő paraméterek heurisztikus úton lettek beállítva, ami szuboptimális szűrő működést eredményez. A paraméterek finomhangolásával vagy optimalizációjával a performancia tovább javítható.
- A szenzorszigeten elhelyezett további érzékelők integrálásának hiánya is hozzájárult a gyengébb performanciához.

A fenti pontok a fő továbbfejlesztési irányokat foglalják össze. A kifejlesztett szenzorsziget ultrahangos távolságmérőt, valamint a kamerás optical flow szenzort is tartalmaz. Ez lehetővé teszi a módszerek bővítését a munka következő fázisában. A mozgás és megfigyelés modellek kiegészítése is értelmes továbbfejlesztési irány lehet, pl. ahogy azt a korábban említett irodalmi hivatkozásban is (Odry, Kecskes, Csik, Hashim és Sarcevic, 2022) javasolták.

IRODALOMJEGYZÉK

- S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, “Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 8729–8736.
- G. Cioffi and D. Scaramuzza, “Tightly-coupled fusion of global positional measurements in optimization-based visual-inertial odometry,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 5089–5095.
- C. Li, S. Wang, Y. Zhuang, and F. Yan, “Deep sensor fusion between 2d laser scanner and imu for mobile robot localization,” *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8501–8509, 2019.
- J. Szrek, P. Trybala, M. Goralczyk, A. Michalak, B. Ziketek, and R. Zimroz, “Accuracy evaluation of selected mobile inspection robot localization techniques in a gnss-denied environment,” *Sensors*, vol. 21, no. 1, p. 141, 2021.
- H. A. Hashim, “Gps-denied navigation: Attitude, position, linear velocity, and gravity estimation with nonlinear stochastic observer,” in 2021 American Control Conference (ACC). IEEE, 2021, pp. 1149–1154.
- H. A. Hashim, M. Abouheaf, and M. A. Abido, “Geometric stochastic filter with guaranteed performance for autonomous navigation based on imu and feature sensor fusion,” *Control Engineering Practice*, vol. 116, p. 104926, 2021.
- H. A. Hashim and A. E. Eltoukhy, “Landmark and imu data fusion: Systematic convergence geometric nonlinear observer for slam and velocity bias,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.
- N. Akai, L. Y. Morales, and H. Murase, “Mobile robot localization considering class of sensor observations,” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 3159–3166.
- J. P. Silva do Monte Lima, H. Uchiyama, and R.-i. Taniguchi, “End-to-end learning framework for imu-based 6-dof odometry,” *Sensors*, vol. 19, no. 17, p. 3777, 2019.

- J. Svacha, J. Paulos, G. Loianno, and V. Kumar, “Imu-based inertia estimation for a quadrotor using newton-euler dynamics,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3861–3867, 2020.
- J. Macdonald, R. Leishman, R. Beard, and T. McLain, “Analysis of an improved imu-based observer for multirotor helicopters,” *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3, pp. 1049–1061, 2014.
- M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, “Aboldeepio: A novel deep inertial odometry network for autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1941–1950, 2019.
- J. Chen, B. Zhou, S. Bao, X. Liu, Z. Gu, L. Li, Y. Zhao, J. Zhu, and Q. Lia, “A data-driven inertial navigation/bluetooth fusion algorithm for indoor localization,” *IEEE Sensors Journal*, 2021.
- Y. Wang, H. Cheng, C. Wang, and M. Q.-H. Meng, “Pose-invariant inertial odometry for pedestrian localization,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.
- H. Yan, Q. Shan, and Y. Furukawa, “Ridi: Robust imu double integration,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 621–636.
- W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, “Tlio: Tight learned inertial odometry,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020.
- M. Brossard, A. Barrau, and S. Bonnabel, “Ai-imu dead-reckoning,” *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.
- M. Brossard, S. Bonnabel, and A. Barrau, “Denoising imu gyroscopes with deep learning for open-loop attitude estimation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4796–4803, 2020.
- W.-Y. Kim, H.-I. Seo, and D.-H. Seo, “Nine-axis imu-based extended inertial odometry neural network,” *Expert Systems with Applications*, vol. 178, p. 115075, 2021.
- A. Roux, S. Changey, J. Weber, and J.-P. Lauffenburger, “Cnn-based invariant extended kalman filter for projectile trajectory estimation using imu only,” in *2021 International Conference on Control, Automation and Diagnosis (ICCAD)*. IEEE, 2021, pp. 1–6.

- A. Odry, I. Kecskes, P. Sarcevic, Z. Vizvari, A. Toth, and P. Odry, “A novel fuzzy-adaptive extended kalman filter for real-time attitude estimation of mobile robots,” *Sensors*, vol. 20, no. 3, p. 803, 2020.
- A. Odry, “An open-source test environment for effective development of marg-based algorithms,” *Sensors*, vol. 21, no. 4, p. 1183, 2021.
- Roh, M.S.; Kang, B.S. “Dynamic Accuracy Improvement of a MEMS AHRS for Small UAVs”. *Int. J. Precis. Eng. Manuf.* 2018, 19, 1457–1466.
- Liu, S.; Lyu, P.; Lai, J.; Yuan, C.; Wang, B. “A fault-tolerant attitude estimation method for quadrotors based on analytical redundancy.” *Aerosp. Sci. Technol.* 2019, 93, 105290.
- Zhang, T.; Liao, Y. “Attitude measure system based on extended Kalman filter for multirotors.” *Comput. Electron. Agric.* 2017, 134, 19–26. [CrossRef]
- Kuipers, J.B. “Quaternions and Rotation Sequences”; Princeton University Press: Princeton, NJ, USA, 1999; Volume 66.
- Diebel, J. “Representing attitude: Euler angles, unit quaternions, and rotation vectors.” *Matrix* 2006, 58, 1–35.
- Aggarwal, P. “MEMS-Based Integrated Navigation” Artech House: Norwood, MA, USA, 2010.
- Höflinger, F.; Müller, J.; Zhang, R.; Reindl, L.M.; Burgard, W. “A wireless micro inertial measurement unit (IMU).” *IEEE Trans. Instrum. Meas.* 2013, 62, 2583–2595.
- Kok, M.; Hol, J.D.; Schön, T.B.; Gustafsson, F.; Luinge, H. “Calibration of a magnetometer in combination with inertial sensors.” In *Proceedings of the 2012 15th International Conference on Information Fusion*, Singapore, 9–12 July 2012. *Sensors* 2020, 20, 80329 of 29
- Papafotis, K.; Sotiriadis, P.P. “MAG. IC AL.–A Unified Methodology for Magnetic and Inertial Sensors Calibration and Alignment.” *IEEE Sens. J.* 2019, 19, 8241–8251.
- Sarcevic, P.; Kincses, Z.; Pletl, S. “Online human movement classification using wrist-worn wireless sensors.” *J. Ambient Intell. Humaniz. Comput.* 2019, 10, 89–106.
- Shuster, M.D.; Oh, S.D. “Three-axis attitude determination from vector observations.” *J. Guidance Control* 1981, 4, 70–77.

- Odry, Á.; Fullér, R.; Rudas, I.J.; Odry, P. “Kalman filter for mobile-robot attitude estimation: Novel optimized and adaptive solutions.” *Mech. Syst. Signal Process.* 2018, 110, 569–589.
- Wen, X.; Liu, C.; Huang, Z.; Su, S.; Guo, X.; Zuo, Z.; Qu, H. “A First-Order Differential Data Processing Method for Accuracy Improvement of Complementary Filtering in Micro-UAV Attitude Estimation.” *Sensors* 2019, 19, 1340.
- Markley, F.L.; Crassidis, J.L. “Fundamentals of Spacecraft Attitude Determination and Control” Springer: New York, NY, USA, 2014; Volume 33.
- Nowicki, M.; Wietrzykowski, J.; Skrzypczynski, P. “Simplicity or flexibility? Complementary Filter vs. EKF for orientation estimation on mobile devices.” In *Proceedings of the 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, Gdynia, Poland, 24–26 June 2015.
- Csík, D.; Odry Á.; Sarcevic P. “Comparison of RSSI-Based Fingerprinting Methods for Indoor Localization”; *SISY 2022 • IEEE 20th Jubilee International Symposium on Intelligent Systems and Informatics*, Subotica, Serbia, 2022
- Bonnet S.; Bassompierre C.; Godin C.; Lesecq S.; Barraud A. “Calibration methods for inertial and magnetic sensors; *Sensors and Actuators*” A: Physical: 2009 Volume 156, Issue 2
- Thanh, Ha.; Hong S. K. “Quadcopter Robust Adaptive Second Order Sliding Mode Control Based on PID Sliding Surface.” *IEEE Access*. PP. 1-1. 2018
- Odry, A., Kecskes, I., Csik, D., Hashim, A. H., Sarcevic, P. “Adaptive Gradient-Descent Extended Kalman Filter for Pose Estimation of Mobile Robots with Sparse Reference Signals.” *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, pp. 4010-4017. 2022
- Odry, A., Kecskes, I., Csik, D., Rodríguez-Reséndiz, J., Carbone, G., Sarcevic P. “Performance Evaluation of Mobile Robot Pose Estimation in MARG-Driven EKF.” *The International Conference of IFToMM ITALY: Advances in Italian Mechanism Science*, pp. 685–692.
- Magony, G. “Quadcopter irányítására alkalmas rendszer tervezése és megvalósítása szenzorfüzió segítségével” 2020
- MPU-9250 “Product Specification Revision 1.1“ InvenSense Inc., 2016

- S. Wang, A. Waadt, A. Burnic, D. Xu, C. Kocks, G. H. Bruck, P. Jung, “System implementation study on RSSI based positioning in UWB networks,” International Symposium on Wireless Communication Systems (ISWCS), 2010, pp. 36-40
- Chitta, S., “MoveIt!: An Introduction.” Koubaa, A. (eds) Robot Operating System (ROS). Studies in Computational Intelligence, vol 625. Springer, Cham., 2016

KÖSZÖNETNYÍLVÁNÍTÁS

Szeretnék köszönetet mondani mindazoknak, akik segítettek a munkámat, valamint ötleteket adtak. Külön köszönet Dr. Odry Ákos, valamint Dr. Sarcevic Péter tanár uraknak, hogy elláttak tanácsokkal és, hogy gyorsan reagáltak szükség esetén. Köszönöm a barátaimnak és a szaktársaimnak, hogy bíztattak a dolgozat megírása alatt.