BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

GÉPÉSZMÉRNÖKI KAR

MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK

Kreinicker Gábor

NANOMŰHOLDAK ORIENTÁCIÓ MEGHATÁROZÁSA

Témavezető:

Pál András
???

Konzulens:

Dr. Nagy Balázs Vince
egyetemi docens

Budapest, 2022.

Tartalomjegyzék

1. Bevezetés **1**

2. Hardver **2**

3. Szimuláció **8**

3.1. Szoftveres szimulálás 8

3.2. Hardveres szimulálás 10

4. Képfeldolgozás **14**

4.1. Nap-keresés 14

4.1.1. Iterációs megoldás 14

4.1.2. Subpixel 15

4.1.3. Szomszédos pixelek módszere 15

4.2. Föld-keresés 16

4.2.1. Intervallumfelező módszer 16

4.2.2. Keretmódszer 17

5. Adatfeldolgozás **19**

5.1. Kvaterniógenerálás 20

5.2. Bázistranszformáció 21

6. Összefoglalás **23**

Irodalomjegyzék **24**

1. fejezet

Bevezetés

A Csillagászati és Földtudományi Kutatóközpont (CSFK) Konkoly-Thege Miklós Csillagászati Intézetével együttműködve célunk egy infraszenzoros műhold orientáció meghatározó rendszer fejlesztése.

Kisméretű, úgynevezett nanoműholdak[1] esetében jelenleg még számos nyitott kérdést tartogat az orientációnak – azaz a műhold térbeli helyzetének – nemcsak a stabilitása, hanem a meghatározása is. Ugyanis ezek Föld körüli pályára állásuktól kezdve általában irányíthatatlan módon forognak, mely megnehezíti a tudományos mérések végzését vagy akár az adatok Föld-irányú továbbítását.

Ahhoz, hogy stabilizálni tudjuk egy műhold helyzetét, ismernünk kell annak megváltozását is. Ezt tehetnénk magnetométerrel, amely a Föld mágneses teréhez képest határozza meg az orientációt. Azonban ez önmagában nem elegendő, ugyanis ezzel a három szabadsági fokból csak kettőt tudunk meghatározni, amely nem definiálja egyértelműen az orientációt. Egy olyan megoldáson dolgozunk, amely képfeldolgozást alapul véve képes meghatározni a műhold térbeli helyzetét.

2. fejezet

Hardver

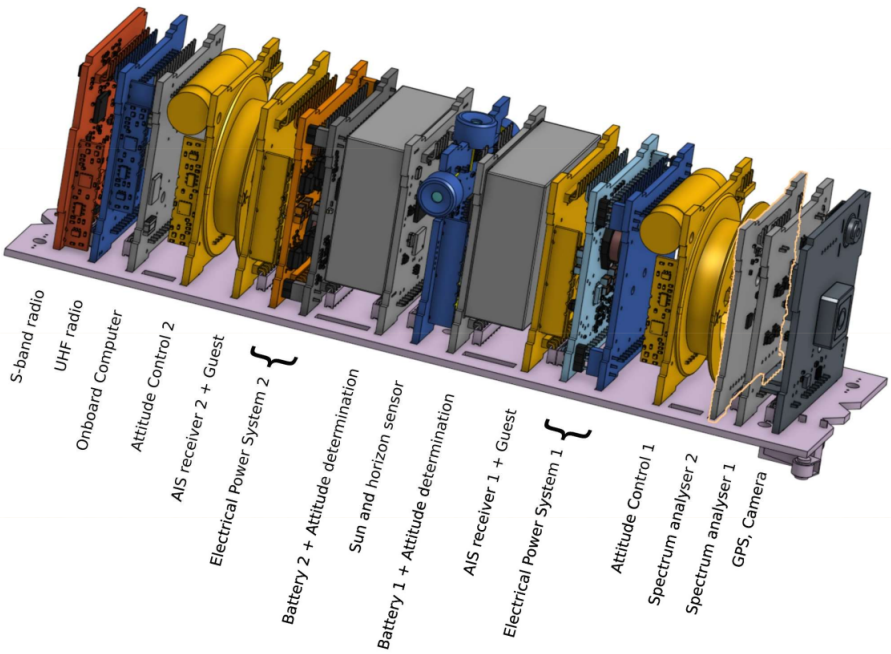
Jelenleg az MRC-100 nevű, BME-s fejlesztésű műholdra készítjük az első, infra szenzoros orientáció-meghatározó rendszerünket. Ez a rendszer lesz felelős azért, hogy orientáció-adatokkal lássa el a műholdat, amely ennek köszönhetően a benne megfelelően elhelyezett tekercsekkel stabilizálni, módosítani tudja térbeli helyzetét.

Az általunk fejlesztett áramkör a műhold közepén fog elhelyezkedni (1. ábra). Itt négy darab infra kamera kapott helyet, egymással 90°-os szöget bezárva, párhuzamosan a műhold oldalaival, ezzel maximalizálva a belátott égboltot. Ez az áramkör több részre bontható, melyek közül azokat emelném ki, amelyekkel tüzetesebben foglalkoztam.

Maguk az MLX90641 [3] típusú szenzorok távoli infratartományban érzékelnek. Felbontásuk csekély, 16×12 pixel, azonban hatalmas a látószögük, $110 \times 75°$, így nagy a lefedettségük. Ezen tulajdonságainak és kis fogyasztásuknak köszönhetően ideálisnak találtuk őket a kitűzött cél elérésére.

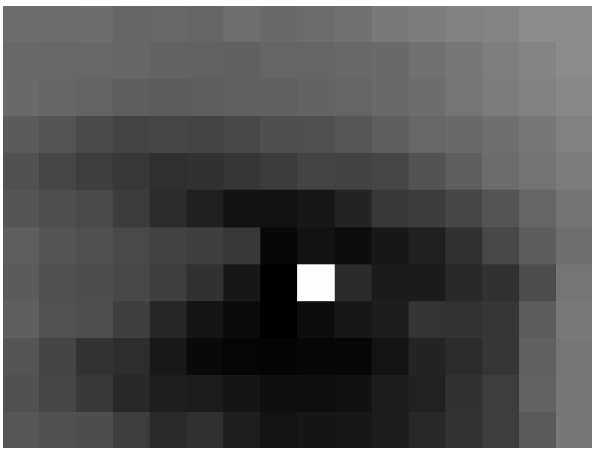
Teszteltünk ugyanebbe a családba tartozó MLX90640 szenzort is, amely felbontása 32x24 pixel. Ezek is jól működtek, azonban energiaigényük miatt maradtunk a kisebb felbontású érzékelők mellett.

A szenzorok olvasására STM32F072 [4] típusú mikrokontrollert alkalmazunk, mely már bizonyított a GRBAlpha [5] műholdon, űrbéli alkalmazások terén. Ennek szintén alacsony energiaigénye, valamint nagyobb tranzisztormérete - mely a különböző sugárzások ellen biztosít nagyobb védelmet - ad igazán kedvező lehetőségeket. Ennek azonban az az ára, hogy jóval kisebb erőforrás áll rendelkezésre.



2.1. ábra. Az MRC-100 műhold felépítése[2]

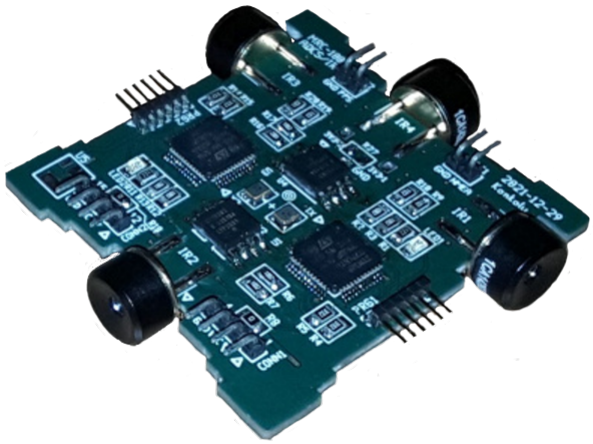
2.2. ábra. Az alkalmazott MLX90641 infraszenzorok[3]



3. ábra Egy szenzorral készített szabadtéri kép a Napról

Így az egyik legnagyobb feladat az optimális szoftver elkészítése, hogy minden a lehető legjobban üzemeljen. Ennek az áramkörnek ugyanis a következő feladatokat kell ellátnia:

1. Infraszenzorok olvasása:
   Ezek a kamerák I^2C protokollt alkalmazva kommunikálnak a mikrovezérlővel. Egy mikrovezérlő egyszerre két szenzort tud maximális sebességgel olvasni, így két mikrovezérlőt alkalmazunk. Közvetlen memóriahozzáférést alkalmazva (Direct Memory Access, DMA) sikerült 16 Hz-en tartósan üzemeltetni ezeket, amely biztosan elegendő, figyelembe véve a lehetséges maximális szögsebességeket.

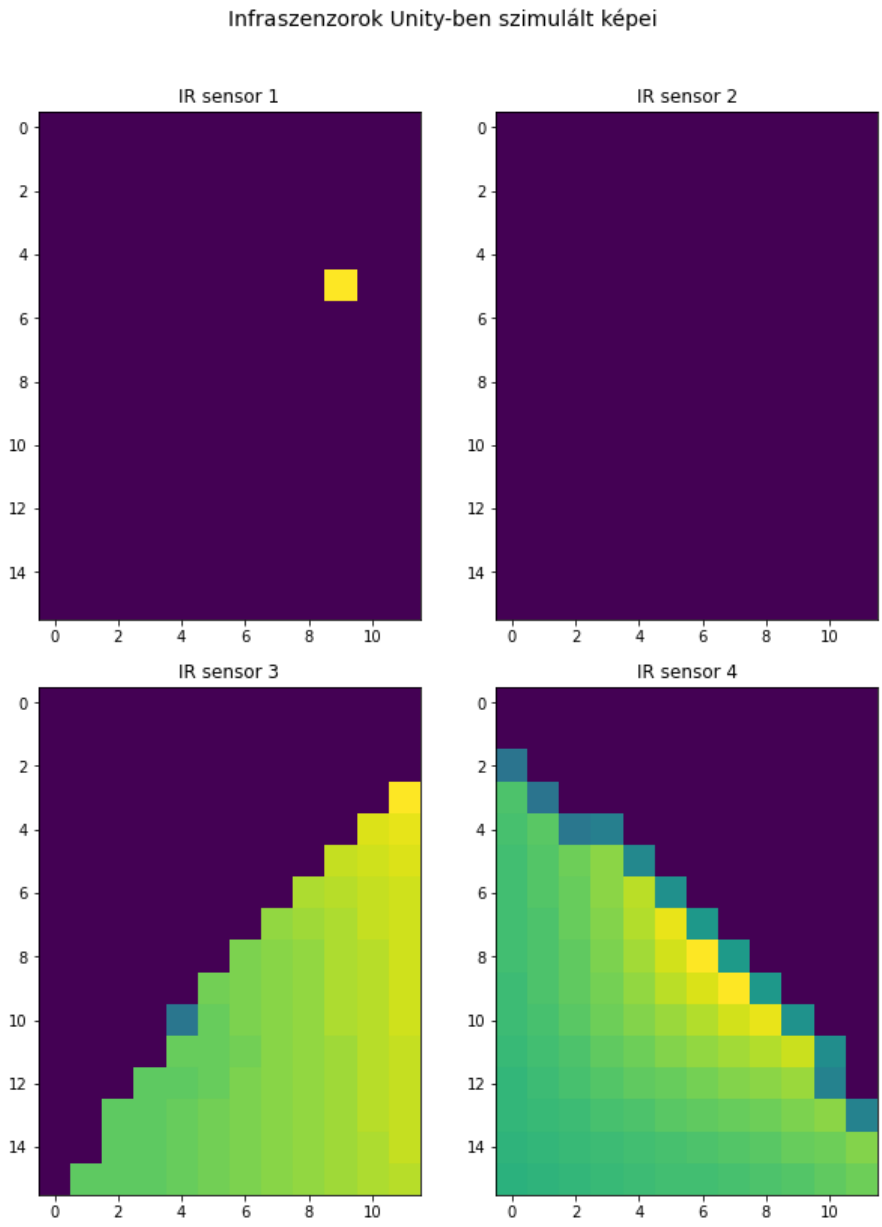


4. ábra Az általunk fejlesztett egység egyik prototípusa

2. Beolvasott 16×12 pixeles kép feldolgozása, Föld és (amennyiben látható) Nap detektálása. A következő lehetőségek fordulhatnak elő egy szenzor képein:

   - A kamera semmit sem lát:
     Ekkor semmit nem tudunk felhasználni, a képen csak termikus zaj jelenik meg. Szimulációink szerint olyan lehetőség azonban nem fordulhat elő, hogy egyik szenzor se lásson semmit, ugyanis a pályamagasság kellően alacsony, és a látószög kellően nagy ahhoz, hogy valamelyik kamerán minden esetben látszódjon a Föld.
   - A kamera képén látszik a Föld:
     Ekkor meg tudjuk határozni a Föld normálisát a detektálószenzor koordinátarendszerében.

- A kamera képén látszik a Nap:
  Ekkor meg tudjuk határozni a szenzor-Nap vektort. A szimulációink alapján egy keringési periódus ($\approx$ 90 perc) során az esetek körülbelül 30%-ában nem látható a Nap, ugyanis azt a Föld kitakarja. Ez ellen sajnos nem tudunk mit tenni, azonban megfelelő idő után ezekben az esetekben is jól meg tudjuk becsülni az orientációt korábbi adatok alapján.
- A kamera képén látszik a Föld és a Nap is:
  Ekkor mindkét vektor meghatározható



5. ábra Saját szimulációban előállított képek

3. Orientáció meghatározása
   Ez az egyik legkritikusabb része az egész rendszernek. A négy szenzor által detektált objektumokból kapott vektorokat kell valamilyen formában összefüggésbe hozni a műhold orientációjával. Egyrészt a szenzorok saját koordinátarendszerében meghatározott vektorokat kell áttranszformálni a műhold saját koordináta rendszerébe. Ebből lesz egy térbeli pozíciónk, azonban ez még határozatlan, ugyanis csak a műhold-Föld- és műhold-Nap-vektorok ismertek.
   Ezért alkalmazzuk a TLE-adatokból[6] nyert Föld-műhold- és Föld-Nap-vektorokat, amelyből at utóbbi közelíthető a műhold-Nap-vektorral, ugyanis a Föld-műhold távolság elhanyagolhatónak tekinthető a Föld-Nap távolsághoz képest.
   Ezeknek köszönhetően már határozottá válik a rendszer, és ismertté válik az orientáció.

4. Orientációadatok továbbítása a műhold további rendszerei felé.
   A feladat ezen részével a dolgozat beadásáig nem foglalkoztam.

3. fejezet

Szimuláció

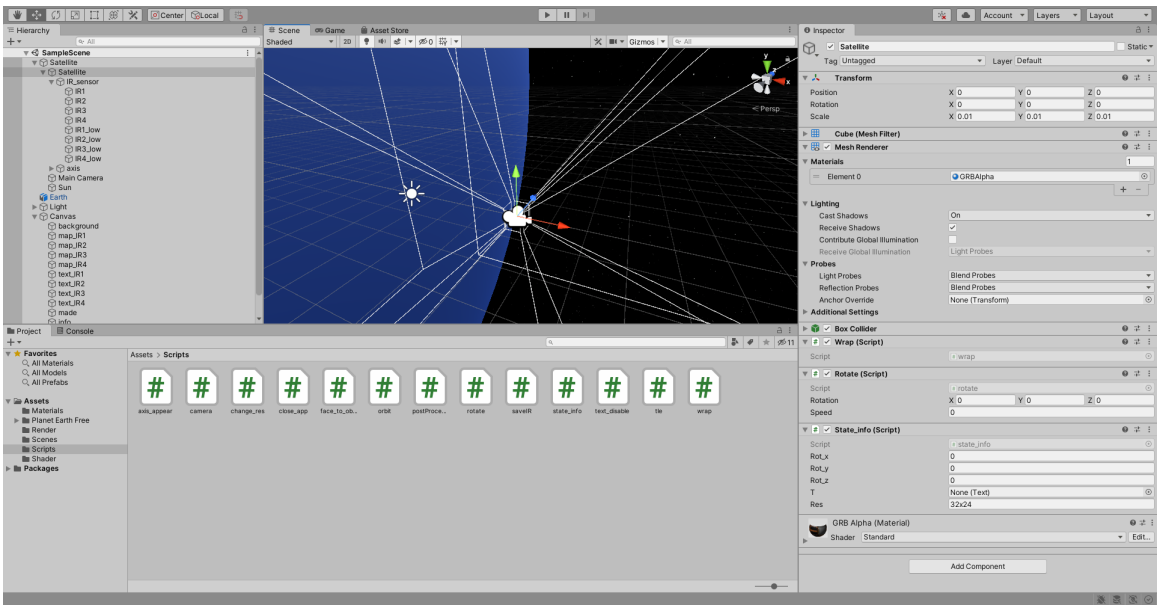
Ahhoz, hogy tesztelni tudjuk a képfelismerő algoritmusainkat, szükségünk van adatokra, azonban valódi, űrben készült képek nem állnak rendelkezéseinkre.

Mivel ismerjük a műhold pályamagasságát (TLE-adatokból), illetve az infrakamera különböző paramétereit, mint a felbontás és látószög, így meg tudjuk becsülni egy, a műholdon készült kép jellegét. Ennek megfelelően két fő hőforrás van, amelyet a műhold szenzorai egyértelműen érzékelni tudnak: a Föld, mely egy összefüggő, enyhén fényes felületként, illetve a Nap, amely egy nagyon fényes pontként jelenik meg a kapott képen. Ezeknek az ismeretében több módszert is kipróbáltunk egy űrben készült kép imitálására.

3.1. Szoftveres szimulálás

Egyik megoldás a szoftveres képgenerálás. Próbáltunk python nyelven megírt programokkal képeket generálni, azonban különböző mozgásokat ezzel szimulálni nem tudtunk.

Ezért a Unity[7], elsősorban játékfejlesztő platform felé fordultunk. Ebben lehetőség van tárgyakat elhelyezni, egyszerűen mozgatni és különböző kameraszögeket megjeleníteni. Miután létrehoztuk a Föld és a műhold méretarányos statikus modelljét, elkészítettük a műhold orientációmódosító programkódját.

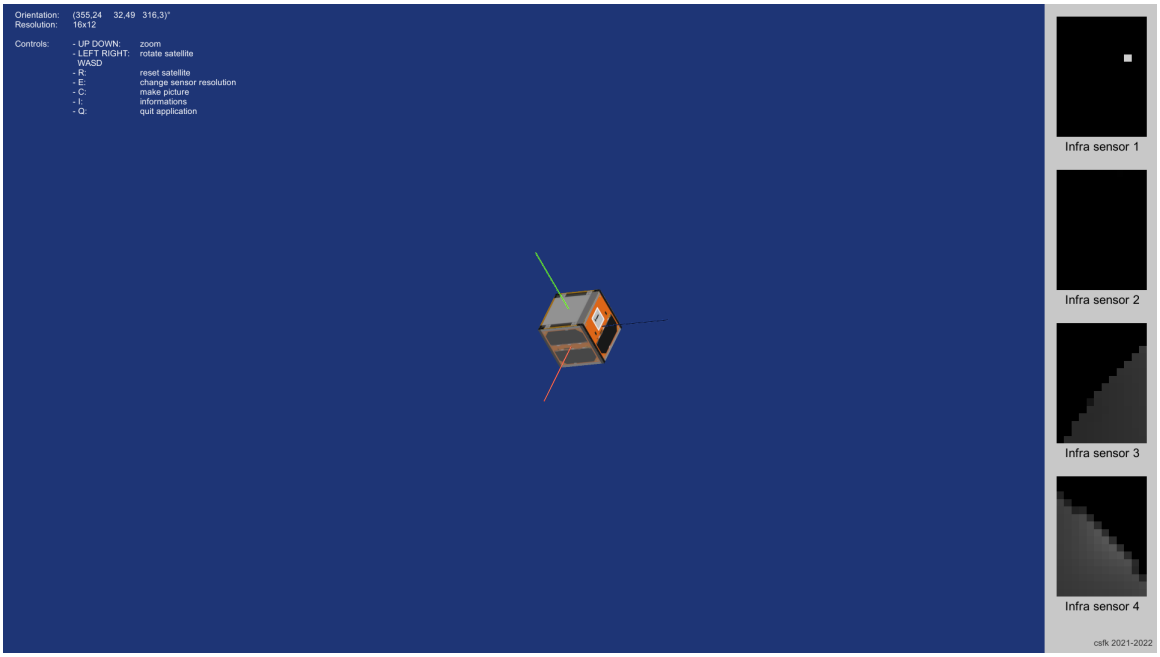


6. ábra Unity kezelőfelület

Ezután elhelyeztük a műholdmodellen a négy kamerát a tervezett elrendezéseknek megfelelően. Beállítottuk ezek látószögét és felbontását, illetve mivel ezen a platformon nem tudunk infrakamerát szimulálni, így monokróm kamerát alkalmaztunk. A Föld illetve a Nap textúráit egyszerű színekre változtattuk úgy, hogy fekete-fehér képen a Nap jóval fényesebbnek tűnjön.

A kamerák által látott képeket fájlba mentjük úgy, hogy a képfrissítés megegyezzen a valódi kamerák képfrissítésével. Ezeket a fájlokat pedig valós időben tudjuk feldolgozni egy python programmal.

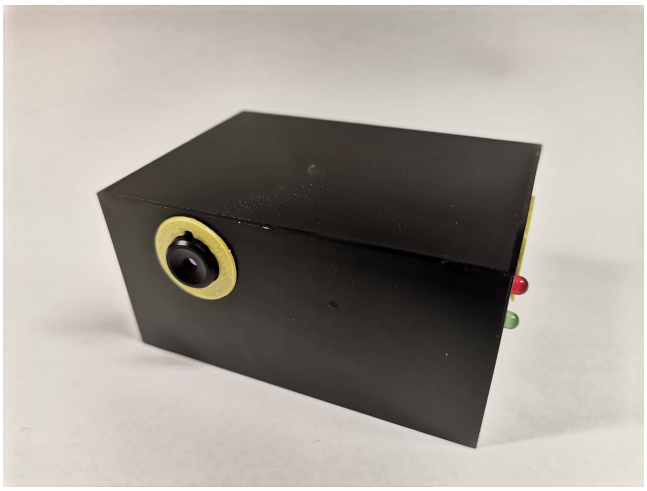
A Nap Földhöz képest vett pozícióját TLE-adatokból ismerjük, így a szimulációban ezt is pontosan meg tudjuk jeleníteni. Ennek a haszna elsősorban a szimuláció pontosságának ellenőrzése más, már a Föld körül keringő műholdak TLE-adatai alapján, illetve a felbocsátás után nyerhetünk képet a műhold kamerái által látottakról. Ezeknek megfelelően billentyűkkel tudjuk a műholdat forgatni, illetve a kamerák felbontásán is tudunk változtatni.



7. ábra A szimuláció futás közben

3.2. Hardveres szimulálás

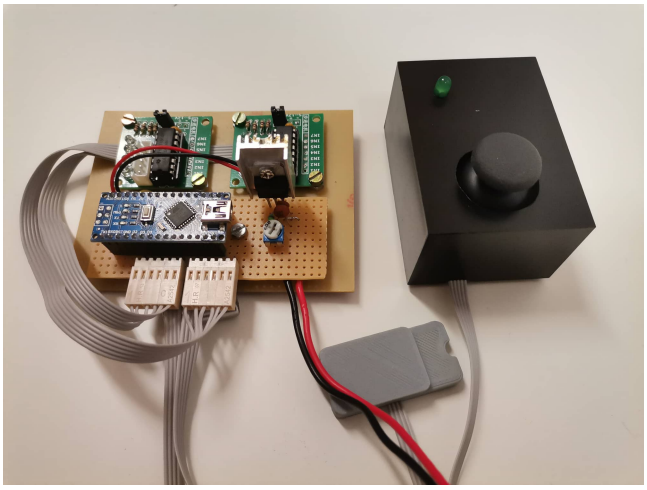
Másik megoldás a műholdra elhelyezni tervezett infrakamerákkal megegyező szenzorokkal képeket készíteni a Napról.



8. ábra A szabadtéri képek készítésére összerakott eszköz

Eleinte egy szabadban elhelyezett kamerával kezdtünk el képeket gyűjteni. Ezeknek a képeknek számos problémája van: a környezet hősugárzása is megjelenik a képen, ami a világűrben nem lesz jelen, így ez megnehezíti a képfeldolgozást, másrészt míg a Föld egyszeri körbefordulása 24 óra, addig a műhold keringési periódusa csak körülbelül 90 perc.

Második problémára a megoldás azt jelentette, hogy a kamerát mozgatjuk. Ahhoz, hogy ezt megfelelően tudjuk végrehajtani, szükség volt egy olyan eszközre, ami előre beprogramozott módon tudja forgatni a szenzort.



9. ábra A forgatórendszer vezérlő áramköre

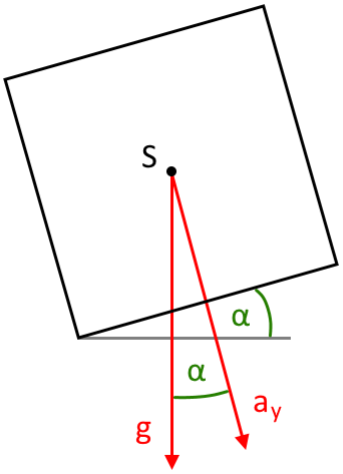
Ez az eszköz el is készült, egy Arduino Nano [8] vezérlésű, két szabadsági fokú állvány. A forgatásért két 28byj típusú léptetőmotor [9] felel. Van lehetőség előre beprogramozott, de akár manuális forgatásokra is egy joystick segítségével. A vízszintessel bezárt szöget egy MPU-9250 [10] IMU szenzor gyorsulásadataiból ismerjük: a függőleges irányú gyorsulás és az ismert nehézségi gyorsulás alapján meghatározható:

$$\alpha = \arccos \frac{a_y}{g} \tag{3.1}$$

Ahol:

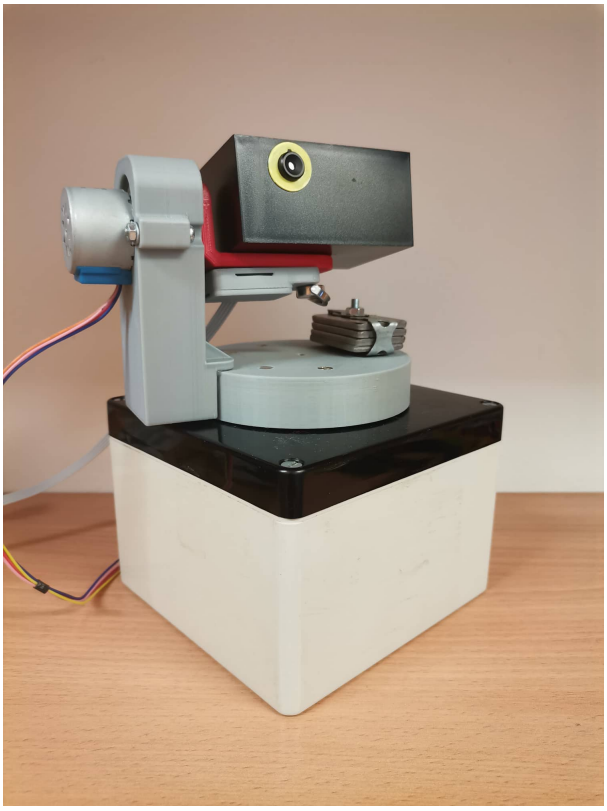
a_y : mért y-irányú gyorsuláskomponens

g : nehézségi gyorsulás $\left(\approx 9.81 \frac{\mathrm{m}}{\mathrm{s}^2}\right)$



10. ábra Az emelkedési szög visszaszámolása

A finom mozgás elérésére megfelelő mechanikát kellett elkészíteni. A függőleges irányú forgatásért egy, a végén és középen csapágyazott kar felel. A vízszintes eset valamivel bonyolultabb, ugyanis nagyobb, a forgás irányával párhuzamos terhelés éri a rendszert, így görgős csapágyat kellett alkalmazni. Költséghatékonyság céljából azonban megfelelően elhelyezett görgőket alkalmaztunk.



11. ábra A forgatórendszer és a szenzor összeállítva

Az első problémára a megoldást az jelentette, hogy elszigeteltük magunkat a környezettől. Levittük a kamerát egy föld alatti, egy ajtót leszámítva nyílászárómentes terembe, amiben ezáltal közel állandó hőmérséklet uralkodott. Azonban így a Nap nem volt látható. Erre volt megoldás egy forrasztópáka felhevített hegye, amely megfelelő távolságból nagyon hasonló képet mutatott, mint a Nap. A mozgás szimulálására pedig ugyanazt a 2 szabadsági fokú eszközt használtuk.

4. fejezet

Képfeldolgozás

A projekt egy sarkalatos pontja a képfeldoldozás. Mivel nincs lehetőségünk a teljes képet leküldeni a Földre, így ezt kis erőforrásokkal kell megtennünk a fedélzeten. Éppen ezért fontos, hogy a lehető legjobb algoritmust használjuk.

A korábban említettek alapján két objektumot kell detektálnunk: Földet és Napot. Ennek megfelelően két részre bontottuk a képfeldolgozást is: a Nap-keresésre és a Föld-keresésre.

4.1. Nap-keresés

A Nap-keresés sok szempontból okoz nehézséget. Egyrészt az esetek egyharmadában a műholdhoz képest a Föld mögött található, másrészt pedig előfordulhat, hogy kívül esik a szenzorok által lefedett térrészen, így nem látjuk. Előbbit TLE-adatokból előre meg tudjuk jósolni, utóbbit viszont már csak az orientáció ismeretében. Ha azonban valamely szenzoron megjelenik, úgy egy fényes pontot tapasztalnánk.

4.1.1. Iterációs megoldás

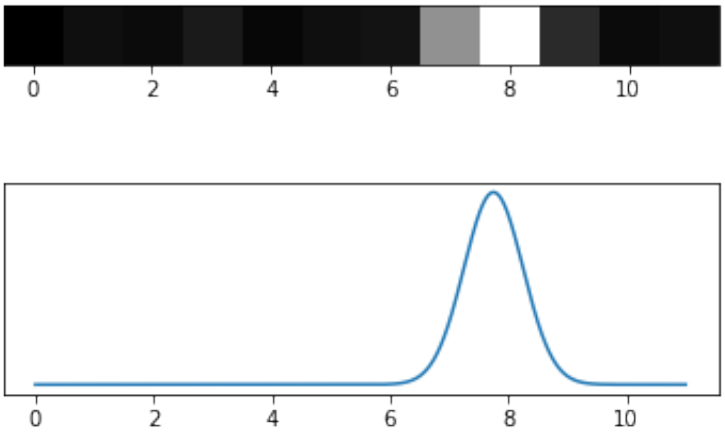
A legegyszerűbb ám legpontatlanabb megoldás, hogy megvizsgáljuk az összes pixel értékét, és amennyiben valamelyik megugrik egy empirikusan előállított küszöbértéket, úgy azt Napként kezeljük.

A gond akkor kezdődik, amikor a Napból érkező sugárzás nem egy, hanem több pixelen oszlik meg. Ekkor előfordulhat, hogy egyik pixel értéke sem lépi át a küszöbértéket, ha azonban mégis, akkor is adódhat pontatlan eredmény.

4.1.2. Subpixel

Alacsony felbontás esetén tudjuk alkalmazni az úgynevezett subpixeleljárást[11]. Ennek a lényege, hogy a kép felbonásánál nagyobb pontossággal meg tudjuk határozni a detektálandó objektum képen vett pozícióját.

A mi esetünkben ez tovább egyszerűsödik, ugyanis ismerjük a pixel értékét abban az esetben, amikor az összes Napból érkező sugárzás csak ezt érinti. Azaz egy függvény illesztésével nagy pontossággal meghatározható a Nap képbeli helyzete.

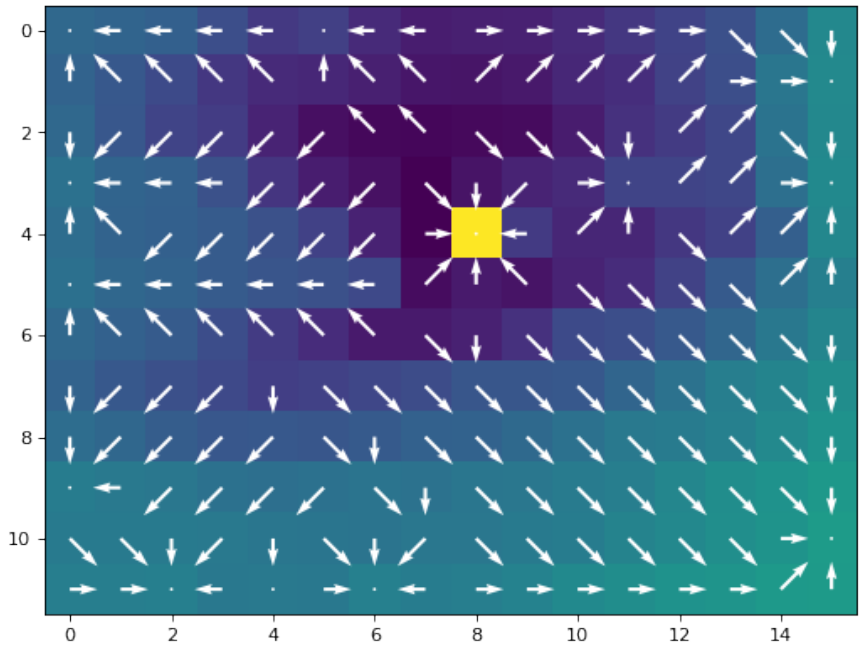


12. ábra A subpixel eljárás alkalmazása a Napon

4.1.3. Szomszédos pixelek módszere

Kipróbáltunk egy módszert, ahol minden pixelhez hozzárendelünk egy vektort, amely a szomszédos pixelek közül (beleértve a sarkuknál érintkező szomszédokat) a legfényesebbre mutat, ezzel kapva egy, az iteratív módszernél precízebb megoldást.

Ezt a módszert a fotózott képeken teszteltük. Azt tapasztaltuk, hogy közvetlen a Nap körüli pixeleket leszámítva nincs rendszer a vektorok irányában. Ebben az esetben viszont sokkal hatékonyabb ennél a módszernél az iteratív módszer.



13. ábra A szomszédos pixel módszer

4.2. Föld-keresés

A Föld-keresés sem egyszerű probléma, azonban a Nappal ellentétben lényeges kiterjedése van a képeken (azaz megjelenése jelentősen nagyobb, mint két pixel), illetve mindig van olyan kamera, amely detektálni tudja.

Éppen ezért lehetőségünk van a napkereső algoritmusoknál hatékonyabb eljárásokat kipróbálni. Ezidáig két módszert teszteltünk: az intervallumfelező módszert és egy ún. keretmódszert.

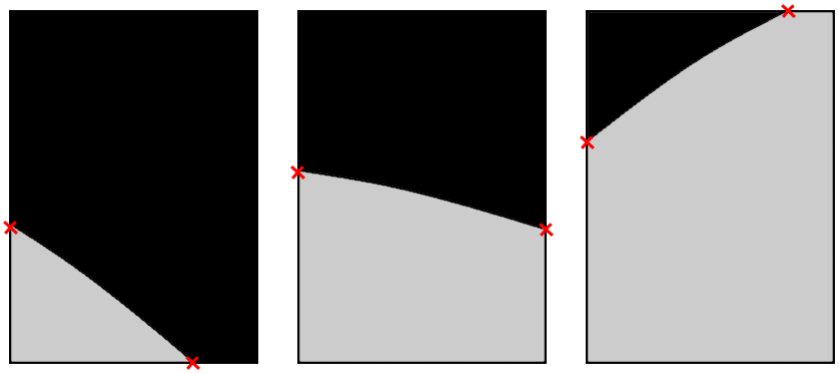
4.2.1. Intervallumfelező módszer

Ez a módszer két részre bontható: egyrészt meg kell határoznunk, hogy a kép mely éleit érinti a Föld, azaz melyek azok az élek, amelyeket a Föld felszín elmetsz. Ezek után pedig intervallum felezéssel közelítjük ezeket a metszéspontokat, majd meghatározzuk az adott szenzorhoz tartozó Föld normálist.

Megvizsgáljuk, hogy a képen melyek azok a sarkok, ahol a Föld sugárzásának megfelelő küszöbértéket átlépi az adott pixel. Ennek megfelelően öt lehetőség adódhat, melyből kettő nem tartalmaz számunkra információt: ezek amikor egyik sarok sem lépi át a küszöbértéket, ekkor ugyanis a Föld méreténél fogva nem található a képen,

illetve amikor mind a négy sarka átlépi ezt az értéket, ekkor pedig az egész képen a Föld található, így nem tudunk normálist meghatározni.

A további három lehetőség esetén mindig van kettő él, amelyet elmetsz a Föld felszíne. Ennek a módszernek köszönhetően ismerjük ezeket az éleket, ezáltal pedig intervallum felezéssel meghatározható a két metszéspont.



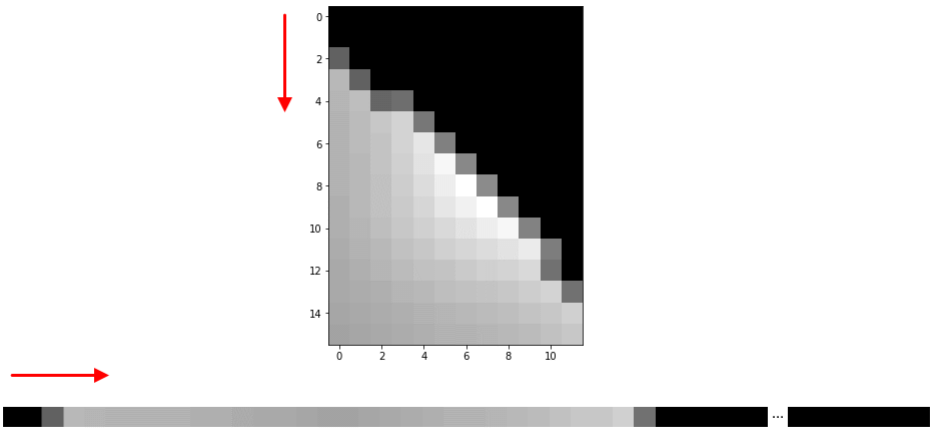
14. ábra Sematikus ábra a Föld lehetséges megjelenéseiről a szenzorok képein

A metszéspontok képen vett koordinátáit kivonva egymásból, majd elforgatva 90°-kal a megfelelő irányba megkapjuk a Föld normálisát. A forgatás irányát az érintett élek számából meg tudjuk határozni.

4.2.2. Keretmódszer

Ez a módszer egy lényegesen egyszerűbb módszer a lépésszámot tekintve. Ahogy az elnevezés is árulkodik, a képnek a kép szélén elhelyezkedő pixeleit használjuk fel.

Körbehaladva a képkereten, a pixelek értékeit egy listába rendezzük úgy, hogy ismerjük az érték képen vett pozícióját. Ezután végignézzük a listát, hogy melyek azok a pixelek, amelyek átlépik a küszöbértéket. Az így kialakult szakasz kezdő és végpontja megadja a metszéspontokat. Innentől pedig a normális meghatározás az előzőekhez hasonlóan meghatározható.



15. ábra A keret módszer

Ami még a képfeldolgozásban nehézséget fog okozni, az a Nap és Földkereső algoritmusok egyesítése. Ugyanis ezt is a lehető legoptimálisabban kell megtegyük, mivel a módszerek igen különbözőek is lehetnek a végleges verzióban alkalmazott rendszerben.

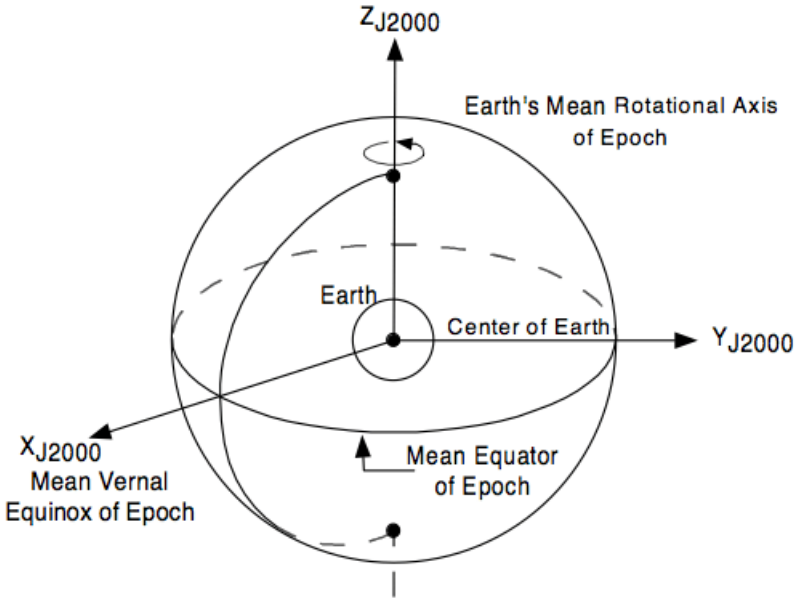
A kutatás képfeldolgozás szempontjából még igen korai fázisban van. A korábban ismertetett képfeldolgozó algoritmusokat, módszereket teszteltük, hogy megtaláljuk a céljainknak legmegfelelőbbet. Az eddig kipróbáltakon felül természetesen még rengeteg módszer van, azonban a téma bemutatásáig eddig sikerült eljutnunk.

5. fejezet

Adatfeldolgozás

A műhold TLE adatait a CelesTrak [12] oldaláról le tudjuk tölteni, amely alapján sok fontos paraméterhez férünk hozzá. Ami az orientáció meghatározása szempontjából a legfontosabb az a Föld-Nap és a Föld-műhold vektorok. Ezek a J2000 koordinátarendszerben [13] vannak megadva.

A J2000 egy Föld központú ortogonális koordinátarendszer, ahol az x-tengely a 2000. január 1. UTC 12:00 órakor a Föld középpontját és a tavaszpontot összekötő szakasszal, illetve a z-tengely a Föld forgástengelyével esik egybe, az y-tengely pedig a jobbsodrásnak megfelelően adódik.



16. ábra A J2000 koordináta rendszer

5.1. Kvaterniógenerálás

A műholdnak is van egy saját koordináta rendszere, melyben meg tudjuk határozni a műhold-föld és műhold-Nap vektorokat az infra szenzorok adatainak feldolgozása után. Ez a rendszer is ortogonális, így ha ismerjük a forgatás mátrixot, amivel áttérhetünk egyik rendszerből a másikba, akkor a műhold orientációját is meg tudjuk határozni.

Ahhoz, hogy tesztelni tudjuk az adatfeldolgozó rendszerünket, ezt a forgatásmátrixot véletlenszerűen generáljuk, ezzel szimulálva a képfeldolgozásból kapott adatokat. Ez azonban nem egy triviális probléma, ugyanis a mátrix elemei összefüggéseknek megfelelően adódnak.

Megoldást jelenthet a következő kifejezés [14], amelynek köszönhetően a következő paramétereket tudjuk generálni:

$$q_r = \sqrt{1-u_1}\sin(2\pi u_2) \tag{5.1}$$

$$q_i = \sqrt{1-u_1}\cos(2\pi u_2) \tag{5.2}$$

$$q_j = \sqrt{u_1}\sin(2\pi u_3) \tag{5.3}$$

$$q_k = \sqrt{u_1}\cos(2\pi u_3) \tag{5.4}$$

Ahol: $u_1, u_2, u_3 \in [0,1]$

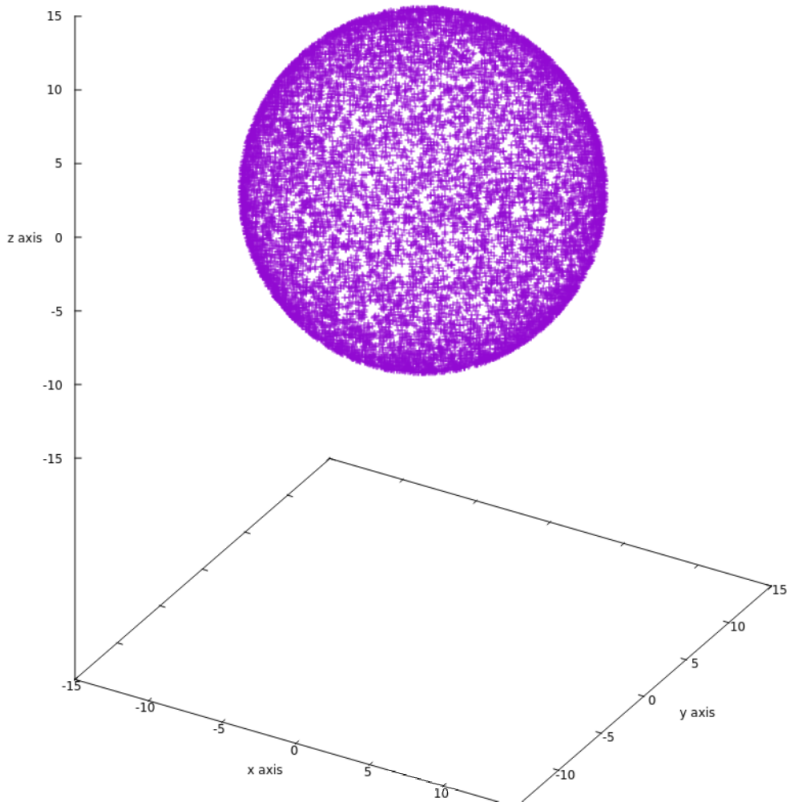
Ekkor megkapjuk a h kvaterniót:

$$h = \left(q_r, q_i, q_j, q_k\right) \tag{5.5}$$

Ezt pedig fel tudjuk használni **R** forgatásmátrix meghatározásához a következő összefüggés [15] alapján:

$$\mathbf{R} = \begin{bmatrix} 1-2(q_j^2+q_k^2) & 2(q_iq_j-q_kq_r) & 2(q_iq_k+q_jq_r) \\ 2(q_iq_j+q_kq_r) & 1-2(q_i^2+q_k^2) & 2(q_jq_k-q_iq_r) \\ 2(q_iq_k-q_jq_r) & 2(q_jq_k+q_iq_r) & 1-2(q_i^2+q_j^2) \end{bmatrix} \tag{5.6}$$

Ennek a generálásnak a megfelelő működését és homogén eloszlását tudjuk ellenőrizni a következő módszerrel: veszünk egy három dimenziós vektort, majd hattatjuk rá a random forgatás mátrixot. Ugyanezzel a vektorral megismételjük ezt a műveletet, minden alkalommal új mátrix generálásával. Akkor mondhatjuk, hogy a random forgatásmátrix generáló megfelelő, ha a vektorok által meghatározott pontfelhő egy gömbfelület, és ezeknek a pontok közel állandó távolságra vannak egymástól, mely a mi esetünkben jól láthatóan teljesül.



17. ábra Generált kvaterniók homogenitásának ellenőrzése

5.2. Bázistranszformáció

Tegyük fel, hogy meghatároztuk a műhold-Föld és műhold-Nap vektorokat a műholdon, illetve TLE adatokból ismerjük a Föld műhold és Föld-Nap vektorokat. Korábban tárgyaltak alapján a Föld-Nap vektor közelíthető a Föld-műhold vektorral, így TLE adatokból származó egységvektorok és a műholdon mért egységvektorok, és a közbezárt szögük is megegyeznek.

Így azonban amit meg kell határoznunk az egy forgatás mátrix, amely ismeretében meg tudjuk határozni a műhold orientációját. Ezzel a dolgozat leadásáig nem végeztünk, de az eddig elért részeredményeket a következőkben tárgyalom.

A keresett forgatásmátrix legyen $\mathbf{M}$, illetve legyen $e^{(sat)}$ a műhold-Föld egységvektor a műhold koordinátarendszerében, $e^{(J2000)}$ pedig ugyanez az egységvektor a Föld J2000-es koordinátarendszerében. Az s műhold-Nap egységvektorok esetén ugyanez a jelölés a mérvadó. A bázistranszformációt meghatározó összefüggések:

$$e^{(sat)} = \mathbf{M}^T \cdot e^{(J2000)} \tag{5.7}$$

$$s^{(sat)} = \mathbf{M}^T \cdot s^{(J2000)} \tag{5.8}$$

Továbbá mivel a vektorkettősök bezárt szöge állandó, így a következő összefüggést is felírhatjuk:

$$e^{(J2000)} \cdot s^{(J2000)} = e^{(sat)} \cdot s^{(sat)} \tag{5.9}$$

A J2000-es koordinátarendszer előre definiált és ortogonális. Ezért ezt létre kell hozzuk a műhold esetén is. Ennek úgy fogunk neki, hogy kijelölünk egy irányt, ez legyen $e_x^{(sat)}$. Ekkor a bázis másik két komponense:

$$e_y^{(sat)} = \frac{s^{(sat)} \times e^{(sat)}}{\left| s^{(sat)} \times e^{(sat)} \right|} \tag{5.10}$$

$$e_z^{(sat)} = \frac{e^{(sat)} \times e_y^{(sat)}}{\left| e^{(sat)} \times e_y^{(sat)} \right|} \tag{5.11}$$

Mely alapján a forgatásmátrixok:

$$\mathbf{M}^{(sat)} = \left(e_x^{(sat)}, e_y^{(sat)}, e_z^{(sat)} \right) \tag{5.12}$$

$$\mathbf{M}^{(J2000)} = \left(e_x^{(J2000)}, e_y^{(J2000)}, e_z^{(J2000)} \right) \tag{5.13}$$

Egy másik, hasonló esetekben alkalmazott módszer a Wahba probléma[16]. Ez egy kompakt összefüggést ad, mellyel szintén meghatározható a forgatásmátrix.

6. fejezet

Összefoglalás

Ez egy nagyon sok komponensű projekt, mely mostanáig is rengeteg bizonytalanságot hordoz magában. Ennek oka, hogy jelenleg is fut a kutatás, számos dolog még nincs tesztelve, illetve nem tisztázódott még minden a végleges formájában. Éppen ezért a dolgozat tartalma és megfogalmazása is néhol bizonytalan.

Éppen ezért ennek a dolgozatnak a célja a jelenleg is aktív kutatásunkban folyó tesztelések, ötletek és módszer alkalmazások ismertetése, egy olyan környezetben, amelynél elengedhetetlen fontosságú az optimalizálás. Rengeteg kiaknázatlan és érdekes probléma vár megoldásra még, melyet a következő hónapokban szeretnénk véglegesíteni. Az elsősorban műegyetemi fejlesztésű MRC-100 műhold egyik egységeként majd a jelen tervek szerint tavasszal kerül felbocsátásra, mely eredményeit izgatottan várjuk.

Irodalomjegyzék

[1] Nanoműhold. [Letöltve - 2022.11.07.].

[2] MRC-100. [Letöltve - 2022.11.07.].

[3] MLX90641 adatlap. [Letöltve - 2022.11.07.].

[4] STM32F072 adatlap. [Letöltve - 2022.11.07.].

[5] András Pál, Masanori Ohno, László Mészáros, Norbert Werner, Jakub Ripa, Marcel Frajt, Naoyoshi Hirade, Ján Hudec, Jakub Kapuš, Martin Koleda, et al. Grbalpha: A 1u cubesat mission for validating timing-based gamma-ray burst localization. In *Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray*, volume 11444, pages 825–833. SPIE, 2020.

[6] Emilian-Ionuţ Croitoru and Gheorghe Oancea. Satellite tracking using norad two-line element set format. *Scientific Research and Education in the Air Force-AFASES*, 1:423–431, 2016.

[7] Unity. [Letöltve - 2022.11.07.].

[8] Arduino Nano. [Letöltve - 2022.11.07.].

[9] 28byj léptető motor. [Letöltve - 2022.11.07.].

[10] MPU-9250. [Letöltve - 2022.11.07.].

[11] Qi Tian and Michael N Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35(2):220–233, 1986.

[12] TLE adatbázis. [Letöltve - 2022.11.07.].

[13] Vivek Vittaldev. The unified state model. derivation and applications in astrodynamics and navigation. pages 30–34, 2010.

[14] Random kvaternió generáló algoritmus. [Letöltve - 2022.11.07.].

[15] Drazen Svehla. Earth orientation quaternion. In *Geometrical Theory of Satellite Orbits and Gravity Field*, pages 355–361. Springer, 2018.

[16] Grace Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409–409, 1965. DOI: 10.1137/1007077.