Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Szeile Aliz

SZENZORHÁLÓZAT ALAPÚ HELYMEGHATÁROZÁS OPTIMALIZÁLÁSA NAPRENDSZERBELI ÉGITESTEN

KONZULENSEK
Dr. Bacsárdi László
Dr. Huszák Árpád

BUDAPEST, 2013

Összefoglaló

Napjainkban egy-egy naprendszerbeli égitest felszíni vizsgálatára költséges űreszközöket készítenek és küldenek. A jövőben azonban elképzelhető, hogy néhány nagyon drága eszköz helyett nagyszámú olcsóbb mérőeszközt juttatnak el egy-egy bolygó vagy kisbolygó felszínére, amelyek autonóm módon mozogva végeznek méréseket és juttatják el az eredményeiket egy földi irányító központba. Jelen dolgozatban a vizsgálataim egyik tárgya, hogy egy szenzorokból álló hálózat mozgásának irányítása hogyan oldható meg anélkül, hogy megszakadna a kapcsolat az egyes szenzorokkal, illetve különböző felszíni körülmények (például egy porvihar) milyen hatást gyakorolnak egy ilyen hálózatra.

A szenzorhálózat kialakításánál figyelembe vettem a hálózat elemeinek teljesítményét is. Ennek eredményeképpen a hálózatban kétféle szenzor különböztethető meg. Egyik, a nagy számban alkalmazott szenzor típus, amelyik a hálózaton belüli kommunikációval és az adatgyűjtéssel foglalkozik, másik pedig a műholdas kapcsolattartásért felelős. Az előbbi alacsonyabb jelszinten sugároz, utóbbi pedig magasabban, de mindkettőt befolyásolja az égitest felszínére jellemző környezet.

Az adatok gyűjtésénél fontos szempont, hogy melyik adatot hol mértük, így a szenzorok pozícionálása az egyik központi téma a dolgozatomban. A helymeghatározásra a háromszögeléses módszert alkalmaztam, miszerint három ismert helyzetű szenzor képes meghatározni egy negyedik szenzor helyzetét. Kidolgoztam a többlépcsős helymeghatározási módszert, mely alkalmas a lefedett terület jelentős kiterjesztésére, azonban a fellépő mérési hibák torlódnak, így a helyzetbecslés egyre nagyobb eltérést mutathat a valós értékektől. Ez a jelenség nagyobb mértékű adatvesztéssel járhat, így a dolgozatom másik fő célja a többlépcsős helymeghatározás hatékonyságának növelése.

C++ nyelven készítettem egy szimulációs programot, ami a szenzorhálózatot és annak kommunikációs, illetve mobilitási működését modellezi. Az égitest felszínén történő helymeghatározás a vett rádiós jelszintek alapján történik, ezért a domborzat hatásait figyelembe vevő Deygout jelterjedési modellt alkalmaztam. A többlépcsős pozícionálásra különböző optimalizáló algoritmusokat valósítottam meg a programban, melyek más és más irányból adnak becslést a szenzorok helyzetére. A valós és a becsült (számolt) értékek elemzésével olyan eredményeket kaptam, amelyek részletes képet adnak az algoritmusok hatékonyságáról.

Abstract

Nowadays, expensive space devices are developed and sent for exploration of the surface of a Solar System body. However, instead of some very expensive equipments, lot of cheap devices could be sent to the surface of a planet or planetoid. These sensors perform different measurements and send the results to central equipment on an autonomous way. In this work, my aim was to examine how to follow and control the movement of the sensor network without losing the connection with any sensor. Moreover, I analyzed the effect of variant aboveground events (e.g., dust storm) on this network.

By the developing of sensor network, I took the performance of the network elements into account. Two types of sensors are perceptible in the network. The first type is deployed in large number and it is responsible for collecting and forwarding environmental data, while the other type of sensors is able to communicate with satellites. The previous transmits on lower sign level, the latter transmits higher, but the environment of the planet affects

The environment of data measurement is a significant issue, so the positioning of sensors is a central subject in my work. I used triangulation for the position estimations, where three sensors with known position can be used to calculate the position of a fourth sensor. I proposed a multistep-positioning scheme that can extend the covered area where the sensor positions can be determined. The measurement errors accumulate by multistep-positioning, so the estimated coordinates show bigger difference from the valid value. This effect can cause considerable data loss, this is why the improvement of multistep-positioning was the other main subject of my work.

I have implemented a simulator program in C++ language, which models the sensor network movements, communication and positioning. The position estimation is based on received signal strength values of the radio communication that is influenced by terrain obstuction therefore in the simulations the Deygout propagation model was used. Several optimization algorithms were developed and used in the simulation software. With the analysis of valid and the estimated (calculated) coordinate values, I could evaluate the efficiency of my algorithms.

Tartalomjegyzék

1. Bevezetés ... 1
2. Űrkutatás, szenzorhálózatok és pozícionálás ... 3
   2.1 Távoli égitestek vizsgálata ... 3
   2.2 Szenzorok, szenzorhálózatok ... 4
   2.3 Jelterjedés ... 6
   2.4 Pozícionálás ... 8
3. A javasolt szenzorhálózat modellje ... 9
   3.1 Szenzorhálózat felépítése ... 9
      3.1.1 Egy bolygón mérhető adatok ... 9
      3.1.2 Műholdak igénybevétele és azok mozgása ... 9
      3.1.3 Szenzorok mozgása ... 9
      3.1.4 Nagyobb teljesítményű szenzorok a hálózatban ... 10
      3.1.5 Szuperszenzorok mozogása ... 10
      3.1.6 Szenzorok közötti kommunikáció (Single- és multi-hop hálózat) ... 10
   3.2 Többugrásos pozícionálás és háromszögelési módszer ... 10
      3.2.1 Matematikai számítások a háromszögelésre ... 11
   3.3 Pozícióbecslési algoritmusok ... 13
      3.3.1 Optimalizálatlan alap algoritmus ... 13
      3.3.2 Számítási hop számot figyelembe vevő algoritmus ... 14
      3.3.3 Átlagolást alkalmazó algoritmus ... 15
   3.4 Térképek alkalmazása ... 15
   3.5 Jelterjedési modellek kezelése ... 17
   3.6 Szimulátor működése ... 17
      3.6.1 Paraméterek kezelése ... 17
      3.6.2 Szenzorok mozgásának szimulálása ... 19
      3.6.3 Porvihar szimulálása ... 20
      3.6.4 Útvonal meghatározás ... 21
      3.6.5 Függőleges elmozdulás ... 22
4. Eredmények ... 23
   4.1 Hibatávolságok vizsgálata a módszerek függvényében ... 23

4.2 A hibatávolság változása a hopszámok függvényében 24
4.3 Hibatávolság változása a szenzorok számának függvényében 24
4.4 Hibatávolság változása a szenzorok hatótávolságának függvényében 25
4.5 A leszakadó szenzorok száma az algoritmusok függvényében 26
4.6 Leszakadó szenzorok száma a porvihar idejének függvényében 28
4.7 Szenzorok százalékos száma a biztonsági sávon kívül és belül a különböző algoritmusokra 29
5. Összefoglalás 31
Ábrajegyzék 32
Irodalomjegyzék 33

1. Bevezetés

A szenzorok hatékonyságát eddig már sokféle témakörben vizsgálták és bizonyították. Ezekkel az apró méretű érzékelőkkel képesek vagyunk parkolókban megállapítani az üres helyek számát, földtani vizsgálatokhoz adatokat gyűjteni, vagy akár biztonsági rendszereket fejleszteni. Jelen dolgozatomban egy a megszokottól eltérő területen vizsgáltam egy általam feltételezett szenzorhálózat működését.

A napjainkban használt űreszközök igen költségesek és bonyolult működtetésűek, így fennáll a lehetőség, hogy ezen eszközök mellett egyszerűbb és olcsóbb módszereket vetnek majd be naprendszerünk megismerésére. A NASA egyik nemrég megjelent cikkében szó volt arról, hogy a közeljövőben lehetséges lehet olyan módszer alkalmazása, miszerint egy naprendszeren kívüli kisbolygót robbantással közelebb hoznak és így elérhető közelségben lesz ahhoz, hogy megismerjük [1]. Ezen tervek megnyitják a szenzorhálózatok alkalmazhatósági területét az űrkutatásban. A dolgozatomban egy naprendszerbeli égitest felszínén mozgó szenzorhálózat alkalmazását és annak hatékonyságát vizsgálom, ami sok újszerű, érdekes problémát vet fel.

A szenzorhálózat kialakításánál fontos szempont volt a szenzorok teljesítőképessége és mobilitása. Ezek alapján a hálózatomban két féle szenzor szerepel, egy nagyobb teljesítményű szenzor, amelyet a földi irányítás közvetlenül irányít, és egy egyszerűbb szenzor, amely méréseket végez. Utóbbinak kisebb ugyan a teljesítőképessége, viszont nem olyan drága, mint az előbbi. Így ha a szenzorhálózatban megfelelő arányban alkalmazzuk a két szenzort, akkor egy olcsó, de hatékony hálózatot kapunk. A szenzorhálózat mobilitását tekintve egy mozgó hálózat, a szenzorok a felszínen gurulva igyekszenek eljutni egy célpontig. Ahhoz, hogy a szenzorok ne szóródjanak szét, hanem egy bolyban maradjanak egy biztonsági sávhatárt vezettem be, amit ha elérnek, akkor irányt változtatnak vissza a boly fele.

Egy naprendszerbeli égitest (pl. Mars) felszínén mozgó szenzorhálózat többféle, számunkra érdekes adatot gyűjthet össze. Mint pl. hőmérséklet, talajminta, légköri jellemző. Azonban ahhoz, hogy ezek az adatok el is jussanak a Földre és feldolgozásra kerülhessenek, biztosítani kell a szenzorhálózat megfelelő működését a földitől eltérő környezetben. Ebbe beletartozik az is, hogy a szenzorok közötti kommunikáció megfelelő legyen, illetve, egyik fő pontja a dolgozatomnak, a szenzorok helymeghatározási képessége is. Ahhoz, hogy hiteles képet kapjunk az égitestről, fontos tudnunk a mérések helyét, így az adatokat pozíciókhoz tudjuk kötni.

A pozícionálás megvalósításához háromszögeléses algoritmust alkalmaztam. A jelterjedést befolyásoló terepviszonyok miatt kialakul egy mérési hiba. Ez a hiba először még minimális, azonban többlépcsős helymeghatározásnál már akkora méretű is lehet, hogy a kommunikációt akadályozza, és ez adatvesztést eredményezhet. Ennek a hibának a visszaszorítása végett kidolgoztam két változatát az algoritmusnak, és ezeket a változatokat vizsgáltam a továbbiakban különböző szempontokból.

A vizsgálatokhoz C++ nyelven írtam egy szimulációs programot, amiben a fent említett szenzorhálózatot és annak viselkedését modelleztem. A mérési eredményeket grafikonokon ábrázoltam és ezek alapján részletesebb képet kaptam hálózat paramétereinek összefüggéseiről.

Ezek alapján a második fejezetben kitérek az irodalomban előforduló modellekre, amelyek alapjául szolgáltak a munkámnak. A harmadik fejezetben részletesen bemutatom a szenzorhálózatom működését mind az elméletben, mind a szimulációs programomban. Majd az eredményeimet ismertetem és a különböző algoritmusváltozatokat hasonlítom össze a negyedik fejezetben, végezetül pedig összefoglalom a dolgozatom főbb pontjait.

2. Űrkutatás, szenzorhálózatok és pozícionálás

2.1 Távoli égitestek vizsgálata

Ahogy már a bevezetőben is említettem, az űrkutatásban használt eszközök napjainkban igen bonyolult működésűek és költségesek. Az űreszközök fejlődése azzal az eredménnyel járt együtt, hogy egyre többet tudtunk meg a Naprendszerünkről és távolabbi bolygókról, kisbolygókról is. A technikai tudás előrehaladtával egyre távolabbi égitestekről alkothattunk egyre részletgazdagabb képet.

Kezdetben még csak távcsövek álltak rendelkezésre a vizsgálatokhoz, amik csak vázlatrajzokat és becsléseket eredményeztek, ma már műholdak és űrszondák segítségével pontos méréseket végezhetünk, és így hitelesebb adatokat nyerhetünk az égitestekről. Az alábbi 1. ábra remekül szemlélteti, hogy napjainkban mennyire kidolgozott képünk van a felfedezésekről.

1. ábra
A Mars felszíne egy 2012. szeptember 28.-án készült felvételen Forrás: www.nasa.gov

Számos magyar vonatkozású űreszköz is ismert. Az űrkorszak kezdete óta eltelt évtizedekben hozzájárultunk az űrkutatás tudományához, akár műholdátviteli kísérletekről, akár fedélzeti eszközökről, akár olyan sugárzásmérő készülékről is van szó, mint a Pille. Napjainkban, hazánkban egyre nagyobb teret kap az űreszközök fejlesztése is. Erre nagyszerű példa a Budapesti Műszaki és Gazdaságtudományi Egyetemen készült Masat-1 műhold is.

A bolygók felszínének feltérképezésében nagy segítséget nyújthatnak a szenzorok, szenzorhálózatok. A szenzorhálózatok irodalma meglehetősen ismert, a modern űrkutatás területén alkalmazható szenzorhálózatok azonban nagyban segíthetik a felszíni megfigyeléseket. A távoli égitest felszínén alkalmazható szenzorhálózatok esetében a

költséghatékonyság mellett nagy előny a széleskörű felhasználhatóság. Apróbb változtatásokkal az alap hálózat többféle mérési funkciókat is elláthat. Ezért szeretném mélyebben bemutatni ezt a témakört a következő fejezetekben.

2.2 Szenzorok, szenzorhálózatok

A szenzorok olyan eszközök, amelyek különböző fizikai jelenségek megfigyelésére képesek (pl. hőmérséklet, fény, páratartalom) és a mért adatokat képesek továbbítani egy speciális eszköz, a nyelő felé. Ezáltal az égitest felszínén végbement mérési eredményeket a nyelő össze tudja gyűjteni és képes lehet továbbítani a Föld felé.

A szenzorhálózatok nagy darabszámú egyszerű szenzor egységből álló, autonóm működésre képes elosztott számítógépes hálózatok. Az érzékelő egységekből kifolyólag megfigyelő, adatgyűjtő funkciót valósítanak meg, esetenként adatfeldolgozó és analizáló feladatokra is képesek. A költséghatékonyságát növeli az alkalmazásának, hogy a szenzor elemei alacsony árban vannak, ugyanakkor gondot okozhat az erőforrás utánpótlása, a lemerült elemek cseréje. Pl. egy bolygó felszínére elhelyezett szenzorhálózat esetében a szenzorok nem rendelkezhetnek nagy napelemmel illetve akkumulátorral/generátorral. Ezért fontos a lehető legalacsonyabb energiafelhasználásra való törekvés. A hatékonyságot növelheti, hogy ha csak kevés szenzor képes a nagy energiatartalékok tárolására, tipikusan ezek a nyelő szenzorok, amelyek képesek kell lenniük a kommunikációra nem csak a szenzorokkal, hanem a Föld felé is műholdak segítségével. Mivel a szenzorhálózaton belül vezeték nélküli hálózati összeköttetés van jelen a szenzorok között, így könnyű azok mozgatása. Ez két szempontból hasznos [2]. Egyrészt, hogy a kezdetben le nem fedett területekre is kerüljenek szenzorok, illetve hogy a szenzorok adott területen belüli egyenletes eloszlását könnyebb legyen fenntartani. A szenzorhálózat megfelelő összetételével egy hatékony alkalmazás jöhet létre a felszíni vizsgálatokhoz.

Ezért először egy rövid áttekintést adok a különbözőféle szenzorokról és mobilitási stratégiájukról, majd a következő fejezetben térek ki a saját hálózatom részletezésére.

A szenzoroknak három főbb fajtáját különböztethetjük meg [3]:

- eseményvezérelt szenzor,
- lekérdezés vezérelt szenzor,
- idővezérelt szenzor.

Az eseményvezérelt szenzor esetében egy adott esemény a katalizátora az adatküldésnek. A lekérdezés vezérelt szenzorokból álló hálózatnál a nyelő intéz egy kérést a szenzorok felé, és ekkor kezdenek adatot küldeni. Az idővezérelt szenzorok esetében pedig egy meghatározott időközönként történik meg az adatküldés.

Magának a szenzorhálózatoknak két nagyobb csoportja van, angol szakszóval a single-hop, illetve a multi-hop. A single-hop esetében a szenzorok csak egy lépcsőben tudnak kommunikálni, azaz közvetlenül a nyelővel tudnak kapcsolatot teremteni. A multi-hop

hálózatokban azonban a szenzorok egymás között is tudnak adatot átküldeni, így az adat több lépésben jut el a nyelőhöz.

Napjaink szenzorjainak nagy előnye, hogy nem csak rögzített vagy vezetékes változatuk létezik, hanem a vezeték nélküli összeköttetés révén mozgásra is képesek. Ez két szempontból hasznos: A kezdetben le nem fedett területekre is kerüljenek szenzorok, illetve a szenzorok adott területen belüli egyenletes eloszlását könnyebb legyen fenntartani. Azon pontok halmazát, amelyek az adott szenzorhoz vannak a legközelebb Voronoi tartománynak nevezzük [4]. Ennek a tartománynak a kiszámítása után megállapítható, hogy ha a szenzor érzékelési sugara kisebb, mint a Voronoi tartomány bármely pontjától való távolság, akkor van olyan térrész, amely nincs lefedve. Ez a megfigyelés a mozgatás végrehajtásában játszik fontos szerepet. Az alábbi mobilitási stratégiákat tartom fontosnak megemlíteni [3].

Első a *VEC (VECtor-based algorithm)*. Ez esetben a célunk az egyenletes eloszlás kivitelezése. Ahol sűrűbben helyezkednek el a szenzorok, onnan átmozgatunk annyi szenzort egy kevésbé lefedett területre, hogy az eloszlás minél inkább kiegyenlített legyen.

Második mobilitási stratégia a *VOR (VORonoi-based algorithm).* Itt a szenzorok a Voronoi tartomány segítségével meghatározott „lyukak”, nem lefedett területek felé mozognak.

Harmadik stratégia a *Minimax*, ami nagyban hasonlít a VOR-hoz**.** Itt azonban azt is figyelembe kell venni a mozgatás során, hogy ne hagyjanak a szenzorok maguk után lyukakat. Azaz az elmozdulás ne eredményezzen lefedetlen területet.

Az előző három stratégiánál hatásosabb azonban az, ha egy meghatározott útvonal mentél mozognak a szenzorok. Ebben az esetben a szenzor nem csak az aktuális helyzetét ismeri, hanem a következő elérendő pontot is. Erre egyik lehetőség az, ha a saját aktuális helyzetén kívül ki tudja számolni a következő elérendő pont helyzetét is. Másik lehetőség, ha a mostani helyzetét és az eddig megtett utat ismeri, így könnyen meg tudja becsülni a következő lépését.

A hálózat hatékonyságán nagymértékben tud javítani, ha a nyelők is mozognak. Ekkor a szenzorok egy lépésben könnyebben el tudják juttatni a mért adatokat a nyelőnek és így kevesebb energiát kell felhasználniuk. Ekkor a nyelő mozgatása igényel valamivel több energia felhasználást. A megfelelő nyelő/szenzor darabszámok meghatározásával energiaspórolást eredményezhetünk. Felvetődik a kérdés, hogy a nyelőt milyen stratégia szerint érdemes mozgatni. Erre három módszer az elterjedtebb. Első a véletlenszerű mozgatás. Ilyenkor a nyelő nem előre meghatározott módon, véletlenszerűen mozog. Második a jósolható mozgatás. Ebben az esetben a nyelő egy előre kijelölt pályán halad végig. Így a szenzorok mozgásához igazodva tud haladni és a szenzorok adatátküldése hatékonyabb lehet. Harmadik módszer a vezérelt mozgatás. Ez hasonlít az előző módszerhez. A különbség annyi, hogy itt a hálózat aktuális állapotát is figyelembe kell venni az útvonal meghatározásakor. Tehát a nyelő útvonala dinamikusan változhat.

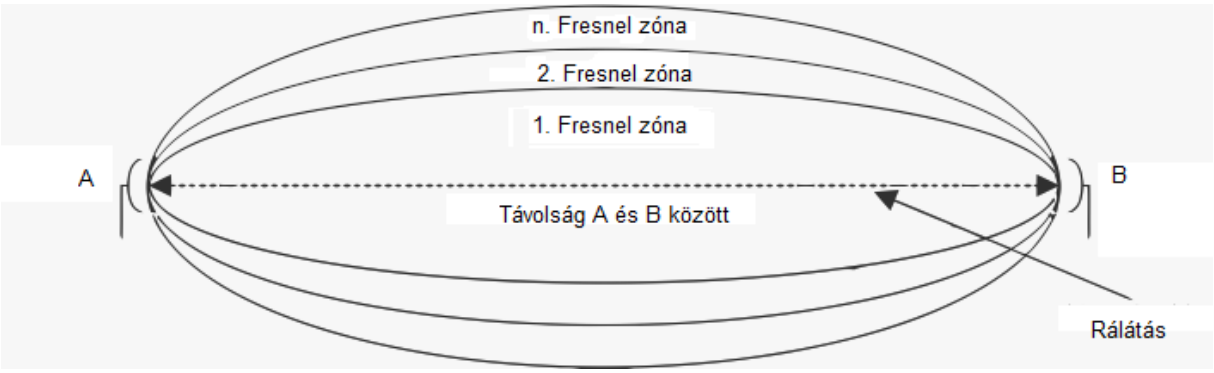
Fontos a szenzorok hierarchiába történő meghatározása. Ez azt jelenti, hogy adott szerepeket osztunk a szenzoroknak több szempont alapján. Figyelembe kell venni például, hogy az átküldendő adatoknak mennyire kell frissnek lenniük. Ez az adat nyelőhöz való kerülésének

idejét köti meg. Ha a mért adat naprakészen kell, hogy a nyelőhöz kerüljön, akkor lehet érdemesebb a szenzoroknak egymás között is kommunikálniuk, így az adat előbb odaérhet a nyelőhöz, mintha ki kellene várni azt az időt, amíg a nyelő az adott szenzor közelébe ér. Ha viszont a mért adat naprakészsége nem fontos, akkor az energiatakarékosság szempontjából hatékonyabb lehet, ha a szenzorok csak a nyelővel tudnak kommunikálni. Sok szempont alapján meghatározható egy olyan hierarchia, ami leírja, hogy az egyes szenzorok mely másik szenzorokkal tudnak kommunikálni.

2.3 Jelterjedés

A szenzorok közötti kommunikációjában nagy szerepet játszik a jel terjedése, csillapodása. Mivel az égitestek felszíne tipikusan nem sík terep, hanem kráterekkel, dűnékkel, sziklás emelkedőkkel tarkított, ezért az átküldött jeleket más és más vevőteljesítményekkel tudják érzékelni. A jelterjedés alkalmazására a Deygout módszert [5] és a szabadtéri jelterjedési modellt vettem alapul [6][7].

A Deygout módszerben a vizsgálni kívánt terepmetszet egy intervallumot határoz meg. Az átvitel szakaszcsillapítását nagymértében befolyásolja ha a Fresnel-zóna takarásba kerül [8]. Kiemelt szerepe van az első Fresnel zónának, mert az energia körülbelül 90%-a ezen a nyomvonalon halad keresztül. A Fresnel-zóna számítása az alábbi 2. ábrán és 1. képleten látható:



2. ábra
Fresnel zóna [Forrás: http://www.zytrax.com/tech/wireless/fresnel.htm]

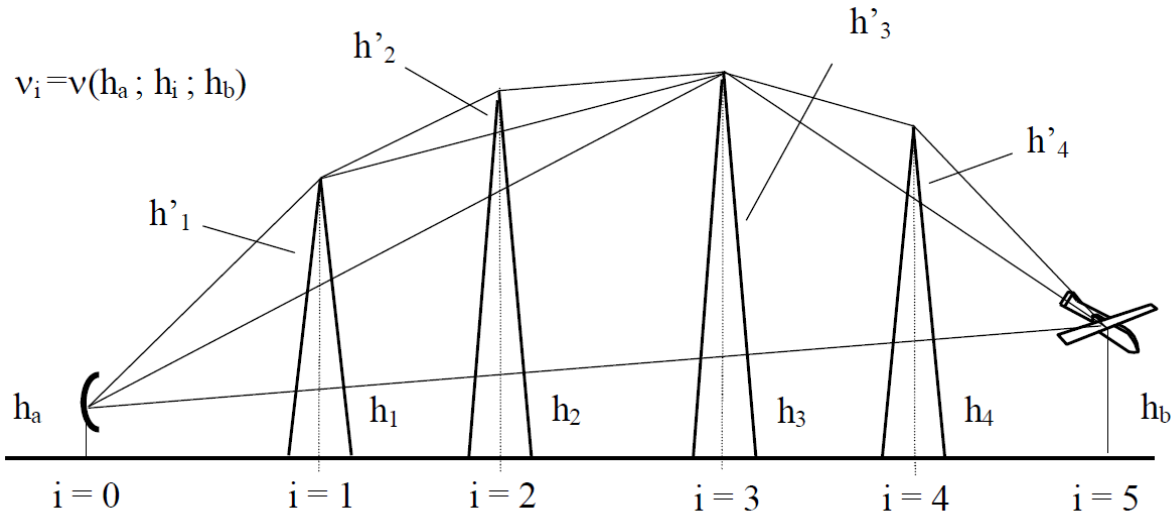
$$r = 17.3 * \sqrt{\frac{d*(D-d)}{f*D}} \quad , \qquad (1)$$

ahol

r = Fresnel zóna sugara, méterben
f = frequencia GHz-ben
d = vizsgált távolság kilométerben
D = teljes távolság kilométerben.

Ezek alapján tehát esetünkben az I. Fresnel zóna gyújtópontjai az intervallum határain lesznek. Ezek után az intervallum összes csúcsát figyelembe véve meg kell határozni a maximális fadinget okozó csúcs helyét, azaz ahol a legnagyobb a benyúlás mértéke. A maximális fadinget okozó késélnél az intervallum két részre válik szét. Majd rekurzívan

ezekre a részintervallumokra is meghatározzuk a hozzá tartozó maximális fading helyét, ami szintén tovább osztja az intervallumot. Ez a rekurzív eljárás addig folytatódik, amíg a részintervallumokban késél található. Az így kiszámított maximális fadingek összege adja ki az eredő diffrakciós fadinget. Ez az eljárás a 3. ábrán és a további képleteken kísérhető figyelemmel.



3.ábra
Deygout modell

$$v = h * \sqrt{\frac{2}{\lambda}} * (\frac{1}{d1} + \frac{1}{d2}) \qquad (2)$$

$$v = h * \sqrt{\frac{2}{\lambda}} * (\frac{1}{d1} + \frac{1}{d2}) \qquad (2)$$

$$L(v0)[dB] = 20 * \lg(1) \qquad \text{ha } v0 \in (-\infty; -0{,}8) \qquad (3)$$

$$L(v0)[dB] = 20 * \lg(0{,}5 - 0{,}62 * v0) \qquad \text{ha } v0 \in [-0{,}8; 0) \qquad (4)$$

$$L(v0)[dB] = 20 * \lg(0{,}5 * 10^{-0{,}95 * v0}) \qquad \text{ha } v0 \in [0; 1) \qquad (5)$$

$$L(v0)[dB] = 20 * \lg\left(0{,}4 - \sqrt{0{,}1184 - (0{,}38 - 0{,}1 * v0)^2}\right) \qquad \text{ha } v0 \in [1; 2{,}4) \qquad (6)$$

$$L(v0)[dB] = 20 * \lg\left(\frac{0{,}225}{v0}\right) \quad \text{ha } v0 \in [2{,}4; \infty) \qquad (7)$$

A szabadtéri jelterjedési modell alapja, hogy nincs fading jelenség, azaz a jeleknek nem kell áthatolni semmilyen akadályon. Ebben az esetben a vett jel erősségét az alábbi képlet írja le.

$$Pr = Pt + Gt + Gr + \frac{\lambda}{4 * \pi * d}^2, \qquad (8)$$

ahol

Pr: vett jel teljesítménye (W)

Pt: adóteljesítménye (W)

Gt: adóantenna nyeresége (viszonyszám)

Gr: adóantenna nyeresége (viszonyszám)

λ: hullámhossz (m)

d: távolság (m)

Az egyenletből kivehető, hogy a vételi teljesítmény egyenes arányban áll az antennanyereségekkel, illetve fordított arányban van a távolság négyzetével.

2.4 Pozícionálás

A helymeghatározás több fontos célt is szolgálhat. Fontos lehet számunka a saját helyzetünk, akár a körülöttünk lévő tárgyak, emberek helyzetének meghatározása. De akár a navigációnak vagy a tudományos és a diagnosztikai képalkotásnak is fontos elme a pozícionálás. Szenzorok segítségével olyan intelligens ágenseket is építhetünk, amelyek emberi beavatkozás nélkül is működhetnek. Ez sokszor nagyban megkönnyítheti a mi feladatainkat is.

A helymeghatározásnak kültéri és beltéri alkalmazása is egyaránt lehetséges. Kültéri alkalmazás szempontjából fontosak a GNSS rendszerek (Global Navigation Satellite System, globális navigációs műholdrendszer), ilyen például az amerikai GPS (Global Positioning System, Globális Helymeghatározó Rendszer), az orosz GLONASS, a kínai Beidou vagy az európai Galileo rendszer. Sokan használják a GPS-t akár autóvezetésnél, akár túrázás során, hiszen a bolygó egész területén méteres pontosság érhető el polgári célokra. Egy távoli égitest felszínén azonban nem tudunk ilyen jellegű rendszert használni, amíg nem építettük ki a megfelelő műholdas rendszert, ezért más módszerekkel kell a helymeghatározást megoldanunk. Beltéri alkalmazás esetében a rádiós alapú RFID (Radio Frequency IDentification, rádiójeles azonosítás) technológia megfelelő szoftveres támogatással alkalmassá válhat helymeghatározásra. Így létrehozhatók olyan intelligens háztartási gépek, amelyek önállóan képesek elvégezni a feladatukat. A pozícionálás használta nem szűkíthető kizárólag a tájékozódásra. Helyzet érzékeny szolgáltatások és információk hozzáféréséhez is elengedhetetlen, valamint biztonsági vagy megfigyelőrendszerek használta is nagyban épít rá.

Ahhoz, hogy meg tudjuk mondani például azt, hogy egy kisbolygón talált vízjég pontosan hol helyezkedik el, vagy azt, hogy pontosan hol találtuk a nekünk fontos talajmintát, ahhoz ismernünk kell a szenzorok mérési pozícióit. Mindez mutatja, hogy a helymeghatározás igen nagy jelentőséggel bír a témám vizsgálatánál.

Mivel GPS-jellegű rendszer nem áll rendelkezésre egy távoli égitest felszínén, ezért a pozíció meghatározáshoz a háromszögelés módszerén alapuló többlépcsős helymeghatározási módszert dolgoztam ki. A pontok egymáshoz viszonyított helyzete alapján való helymeghatározásra az egyik legpontosabb módszer [9]. A háromszögelés valójában nem áll másból, mint háromszögek oldalainak és szögeinek meghatározásából. Ezért bármilyen sok pont esetén is, ha csak a pontok nem esnek egy egyenesbe, akkor meghatározhatóak a nekünk kellő háromszögek. Így ha a háromszögek alkotóelemeit ismerjük, akkor a pontok egymáshoz viszonyított helyzetét is ismerjük. A háromszögeléses módszert bővebben a következő fejezetben ismertetem.

3. A javasolt szenzorhálózat modellje

3.1 Szenzorhálózat felépítése

A választott témám egy elég tágan értelmezhető témakör, ezért nagyon sok kérdés vetődik fel a mérési környezettel kapcsolatban. Összegyűjtöttem a legfontosabbnak talált kérdéseket és a lehetséges válaszok közül kiválasztottam azokat (az előző fejezetben bemutatott csoportosításokat is figyelembe véve), amelyek az általam vizsgált hálózatra, környezetre jellemzőek.

3.1.1 Egy bolygón mérhető adatok

Fontos figyelembe venni, hogy olyan adatot is mérhetünk, amelynek nem feltétlenül kell naprakésznek lennie. Például ilyen a talált víz, jég pozíciója, vagy a talajminta, amelyek egy bizonyos idő eltelte után is érvényesek maradnak. Ezzel szemben olyan adatra is szükségünk lehet, mint például az aktuális hőmérséklet, ahol fontos, hogy azt az adatot mikor mérték és időben fel tudják dolgozni. Ezen kívül meghatározó tulajdonság lehet, hogy mikor érdemes mérni az adott adatot. Periodikusan mérhető adat lehet a hőmérséklet, ahol elég csak meghatározott időközönként méréseket végrehajtani. Esemény hatására bekövetkező mérést légköri jelenségek mérésénél érdemes választani. A vizsgált hálózatom esetében olyan adatok mérését választottam, ahol nem kell naprakésznek lennie az adatoknak, a nélkül is érvényesek maradnak, illetve a méréseket periodikusan hajtják végre a szenzorok.

3.1.2 Műholdak igénybevétele és azok mozgása

Ez a kérdés leginkább a költséghatékonyság szempontjából fontos. Ha több műholdat használunk a hálózathoz, akkor ugyan könnyebben fenntartható a műhold-szenzor kapcsolat, de ez jóval drágább megoldás. Ha kevesebb műholdat használunk, az jóval olcsóbb, ellenben nem tartható fent egy állandó kapcsolat a szenzorokkal. A Földre csak meghatározott időközönként lehet eljuttatni az adatokat, mert a nyelők csak bizonyos időszakokban képesek a műholdakat fenntartani a kapcsolatot. Mivel az előző kérdésnél meghatároztam, hogy az adatok érvényessége sokáig fenn áll, így a vizsgált hálózatomnál kevés műhold használatát feltételezem.

3.1.3 Szenzorok mozgása

Két, az előző fejezetben már említett fő csoportra osztottam a mozgás fajtáit. Egyik a véletlenszerű mozgás, ahol előre nem meghatározott pályán mozognak a szuperszenzorok, másik az adott pályán való mozgás. A két mozgásfajtának egy kombinációját vettem. Eszerint a szenzorok kezdetben egy véletlenszerű irányba kezdenek el mozogni és ezen a vonalon haladnak tovább egy előre meghatározott pont síkja felé. Egy általam biztonsági sávnak nevezett sávot is megadtam (a szenzorok y koordinátájára egy felső és egy alsó korlát), amelyen belül a szenzorok mozoghatnak. A biztonsági sáv alkalmazásával nem lesz olyan szenzor, amelyik a sávtól eltávolodna, és ezért kiesne a szenzorbolyból. Ha a szenzor elérte

ennek a sávnak a határát, akkor irányt vált, az eddigi *y* koordinátája a mínusz egyszeresére módosul, így érve el, hogy a szenzorok a biztonsági sávon belül mozogjanak és ne távolodhassanak el túlságosan egymástól.

3.1.4 Nagyobb teljesítményű szenzorok a hálózatban

A nagyobb teljesítményű szenzorokra innentől kezdve szuperszenzorokként fogok hivatkozni. Ha nincsenek szuperszenzorok a hálózatban, akkor a szenzorok energiaellátás szempontjából olcsóbbak, ugyanakkor mindegyik szenzornak képesnek kell lennie kommunikálni a földi irányítással, hogy a mérési adatok eljuthassanak a Földre. Ha vannak a hálózatban szuperszenzorok, akkor ugyan ezek energiaellátása drágább, de csak a szuperszenzornak kell képesnek lennie a nagy teljesítményű kommunikációra, ami jelentős költségcsökkenést eredményez. Ezért úgy döntöttem, hogy a szuperszenzorok lesznek a szenzorhálózat nyelői. Dolgozatomban azzal a feltételezéssel éltem, hogy a szuperszenzorok pontosan ismerik a mindenkori pozíciójukat, így a többlépcsős helymeghatározás első lépésében referenciaként használhatók.

3.1.5 Szuperszenzorok mozogása

Ha a szuperszenzorok képesek a mozgásra, akkor ugyan plusz energiát igényelnek, de könnyebben összegyűjtik a mérési adatokat a velük együtt mozgó szenzoroktól, így kevésbé léphet fel adatvesztés. Ha nem tudnak mozogni, akkor kevesebb energiaellátás szükséges nekik, de sok olyan szenzor lehet, akik nem tudják eljuttatni hozzájuk az adatokat. Ebből kifolyólag úgy döntöttem, hogy a vizsgált hálózatomban a szuperszenzorok mozoghatnak. A mozgás mikéntjére ugyanazokat a lehetőségeket vettem figyelembe, mint a szimpla szenzorok esetében és ugyanarra a megoldásra is jutottam.

3.1.6 Szenzorok közötti kommunikáció (Single- és multi-hop hálózat)

Ha a szenzorok egymással is képesek kommunikálni (multi-hop hálózat), az jóval több energiafelhasználást igényel, mintha csak a szuperszenzorokkal tudnának kapcsolatba lépni (single-hop). Ellenben ha a hálózat multi-hop, akkor az adatok gyorsabban elérnek a nyelőhöz és előbb jutnak továbbításra. Mivel előzőleg meghatároztam, hogy az adatok érvényessége sokáig kitart, ezért az energiatakarékosság szempontjából a hálózatom single-hop lesz.

3.2 Többugrásos pozícionálás és háromszögelési módszer

A pozícionáláshoz a háromszögelés módszerét alkalmaztam, mely a javasolt többugrásos helymeghatározás alapjául is szolgál. Az előző fejezetben leírtak szerint egy szenzor helyzetének meghatározásához szükség van másik három szenzor helyzetének ismeretére. Ez alapján kiválasztottam három szuperszenzort, amelyek helyzete kezdettől fogva ismert a földi irányítás miatt, kezdettől fogva referenciapontoknak számítanak. Ezekből megállapítható azon szenzoroknak a helyzete, amelyek beleesnek a szuperszenzorok hatótávolságaiba, azaz abba a távolságba, amelyen belül még látják egymást és képesek kommunikálni. A kiszámítás matematikai módja a következő alfejezetben írtak alapján történik.

A többugrásos helymeghatározás során a kiszámolt helyzetű szenzor is referenciaponttá válik és így az ő segítségével is lehet tovább számolni a többi szenzor koordinátáit. Ha a szenzor helymeghatározásában szerepet játszik egy bizonyos hibaérték, akkor a belőle tovább számított szenzorok helymeghatározásában is szerepelni fog ez a hiba valamilyen súlyozással, azaz a hiba torlódik.

A helymeghatározási folyamatot rekurzívan folytatva meghatározhatjuk az egész szenzorboly helyzetét. A szenzorboly azoknak a szenzoroknak az összessége, amelyek a szenzorhálózat részei. Ha azonban valamelyik szenzor olyan messze kerül a bolytól, hogy nincs három olyan másik referenciapont, amelyek hatósugaraiban benne lenne, akkor az ő helyzete már nem határozható meg, vagyis leszakad a bolytól. Később visszatérhet és újra kiszámolható lesz a helyzete, de ez a visszatérés nem garantált. Ennek a kiküszöbölésére vezettem a biztonsági sávhatárt, ami egy felső és alsó korlátot ad a szenzorok *y* koordinátájára. Ha ezt a sávhatárt átlépi a szenzor, akkor már nagy valószínűséggel nem határozható meg a helyzete, túlságosan távol kerül a bolytól. Ezért ha a szenzor *y* koordinátája eléri ezt a korlátot, akkor ellentettjére változik, ezáltal visszakerül a bolyba. Gond csak akkor léphet fel, ha a hibatorlódás miatt elcsúszik egymástól a számolt és a valós koordináta érték, így a szenzor nem ismeri fel, hogy túllépte a sávot.

3.2.1 Matematikai számítások a háromszögelésre

A háromszögeléses módszer alkalmazásához az alábbi számításokat használtam fel.

Legyen a *B1*, *B2* és *B3* három olyan szenzor, amelyek referenciapontként viselkednek, ezek helyzeti koordinátája rendre (*x1, y1*), (*x2, y2*), (*x3, y3*). Az A szenzor legyen a kiszámítandó helyzetű szenzor, az ő helyzetének koordinátája (*x, y*). Az AB1, AB2, AB3 távolságokat pedig *d1*, *d2*, *d3* jelöléssekkel nevezem el. A következő egyenletekből állítottam fel egy egyenletrendszert, melynek megoldásai a keresett A pont (*x,y*) koordinátái.

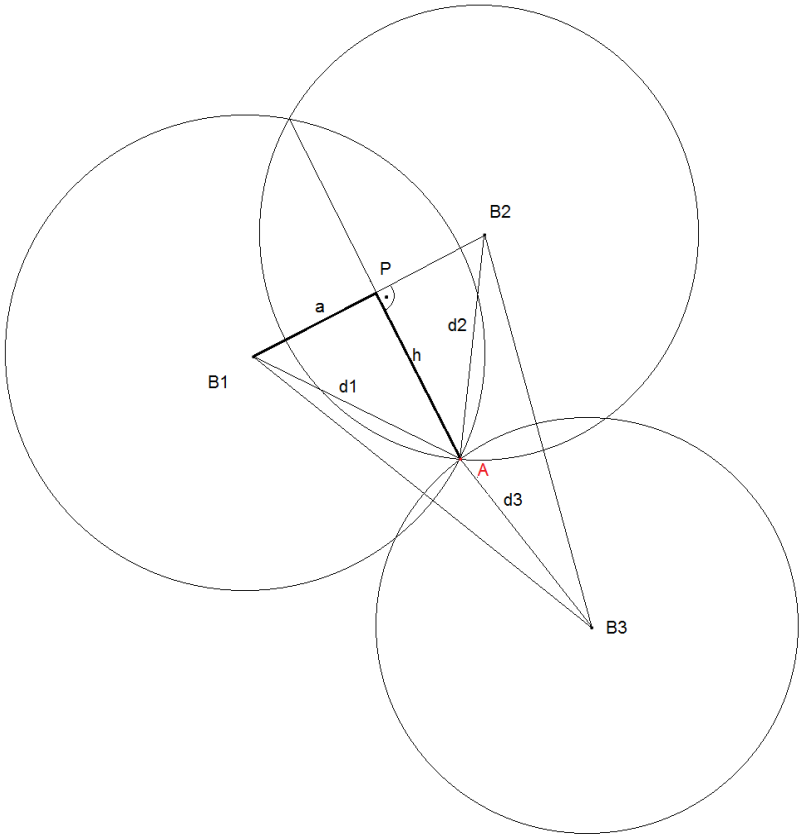
$$\mathrm{d1} = \sqrt{(\mathrm{x1} - \mathrm{x})^2 + (\mathrm{y1} - \mathrm{y})^2}, \qquad (9)$$

$$\mathrm{d2} = \sqrt{(\mathrm{x2} - \mathrm{x})^2 + (\mathrm{y2} - \mathrm{y})^2}, \qquad (10)$$

$$\mathrm{d3} = \sqrt{(\mathrm{x3} - \mathrm{x})^2 + (\mathrm{y3} - \mathrm{y})^2}, \qquad (11)$$

Geometriai ábrázolással és számításokkal sokkal látványosabb és programozhatósági szempontból egyszerűbb egyenleteket kaptam, ezért ezekkel vezettem le a megoldás menetét.

A 4. ábrán láthatóak a geometriai összefüggéseket megadó alakzatok.



4. ábra
A háromszögelés geometriája

A *B1* és a *B2* pont távolságát az alábbi módon tudjuk kiszámolni:

$$\text{tavB1B2} = \sqrt{(\text{x1} - \text{x2})^2 + (\text{y1} - \text{y2})^2}, \qquad (12)$$

Ezután a *B1PA* és a *B2PA* háromszögekben alkalmazott Pitagorasz-tétel segítségével kiszámolható az a oldal hossza:

$$\text{d1}^2 = \text{h}^2 + \text{a}^2, \qquad (13)$$

$$\text{d1}^2 = \text{h}^2 + (\text{tavB1B2} - \text{a})^2, \qquad (14)$$

A két egyenletet egyenlővé téve:

$$\text{a} = \frac{\text{d1}^2 - \text{d2}^2 + \text{tavB1B2}^2}{2 * \text{tavB1B2}}, \qquad (15)$$

Ebből következik a h oldal hossza az egyik előző Pitagorasz-tételből adódóan:

$$\text{h} = \sqrt{\text{d1}^2 - \text{a}^2}, \qquad (16)$$

Ezután már csak a *P* pont koordinátáit kell kiszámolni és onnan megkapható a keresett *A* pont koordinátája is. A *P* pont koordinátái *px* és *py*, de mivel várhatóan két értékpárt kapok, ezért *px1, py1* és *px2, py2*. Legyen b az alábbi kifejezés:

$$\mathrm{b} = 1 + \left(\frac{\mathrm{y2}-\mathrm{y1}}{\mathrm{x2}-\mathrm{x1}}\right)^2, \qquad (17)$$

Ezt behelyettesítve a másodfokú egyenlet megoldó képlet eredményeibe:

$$\mathrm{px1} = \frac{2*\mathrm{x1}+\sqrt{4*\mathrm{x1}^2-4*\mathrm{x1}^2+4*\mathrm{a/b}}}{2}, \qquad (18)$$

$$\mathrm{px2} = \frac{2*\mathrm{x1}-\sqrt{4*\mathrm{x1}^2-4*\mathrm{x1}^2+4*\mathrm{a/b}}}{2}, \qquad (19)$$

$$\mathrm{py1} = \frac{\mathrm{y2}-\mathrm{y1}}{\mathrm{x2}-\mathrm{x1}} * (\mathrm{px1}-\mathrm{x1}) + \mathrm{y1}, \qquad (20)$$

$$\mathrm{py2} = \frac{\mathrm{y2}-\mathrm{y1}}{\mathrm{x2}-\mathrm{x1}} * (\mathrm{px2}-\mathrm{x1}) + \mathrm{y1}, \qquad (21)$$

Ez a megoldás, amint látható két eredményt ad a *P* pontra, mert a két kör két pontban metszi egymást. E két pont közül azt választom ki azt, amelyik számomra megfelelőbb a keresett *A* pont kiszámításához. Mivel itt is valószínű, hogy több eredményt kapok, ezért innentől az *A* pont koordinátáit *Ax1, Ay1* és *Ax2, Ay2*-vel jelölöm.

$$\mathrm{Ax1} = \mathrm{px} + \frac{\mathrm{h}*(\mathrm{y2}-\mathrm{y1})}{\mathrm{tavB1B2}}, \qquad (22)$$

$$\mathrm{Ax2} = \mathrm{px} - \frac{\mathrm{h}*(\mathrm{y2}-\mathrm{y1})}{\mathrm{tavB1B2}}, \qquad (23)$$

$$\mathrm{Ay1} = \mathrm{py} + \frac{\mathrm{h}*(\mathrm{x2}-\mathrm{x1})}{\mathrm{tavB1B2}}, \qquad (24)$$

$$\mathrm{Ay2} = \mathrm{py} - \frac{\mathrm{h}*(\mathrm{x2}-\mathrm{x1})}{\mathrm{tavB1B2}}, \qquad (25)$$

Itt is két eredményt kapok, ezek közül pedig szintén a számomra megfelelőbbet kell kiválasztani. Ez esetben ez az a pont lesz, amelyik benne van a harmadik kör hatósugarában, azaz a *B3* és a *W1* vagy *W2* pont távolsága kisebb vagy egyenlő, mint *d3*.

Ezzel megkaptam a keresett *A* szenzor helyét.

3.3 Pozícióbecslési algoritmusok

3.3.1 Optimalizálatlan alap algoritmus

A kezdeti algoritmusom esetében még nincs optimalizáló eljárás. A kiszámítandó helyzetű szenzor egy egyszerű, bár kevésbé hatékony módszerrel választja ki a potenciális referencia szenzorok közül a három szülőjét. Az algoritmusban abból áll, hogy a bolyban szereplő szenzorokat elkezdi vizsgálni, és ha referencia pont, akkor beveszi a szülők közé. Az első

három találat után a vizsgálat megáll és adottak lesznek a használni kívánt referencia szenzorok. Tehát a bolyból az első három referencia szenzort választja ki.

A módszer azért nem hatékony megoldás, mert ha a lehetséges referencia szenzorok közül három olyat választ ki az algoritmus, akik már torlódott hibát hordoznak magukkal, akkor ő hibájukat az adott szenzorunk is hordozni fogja. Tehát a számítása valamilyen mértékben biztosan eltér majd a valós helyzetétől. Ezzel szemben, ha választhatná szülőknek a három alap szuperszenzort, akkor jóval kisebb hiba adódna a kiszámolt értékhez, mint az előbbi esetben, mert a szuperszenzorok pozíciója biztosra ismert.

Erre az algoritmusra a továbbiakban 1. algoritmusként fogok hivatkozni.

3.3.2 Számítási hop számot figyelembe vevő algoritmus

Ez az algoritmus leginkább abban különbözik az előzőtől, hogy minként választ magának egy szenzor szülőket. A szenzorok tárolnak egy olyan hop értéket, ami arra utal, hogy hány lépcsős pozícionálás eredménye a kiszámolt koordinátájuk. Minél későbbi több lépcsőben kerül sor egy szenzor pozíciójának becslésére, a hibatorlódás annál jelentősebb. A cél ezért az, hogy olyan szenzorokat jelöljünk ki referenciaként, melyeknek koordinátáit a lehető legkevesebb lépcsőben számíthatók ki. A szuperszenzorok hop száma alapból az egész vizsgálat során 0. A szülők hopszáma átlagolódik, és ehhez hozzáadódik egy egységnyi érték, ezzel jelezve, hogy plusz egy számítás történt.

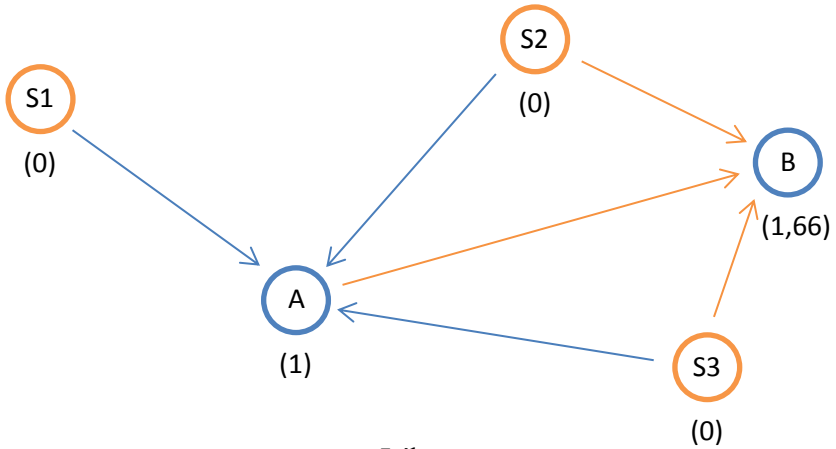
Példaként vegyünk egy *A* szenzort, aki nem referencia pont és a szülői legyenek a szuperszenzorok, *S1, S2, S3*. Ekkor *A* hopszáma az alábbi lesz:

$$(1+1+1)/3 = 1 \qquad (26)$$

Ha *B* szintén nem referencia szenzor szülői közé beletartozik ezután *A* szenzor és két szuperszenzor, akkor az ő hopszáma a következőképp alakul.

$$(1+1+2)/3 = 1{,}66 \qquad (27)$$

A bemutatott példa a következő 5. ábrán látható.



5 ábra
A hopszám számítása

Ebből kifolyólag a hopszám azt is mutatja, hogy mennyire pontos számítást tudunk végezni a pozícionálás során. Minél kisebb egy szenzor hopszáma, annál pontosabb számítási eredményt kapunk. Pont ezért egy szenzor az alapján választ magának szülőket, hogy minél kisebb legyen a hopszáma. Így mindig a lehető legpontosabb koordinátákkal tud majd számolni.

Erre az algoritmusra a továbbiakban 2. algoritmusként fogok hivatkozni.

3.3.3 Átlagolást alkalmazó algoritmus

A következő algoritmus is a szülők választásában tér el az előzőektől. Ebben az esetben nem csak egy lehetőséget vizsgál a módszer, hanem minden lehetséges hármas csoportot kiválaszt, akik szülők lehetnek és ezek alapján kiszámolja mindegyikre a koordinátákat. Végül pedig ezen értékek átlagát veszi.

Erre az algoritmusra a továbbiakban 3. algoritmusként fogok hivatkozni.

3.4 Térképek alkalmazása

A szimulációm során a vizsgálandó terület adatait térképekből nyerem ki. Két térképet használok, egyik egy domborzati térkép, a másik pedig egy talajtérkép. Mindkét szempont nagy fontosságú a szenzorok mozgásában, ezzel befolyásolja a feltérképezendő terület bejárását.

A domborzati térképen a RGB (*Red, Green, Blue)* színkódokból olvasom ki a felszíni magasságot. A fehér, azaz a (255, 255, 255) RGB kódú pixel jelöli a legalacsonyabb pontot, míg a fekete, a (0, 0, 0) RGB színkódú pixel jelöli a legmagasabb pontot. Ez alapján a pixelek színkódjait átlagoltam és azok szerint határoztam meg a magasságukat. A térkép felbontása pixel/méterben értendő. így az adott pixelhez tartozó területrész felszíni magassága azonos lesz. Az RGB színkódok átlagolása egy szürkeárnyalatos színt ad. Így ezzel az eljárással egy egységes és mérhető skálát tudtam vizsgálni a magasságértékekre. Ilyen térképre példa látható az alábbi 6. ábrán.

6. ábra
Példa a szimuláció során használt domborzati térképre

A domborzati jellemzők igen nagymértékben számítanak bele a szenzorok haladási sebességébe. Emelkedőn természetesen lassabban, leejtőn pedig gyorsabban tudnak mozogni.

A talajtérkép szintén RGB színkódokat tartalmaz. Itt a barna (185, 122, 87) és szürke (127, 127, 127) színek jelölik a különböző talajtípusokat. A barna szín a homokos talajt mutatja, míg a szürke rész egy apró törmelékes, kavicsos rész ad vissza. Példa a talajtérképre az alábbi 7. ábrán látható.

7. ábra
Példa a szimuláció során használt talajtérképre

A talajtípus azért fontos meghatározni, mert mindegyik fajtája különböző mértékkel lassítja a szenzorokat. Ezt a szimulátorban egy [0; 1] intervallumba eső szorzóval jelöltem.

3.5 Jelterjedési modellek kezelése

A 2.3 fejezetben kifejtett két jelterjedési modellt alkalmaztam a szimulációm során. Amikor egy szenzor adatot küld át egy másiknak, akkor a jel egy valós terepviszonyokkal rendelkező területen halad keresztül. A terepviszonyok befolyásolják a jel csillapítását, ezért ehhez a folyamathoz a Deygout modellt használtam fel. Ezután a vevő szenzor megkapja a jelet, de nem ismeri a befolyásoló terepviszonyokat, ezért ő a szabadtári jelterjedési modell szerint számolja ki a küldő szenzortól való távolságát. Mivel a jel valóságban megtett távolsága eltér a szabadtéri modell szerinti kiszámolt távolságtól, így a szenzor valamelyest arrébb helyezi magát, mint ahogy valójában van. Ekkor történik az a mérési eltérés, ami a későbbiekben hibatorlódáshoz vezet.

Minden pozícionálásnál a két modell által kiszámolt távolságok közötti eltérés miatt a számolt koordináták eltérnek a valós koordinátáktól. Ez az eltérés kezdetben még minimális, ugyanakkor többlépcsős helymeghatározásnál már olyan nagymértékű lehet, hogy adatvesztés léphet fel. Ez abból adódik, hogy a szenzor úgy érzékeli, hogy már nincs olyan közelségben a többiekhez, hogy továbbküldje az adatát és így az adatátvitel nem valósul meg. Ellenben a valóságban még beleférne a hatótávolságba. Másik jelentős probléma, amikor a szenzor úgy érzékeli, hogy még nem lépte át a biztonsági sávhatárt és nem változtat irányt, a valóságban azonban már túllépte, ennek következtében leszakadhat a bolytól.

Mindez azt jelenti, hogy a hibatorlódásból adódó következmények súlyosak lehetnek, ezért tartom fontosnak foglalkozni a továbbiakban a problémával. Ezzel kapcsolatban végeztem vizsgálatokat, amikre majd a 4. fejezetben térek ki részletesen.

3.6 Szimulátor működése

Azért, hogy az előző fejezetben bemutatott szenzorhálózatot megfelelően tudjam vizsgálni, egy szimulációs programot írtam C++ nyelven. A szimulációs programot úgy írtam, hogy minél inkább közelítse egy valós szenzorhálózat működését. Különböző paramétereket értékét lehet változtatni a programban a vizsgálandó eseteknek megfelelően. Ezek egy olyan kimenetet generálnak, amelyekből számszerű értékek kaphatók meg. Ezeket ábrázoltam grafikonokon és elemeztem őket.

3.6.1 Paraméterek kezelése

A programomat úgy írtam, hogy a vizsgálatoktól függően más és más paraméterek összefüggéseit elemezni tudjam. Ezért azokat a paramétereket, amelyeket később vizsgáltam, globális paraméterként vettem fel, így az értékük könnyen állítható, változtatható. Ez azért szükséges, hogy a szimuláció ne csak egy értékre fusson le, hanem egy egész mintasorozatra is. Így kézzel tudtam az adott mintaértéket beállítani. Ezen kívül fontos, hogy ezek a globálisként megadott paraméterek jellemzik a szimulációs környezetet is. Azok a paraméterek, amelyek értékét az adott vizsgálat során nem változtattattam, azok egy-egy

környezeti jellemzőként viselkednek. Fontos, hogy egy szimulációs vizsgálat alatt több mintaérték esetén a környezeti tényezők ne változzanak, ezzel biztosítottam, hogy csak a vizsgált értéktől függjön az eredmény. A globális paramétereim az alábbi táblázatban (3.6.1 táblázat) találhatók. Ezeket az értékeket használtam környezeti jellemzőkként, az éppen aktuálisan vizsgált paramétertől eltekintve, ezek az értékek nem változtak a vizsgálatok során.

A paraméter neve	**Alapértelmezett érték [mértékegység]**
Szenzorok száma	200 [db]
Terület	400 [m^2]
Elmozdulás maximális értéke	4 [m]
Szenzorok hatótávolsága	30 [m]
D pont távolsága	400 [m]
Pixel/méter	0.5 [pixel/m]
Színváltozás	0.1 [m]

3.6.1. táblázat

Globális paraméterek és értékük

A *szenzorok száma (n)* adja meg, hogy összesen hány szenzor alkotja a szenzorhálózatot. Ez a szám magában foglalja a szuperszenzorokat és a mérésre szánt szenzorokat is. Ebből a 200 szenzorból 3 mindig szuperszenzor. Azért választottam ennyi szuperszenzort, mert 3-mal már helymeghatározás végezhető el. Tehát az lenne az ideális, ha 197 sikeres helymeghatározási algoritmus futna le minden lépésben.

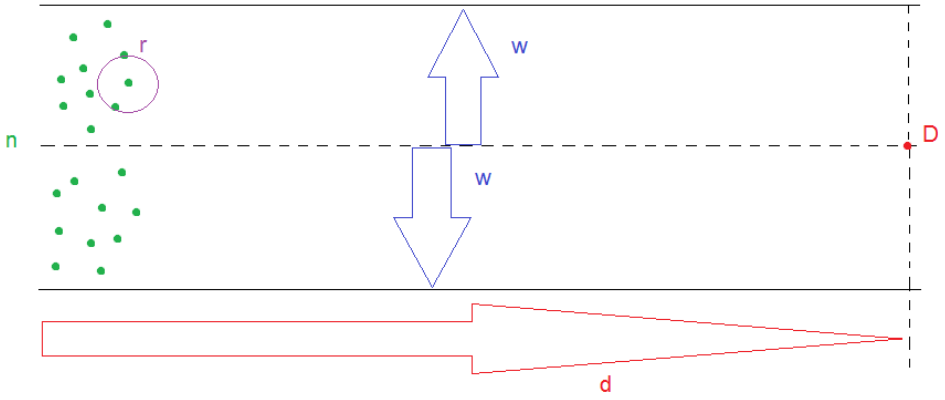
A *terület (w)* azt a felszíni részt foglalja magában, amelyen a szenzorok elhelyezkednek, és amely területet vizsgálnak a naprendszerbeli égitesten. Ezt a területet kell bejárni a szenzoroknak, az itt mérhető adatokra van szükség. Ezzel az értékkel jelöltem ki azt a már fentebb említett sávot is, amely meghatározza, hogy a szenzorok milyen térrészen mozoghatnak még anélkül, hogy könnyen leszakadjanak a bolytól.

Az *elmozdulás maximális értékével* (m) azt a számot határoztam meg, amely egy felső korlátja a mozgásvektor értékének. Azaz egy lépésben maximálisan ekkora lépést tehet meg szenzor. Mivel a szenzorok mozgása kezdetben véletlenszerű irányba indul el, ezért ezt egy randomszám generátor segítségével oldottam meg. Ez alapján a kezdeti elmozdulás értéke 0 és ez a maximális érték közötti véletlenszerű egész számot veszi fel.

A *szenzorok hatótávolsága (r)* az a távolság, amelyen belül egy másik szenzort még lehet érzékelni és így képes a vele való kommunikációra. Ez a hatótávolság minden szenzorra ugyanúgy jellemző, és ugyanazt az értéket kapja. Tehát minden szenzor (a szuperszenzorok is) hatótávolsága megegyezik. Ha két szenzor kívül van egymás hatótávolságán, akkor nem

látják egymást és így nem tudnak kommunikálni sem. Ha egy szenzor olyan messze kerül a szenzorbolytól, hogy egyik másik társa hatótávolságába sem esik bele, akkor mondjuk azt, hogy „leszakadt” a bolytól.

A *D pont távolsága (d)* pedig azt a távot határozza meg, amelyet el kell érnie legalább egy szenzornak, ahhoz, hogy a szimuláció véget érjen. Amíg a terület paraméter a vizsgálandó térség szélességét adta meg, ezzel a hosszát állítottam be. Ezt a távolságú területet kell a szenzoroknak feltérképezniük, itt végzik el a méréseket. Ha egy szenzor is elérte azt az *x* koordináta értéket, ami megegyezik ennek a D pontnak az *x* koordinátájával, akkor a szimulációt befejezettnek tekintem. Ezeknek a paramétereknek az illusztrálása a 8. ábrán látható.



8.ábra

A szenzorhálózat kiinduló környezete

A *Pixel/méter* változó azt adja meg, hogy a beolvasott térképen egy pixel hány métert jelöl. Így a 400 x 400-as felbontású térkép 160000 m^2 területet jelöl.

A *Színváltozás* változónak pedig a térképek beolvasásánál van nagy jelentősége. Ez adja meg, hogy az RGB színkódok átlagolásánál egy egységnyi értéknövekedés hány méter magassági különbséget jelöl. Tehát, ha két pont szürkeárnyalatos színkódjai között 1 egységnyi különbség van, akkor az 0,1 méteres szintkülönbséget jelent.

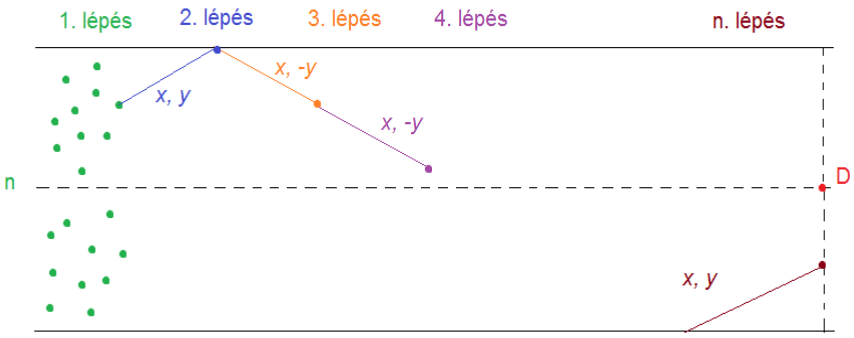
3.6.2 Szenzorok mozgásának szimulálása

A szenzorhálózatot a fenti fejezetekben úgy építettem ki, hogy a benne lévő szenzorok (a szuperszenzorokkal együtt) mozogjanak. Minden szenzor kap egy valós kezdeti koordinátát, ez lesz a kiinduló állapota, innen indul majd a mozgása. A mobilitás megvalósítására egy-egy véletlenszerű számpárt rendeltem minden szenzorhoz. Ezek lettek a kezdeti elmozdulás vektor értékei. Első lépésben a helymeghatározás után ezek az értékek hozzáadódnak a kezdeti koordinátákhoz, így szimulálva a szenzorok elmozdulását. A további lépésekben a szenzorok megtartják a kezdeti irányukat egészen addig, amíg el nem érik a biztonsági sávhatárt. Ha egy szenzor számolt *y* koordinátája eléri a sáv koordinátájának az értékét, akkor a szenzor iránya megváltozik. Az irány *y* koordinátája ebben az esetben a korábbi érték mínusz egyszeresre módosul. Az így kapott elmozdulással halad tovább a szenzor, amíg újra el nem éri a

sávhatárt. Egy lépés egy időegység alatt tevődik meg, tehát esetünkben egy másodperc alatt egyszer adódik hozzá az elmozdulás a szenzor koordinátáihoz. Az elmozdulás koordinátái mindig a valós értékekhez adódnak hozzá. A számolt értékek meghatározásához minden lépés esetében lefut a háromszögelés algoritmusa. Ez azért van, mert a szenzorok (a szuperszenzorokat kivéve) a valóságban nem tudják, hogy merre mozdulnak el, mindig csak a többi szenzor helyzetéhez viszonyítva tudják meghatározni, hogy éppen hol vannak.

A szenzorok egy adott *D* pontot próbálnak meg elérni. A szimuláció addig tart, amíg legalább egy szenzor nem éri el ennek a *D* pontnak a síkját. Ez esetben ez azt jelenti, hogy ha legalább egy szenzor számított *x* koordinátája eléri a *D* pont *x* koordinátáját, akkor vége a szimulációnak.

A mozgás illusztrálása a 9. ábrán látható. A zöld pontok jelölik a szenzorok kiinduló állapotát, a piros *D* pont pedig ez elérendő célt. Az ábrán egyetlen szenzor útját követhetjük végig. A kék színnel megadott *x,y* koordináta jelöli az elmozdulás koordinátáit, az azonos színnel jelölt pont pedig a 2. lépés során elért helyét a szenzornak. Látható, hogy itt eléri a sávhatárt, így a következő (narancssárgával jelölt) lépésben az elmozdulás *y* koordinátája megváltozik. Az *n.* lépés után a szenzor eléri a *D* pont síkját, itt megáll a szimulációs program.



9. ábra

Egy szenzor mozgása a meghatározott területen belül (*x*, *y* koordináta az elmozdulás koordinátái)

3.6.3 Porvihar szimulálása

A porvihar jelensége szintén egy naprendszerbeli égitest lehetséges jellemezője, ezért ezt is építettem bele a szimulációmba. A porvihar egy megadott időben (megadott lépésnél) jelentkezik. A porvihar esetében a szenzorok közötti kapcsolat megszakad, nem látják egymást. Így a jelenség megszűnésekor a kapcsolatot újból fel kell építeni. Ennek az lehet a következménye, hogy több szenzor is leszakadhat a bolytól, hiszen amíg nem volt kapcsolat, nem tudták meghatározni a helyzetüket sem. Ezért könnyen lehet, hogy időközben a szenzorok olyannyira eltávolodnak egymástól, hogy kiesnek egymás hatótávjából. Így azt sem tudják érzékelni, hogy a megadott sávon kívülre érnek. A sávon kívül viszont már nagy eséllyel olyan távol lesznek, hogy nem tudják újból meghatározni a helyzetüket, leszakadnak.

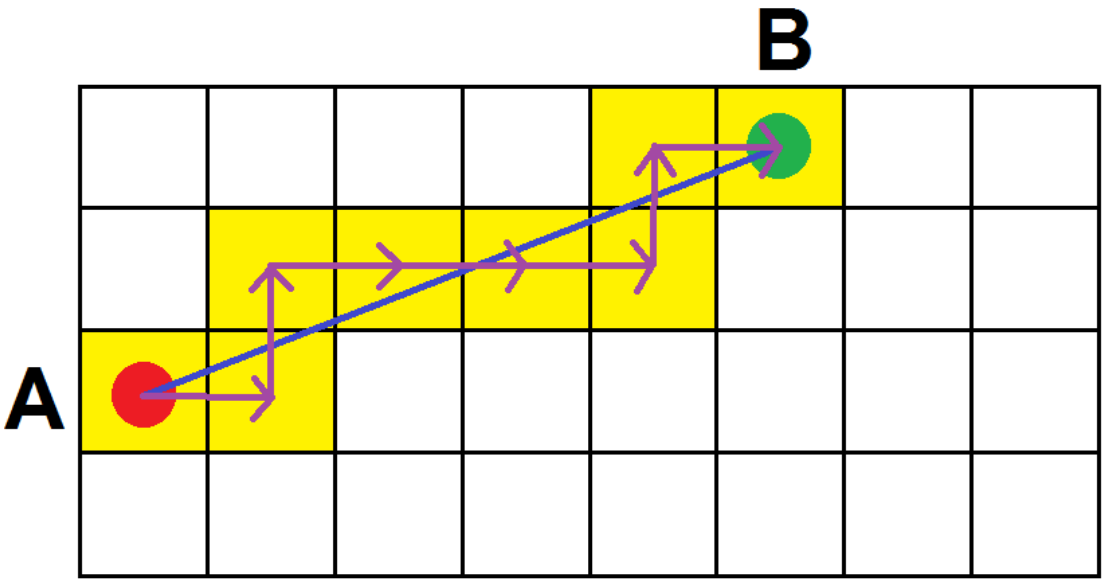
Az általam „porviharnak” nevezett jelenség tehát veszteségekkel jár, ezért tartottam fontosnak megvizsgálni.

A kivitelezése az alábbiak alapján történik. Ha elérkezik a megadott időintervallum kezdete (azaz a szenzor az adott lépéseket hajtja végre), akkor a szenzorokra a háromszögelés függvény helyett egy másik függvény hívódik meg. Ez a másik függvény csak annyit csinál, hogy a szenzor számolt értékéhez hozzáadja még azt az elmozdulás értéket, amelyet a szenzor a porvihar ideje előtt ismert. Azt feltételezem, hogy továbbra is ezzel az iránnyal mozog tovább. Ezáltal az időegységenkénti lépéseket megtartja a szenzor, csak a helymeghatározás módját változtatja meg. Ha esetleg a szenzor megadott koordinátája eléri a biztonsági sáv koordinátáját, azt nem érzékeli a szenzor, halad tovább az eredeti irányba. Ha a porvihar véget ér, akkor ugyanúgy meghívódik a háromszögelés függvénye, mint eddig. Csak itt már nem biztos, hogy mindegyik szenzor megtalálja a másikat úgy, mintha nem lett volna porvihar.

3.6.4 Útvonal meghatározás

A szenzorok elmozdulásánál vizsgálni kell, hogy képes-e a szenzor egyáltalán a következő pontra továbblépni. Ez azért fontos, mert lehetséges, hogy olyan akadály kerül az útjába, ami meggátolja a továbbhaladásban. Ezért ki kell jelölni egy útvonalat a szenzor minden lépésére, és ezt az útvonalat kell vizsgálni a megfelelő szempontok szerint.

Az útvonal meghatározását a 10. ábrán szemléltetem. Az adott szenzor a kiindulópontból (pirossal jelölt pixel) a végpontba (zölddel jelölt pixel) a sárga színnel kijelölt pixeleken keresztül jut el. Számításilag a pixelek középpontját vettem figyelembe, mert így egész értékű koordinátákat tudtam kezelni. Ezért azok a pixelek kerülnek vele az útvonalba, amelyeket valamilyen mértékben is érint a légvonal a kiinduló és végpont között.



10.ábra
Útvonalválasztás szemléltetése

3.6.5 Függőleges elmozdulás

A függőleges elmozdulás a fentiekben említett fontos szempont, ami eldönti, hogy a szenzor tovább tud-e haladni. Döntésem alapján a szenzor 45°-osnál magasabb emelkedőn, illetve -45°-osnál alacsonyabb leejtőn nem képes átjutni. Így ilyen esetekben a szenzor mozgása leáll. Ez azt vonja maga után, hogy ezt a szenzort elveszítjük a szimuláció során, mert bizonyos idő elteltével, ha a többi szenzor már továbbhaladt, leszakad a bolytól.

Kiemelkedő szerepe van a meredekségi szögnek a szenzorok haladási sebességének meghatározásánál is. Értelemszerűen minél magasabb emelkedőn szeretne feljutni a szenzor, annál lassabban mozog, és minél nagyobb leejtőn halad tovább, annál gyorsabban mozog.

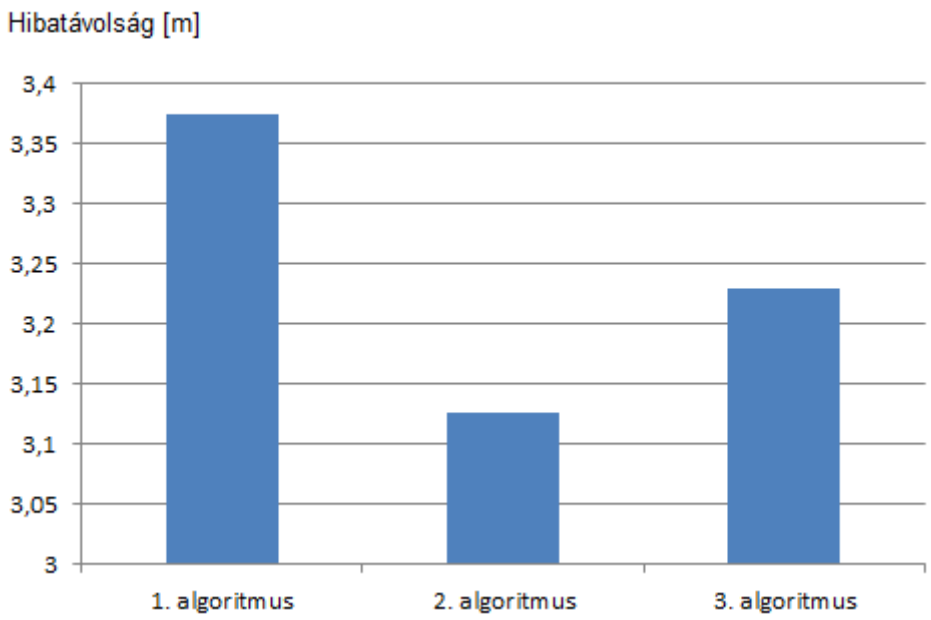
4. Eredmények

A szimuláció során az előző fejezetben lévő 3.6.1 táblázatban ismertetet paraméteres értékeket vettem alapul. Ezek az állandó értékek, kivéve, ha a vizsgálati szempontok miatt szükség van a változtatására. Ilyen esetben az eredmények ismertetésénél ezt jelzem.

Minden esetben, minden vizsgálandó paraméter értékre vizsgálattól függően legalább tízszer futtattam le a szimulációs programot. Erre azért volt szükség, mert a véletlenszerűen generált számok minden futás alkalmával más értékeket kaptak és a többszörös lefuttatás estén így egy átlagolt értéket kaphattam. A kapott eredményeket ezért a vizsgálandó paraméterenként átlagoltam és azzal dolgoztam tovább, az ábrázolásban is ezek szerepelnek. Így sokkal általánosabb eredményeket vizsgálhattam.

4.1 Hibatávolságok vizsgálata a módszerek függvényében

Ebben az esetben azt vizsgáltam, hogy a három algoritmusnál hogyan alakulnak a hibatávolságok. A hiba itt a valós koordináták és a helymeghatározó algoritmussal kiszámolt koordináták közötti eltérést jelöli méterben kifejezve. A hibákat a szenzorokra és a megtett lépésekre nézve is átlagoltam, így egy átlátható képet kaptam az eredményekről.



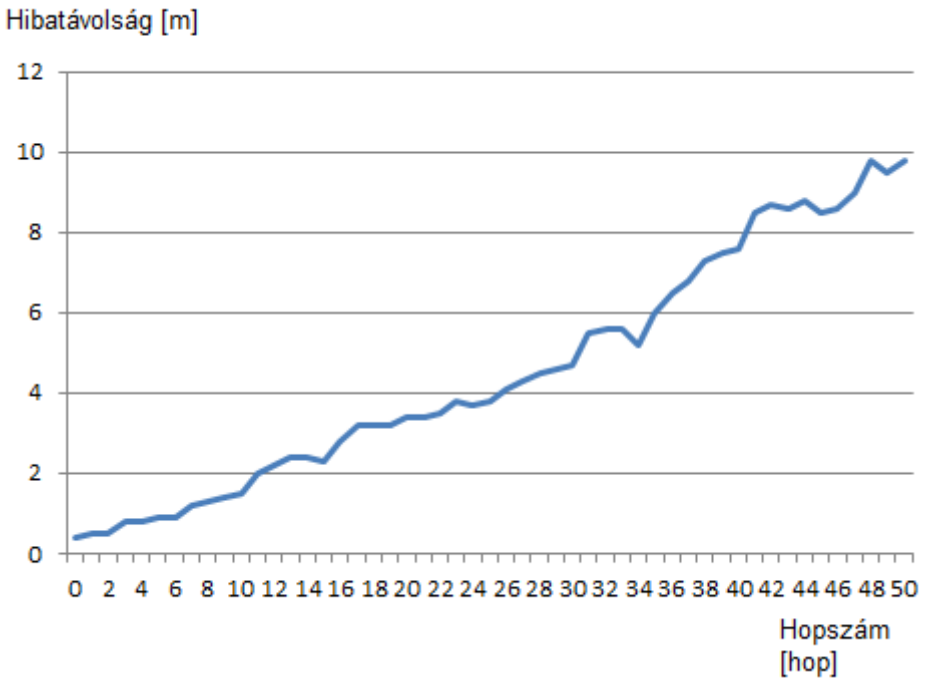
11.ábra
Hibatávolságok változása az algoritmusok függvényében

A 11. ábrán látható, hogy az optimalizálatlan első algoritmus átlagban majdnem 3,4 méteres hibával dolgozik. Ehhez képest a második, hopszámokat alkalmazó algoritmus jóval kisebb, ~3,1 méteres hibákat generál, míg az utolsó átlagolást alkalmazó algoritmus a kettő közé lő be ~3,2 méteres hibával. Leolvasható, hogy mindkét javított módszer érzékelhetően jobb eredményt hoz. A kisebb hibatávolság kevesebb leszakadó szenzorral is jár, hiszen kevesebb eséllyel lépik túl észrevétlenül a

biztonság sávot, illetve az adatvesztés lehetősége is csökken. A leghatékonyabb algoritmus ebből a vizsgálati szempontból a 2. algoritmus. Ez azzal magyarázható, hogy ebben az esetben mindig a lehető legpontosabb értékekkel történik a számítás, mindig a legbiztosabb helyzetű szenzorok segítségével történik a pozícionálás.

4.2 A hibatávolság változása a hopszámok függvényében

A következő vizsgálatommal a hibatávolságok változását néztem meg a hopszámok növekedésének függvényében. A hopszámokból adódóan ez a vizsgálat a 2. algoritmusnál értelmezhető, hiszen csak ennél alkalmaztam őket.

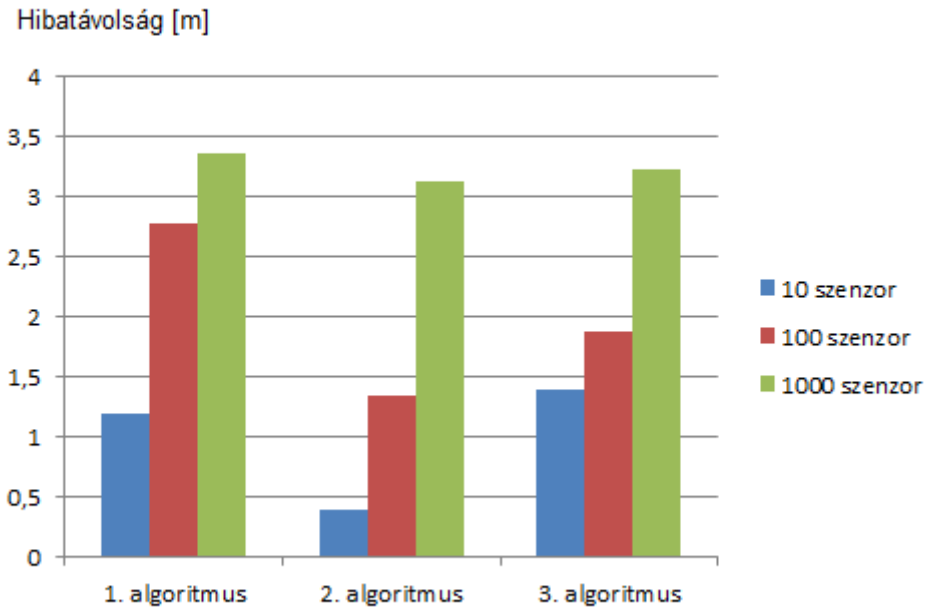


12.ábra
Hibatávolság változása a hopszámok függvényében

Mint az a fenti 12. ábrán is látható, a hopszám növekedésével a hibatávolság is növekszik. Ez az eredmény várható volt, hiszen minél nagyobb hopszámú szenzorok vannak jelen a számítások során, annál pontatlanabb eredményt kapunk. A kisebb hopszámok esetében a torlódott hiba még kisebb, így a valós és becsült koordináták között is kisebb az eltérés. Ez a vizsgálat azért volt jelentős, mert ez az előző vizsgálati eredménytől függetlenül bizonyítja, hogy a kevesebb lépcsőben számolt szenzorok nagyobb pontosságot biztosítanak, így a 2. algoritmusom megalapozottan hatékonyabbnak minősíthető, mint az optimalizálatlan 1. algoritmus.

4.3 Hibatávolság változása a szenzorok számának függvényében

Ez a vizsgálatom abból a célból készült, hogy megfigyelhető legyen, hogyan alakul a hibatávolság átlagos értéke akkor, ha a szenzorok száma változik. Ez az eredmény mindhárom algoritmusnál értelmezhető, így egy grafikonon ábrázoltam a kapott értékeket az átláthatóság miatt.

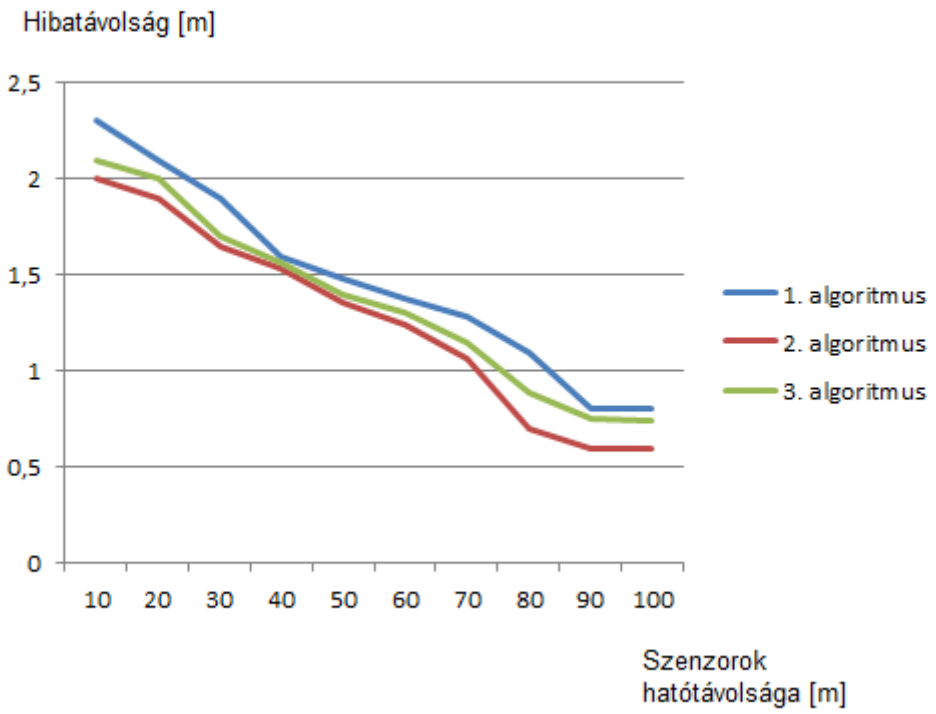


13.ábra
Hibatávolság változása a szenzorok számának függvényében

A mérési eredményeket a 13. ábra mutatja. Az 1. algoritmus esetében, ha növekszik a szenzorok száma, akkor egyre kisebb valószínűséggel lesz benne a három szuperszenzor az elsőnek kiválasztott szülők között. Így a kiszámolt koordináták is pontatlanabbá válnak. A 3. algoritmus értéke kevés (10) szenzorra az 1. algoritmus értéke felett van. Ez azt jelenti, hogy kevés szenzorszám esetén nagyobb átlagos hibát kapunk, ez viszont nagyobb szenzorszám esetében megfordul. Ennek az oka, hogy a 3. algoritmus esetében a kiugróan magas eltérések elhúzzák az átlagolt koordináta értékeket rossz irányba, míg az 1. algoritmus esetében kevés szenzornál nagyobb a valószínűsége, hogy a szülők maguk a szuperszenzorok lesznek. Itt is látható, hogy a 2. algoritmus a hatékonyabb a három közül.

4.4 Hibatávolság változása a szenzorok hatótávolságának függvényében

Azt vizsgáltam ebben az esetben, hogy a szenzorok hatótávolságának változása hogyan hat a hibatávolság átlagos értékére. Itt is mindhárom algoritmusra értelmezhető a vizsgálat, így ezek görbéi egy grafikonon, a 14. ábrán láthatóak.



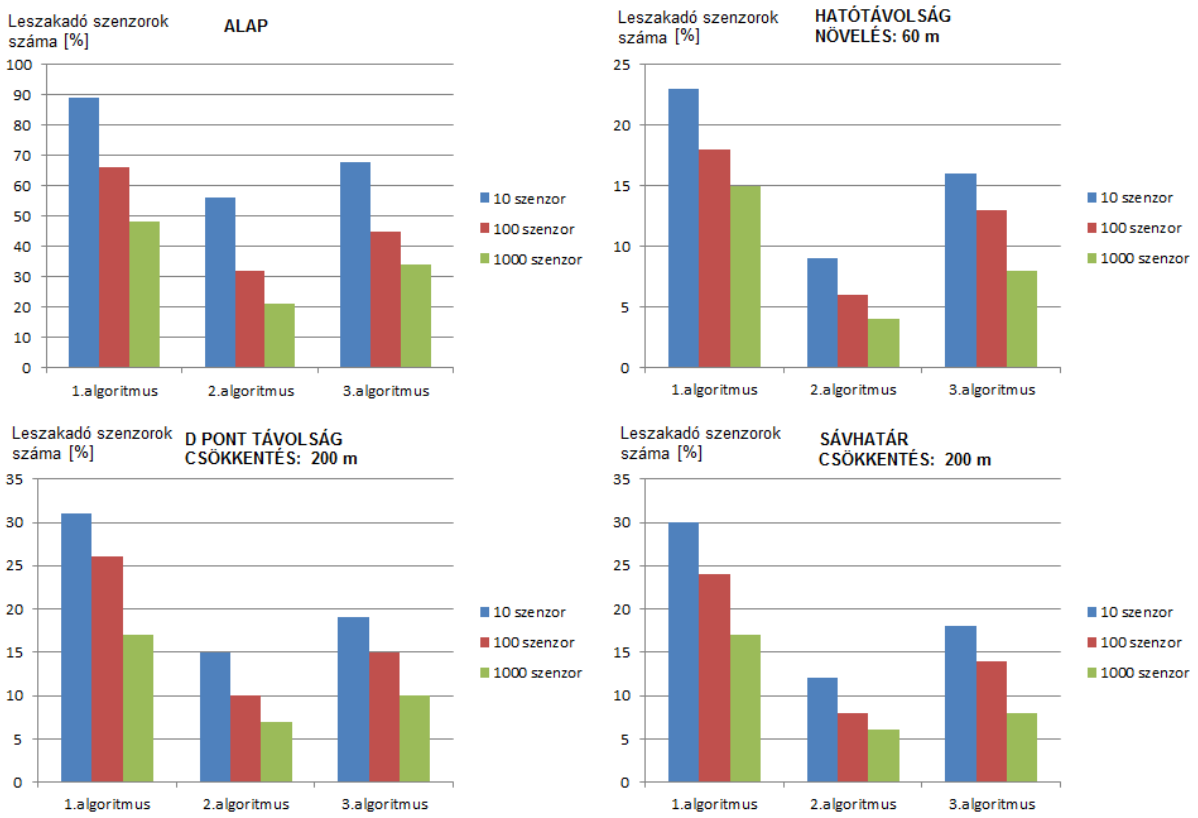
14.ábra
Hibatávolság változása a szenzorok hatótávolságának függvényében

Mindegyik esetben egy csökkenő meredekségű görbe figyelhető meg. Mivel a nagyobb hatótávolság azt jelenti, hogy több szenzor fér bele egy másik hatótávjába, így ez azzal is jár, hogy több szenzor közül választhat szülőket. Ez az összes bemutatott algoritmusnál hibaérték csökkenéssel jár. A 2. algoritmus esetében lesz leginkább szembetűnőbb ez a csökkenés, ekkor a hatótáv növelésével egyre több szenzor választhatja a kisebb hibával rendelkező szenzorokat, vagy akár a szuperszenzorokat is szülőjévé. Jól látható, hogy a 2. algoritmus generálja a legkevesebb hibát a három algoritmus közül a hatótávolság függvényében.

4.5 A leszakadó szenzorok száma az algoritmusok függvényében

A következő vizsgálatomnál azt néztem meg, hogy hány leszakadó szenzor lesz a kijelölt *D* célpont eléréséig az algoritmusok függvényében.

Ebben az esetben a kezdeti alap értékekkel indítottam a szimulációt. Ennek eredményeképpen olyan értékeket kaptam a leszakadó szenzorok számára, amelyek túl magasak voltak ahhoz, hogy megérje működtetni egy ilyen paraméterekkel rendelkező szenzorhálózatot. Ezért kiemeltem három olyan jelentős paramétert, amelyek leginkább befolyásoló hatással bírtak az eredményekre. Ezek a paraméterek, a hatótávolság, a *D* célpont távolsága a kiinduló ponttól és a sávhatár szélessége. Ezeket növelve vagy éppen csökkentve a megfelelő arányban már olyan eredményeket kaptam, amik elfogadhatóak egy ilyen szenzorhálózat működése során.

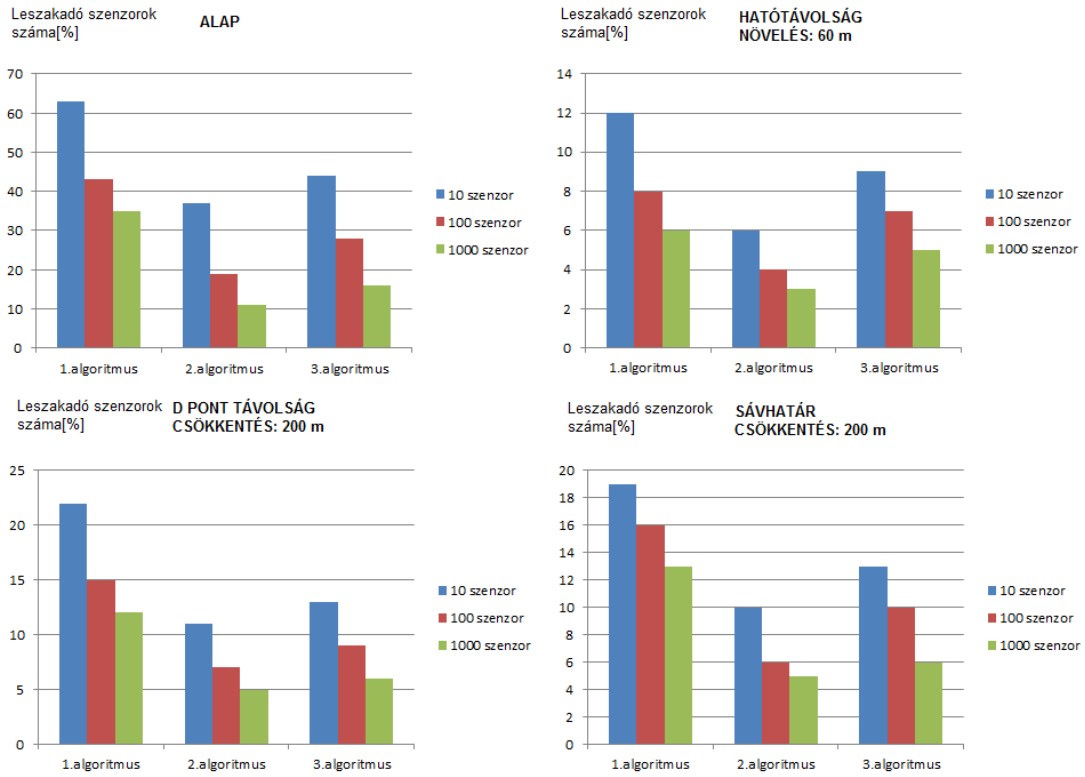


15.ábra
Leszakadó szenzorok száma az algoritmusok függvényében

A fenti 15. ábrán látható a kapott négy grafikon. Az első azokat az eredményeket ábrázolja, ahol a 3.6.1 táblázatnak megfelelő paraméterekkel futott a szimuláció. Ekkor mindhárom algoritmus esetében elegendően kevés szenzorra nézve van olyan opció, amikor több mint 50%-a leszakad a szenzor bolynak. Ez elfogadhatatlanul nagynak bizonyult. A mellette lévő grafikonon már egy módosult paraméteres eredmény látható. Itt a szenzorok hatótávolságát növeltem 30m-ről 60m-re. Az tapasztalható, hogy itt már mindhárom esetben a szenzorok kevesebb, mint 25%-a szakad csak le. Ez az előző értékekhez viszonyítja jelentős változást jelent. Azért kaptam ilyen eredményeket, mert ha a hatótávolságot növelem, akkor jóval több szenzor látja egymást, mint az előző esetben, így több lehetséges szenzor képes meghatározni egymás helyzetét. Ezek alatt látható az a grafikon, ahol a D célpont távolságát csökkentettem le 400m-ről 200m-re. Ez a megoldás valamivel rosszabb eredményeket adott, mint az előző, de még így is számottevően jobbnak bizonyult, mint az alapeset. Ez azért lehetséges, mert itt ugyan csökken a távolság, amit be kell járniuk a szenzoroknak, de még elég széles ahhoz, hogy a biztonsági sávhatáron túlra kerüljenek. Az utolsó grafikonon a sávhatár szélességét csökkentettem le 400m-ről 200m-re. Itt is hasonló eredményeket kaptam, mint az előző esetben, amikor a *D* pont távolságát változtattam. Ennek az az oka, hogy bár a sávhatárt lecsökkentettem és ez javított az alapértékeken, de a bejárandó terület hosszúsága nem változik. Így ha a sávhatáron nem is mozdulnak kívülre a szenzorok, de lemaradás miatt leszakadhatnak a bolytól.

Az algoritmusok összehasonlítva megállapítható, hogy itt is a 2. algoritmus a hatékonyabb. Hiszen ha a hibatávolság kicsi (a minimális hopszámok alkalmazása miatt), akkor a leszakadó szenzorok száma is alacsony lesz. Pontosabb számított koordinátákkal elkerülhető a leszakadás, könnyebben megakadályozható a biztonsági sávhatár átlépése.

Ennek a vizsgálatnak az eredményeit érdekesnek tartottam összehasonlítani ugyanilyen paraméterekkel, ugyanakkor sík terepen végzett szimuláció eredményeivel. Itt látható, hogy mennyire sokat befolyásol a leszakadó szenzorok számán a domborzat alakulása. Egy távoli égitest felszíni jellemzői közé tartoznak a dűnék, a kráterek, a törmelékes területek is. Ezek kisebb, illetve nagyobb mértékben lassítják a szenzorokat, vagy akár meg is állíthatják őket. Ha egy emelkedő olyan magas, hogy a szenzor nem tud túljutni rajta, akkor mozgása leáll. Ugyanakkor, ha egy leejtő is lehet olyan meredek, hogy a szenzor károsodik az esés közben és ezért nem tud továbbhaladni.

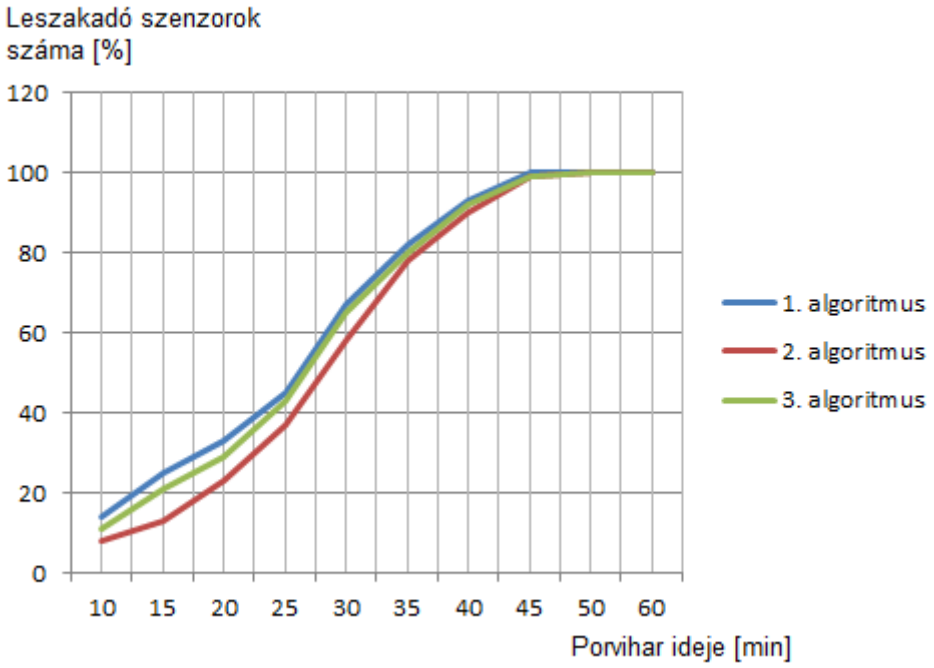


16.ábra
Leszakadó szenzorok száma sík terepen az algoritmusok függvényében

A 16. ábra mutatja, hogy miként alakulnak az előző mérések egy olyan terepen, ahol semmilyen felszíni akadályozó tényező nincsen. Megfigyelhető, hogy a leszakadó szenzorok száma nagymértékben lecsökkent az előzőekhez képest. Hiszen itt már csak a biztonsági sávhatáron való túllépés és a bolytól való lemaradás adja a leszakadt szenzorok számát. Ez a vizsgálat bár szebb eredményeket hozott, de nem jellemzi valósághűen a feltérképezendő bolygó felszínét.

4.6 Leszakadó szenzorok száma a porvihar idejének függvényében

Az alábbi vizsgálat megmutatta, hogy a porvihar idejétől függően a szenzoroknak hány százaléka szakad le a vihar alatt a különböző algoritmusok használata esetén. Ezzel az előző mérés mellett egy további tényezőt mutatok be, amely egy olyan távoli égitestre, mint pl. Mars jellemző

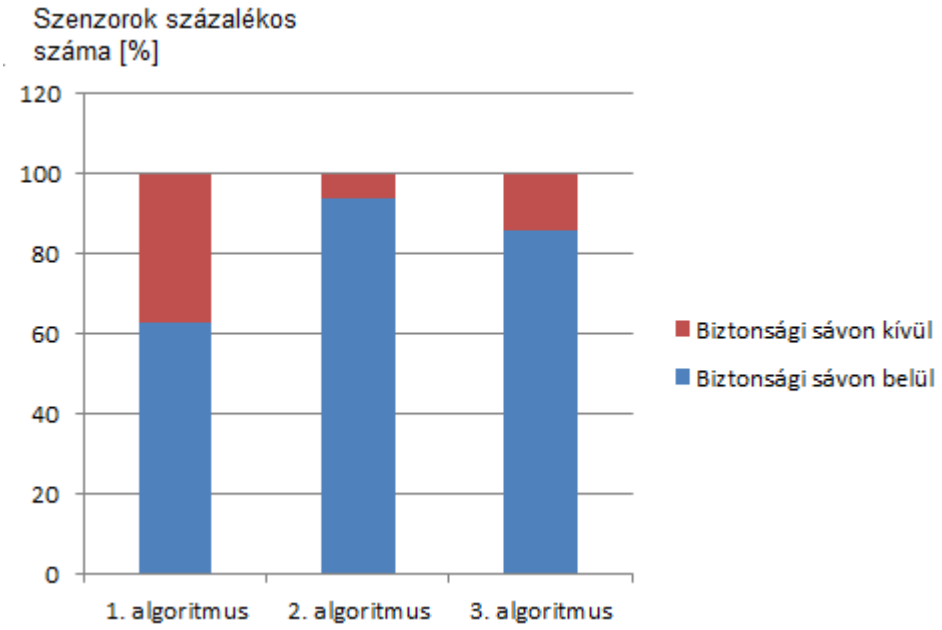


17.ábra
Leszakadó szenzorok számának változása a porvihar időtartamának függvényében

Ahogy a 17. ábrán is látható mindhárom algoritmus esetében minél tovább tart a porvihar, annál több szenzor szakad le. Közel egy óra időtartamú vihar estében már az egész szenzor boly elveszik. Látható, hogy az algoritmusok hasonló léptékben veszítik el a szenzorjaikat. Kezdetben, ~ 30 percig még észrevehető minimális eltérés, de később már egybefüggnek az eredmények. Ez azért van, mert kezdetben még kevesebb idejük van a szenzoroknak eltávolodni a sávhatáron túlra, így a sávhatár mellett lévő referencia szenzorokkal még visszahozhatóak a bolyba. Ekkor van még jelentősége, hogy melyik algoritmust alkalmaztam. Később azonban ez már nem számít, a szenzorok olyan messzire kerülnek, hogy már nem lehet őket visszahozni a határon lévő referencia szenzorokkal sem. Ebben az esetben is a 2. algoritmus mutatkozik hatékonyabbnak az adott vizsgálati feltételek mellett.

4.7 Szenzorok százalékos száma a biztonsági sávon kívül és belül a különböző algoritmusokra

Végül azt vizsgáltam, hogy a szenzorboly hány százaléka van átlagosan a biztonsági sávhatáron beül, illetve kívül a különböző algoritmusokra. Azok a szenzorok, akik a biztonsági sávhatáron kívül vannak, még nem biztos, hogy le is szakadnak, hiszen egy megadott tartományból még vissza lehet őket hozni, ha van minimum három referencia szenzor a sávhatár mellett közvetlenül. Ugyanakkor a biztonsági sávhatáron kívül eső szenzorok nem teszik ki az egész részét a leszakadó szenzoroknak, hiszen ebbe azok nem tartoznak bele, akik a bolytól való lemaradás, akadályba ütközés miatt esnek ki.



18.ábra
Szenzorok eloszlása a biztonsági sávon belül és kívül a három általam használt algoritmusra

A fenti 18. ábrán az oszlopdiagram mutatja a százalékos értékeket. A 2. algoritmus alatt jóval több hányada a szenzoroknak esik bele a sávhatárba, hiszen itt a pontosabb számítások miatt az irányváltó eljárás nem engedi ki őket a sávhatáron kívülre. Ezzel szemben az 1. algoritmus esetében már meghatározó rész kerül a biztonsági sávon kívülre.

5. Összefoglalás

A távoli égitestek feltérképezése egyre nagyobb szerepet kap nemcsak nemzetközi tekintetben, hanem hazánkban is. A gyors fejlődés maga után vonja az űreszközök fejlesztését is, ami egyre részletgazdagabb és pontosabb eredményeket produkál. Ezek inspiráltak arra, hogy kifejtsem a szenzorhálózatok alkalmazhatóságát egy ilyen feladaton belül.

Egy olyan szenzorhálózatot feltételeztem a munkám során, amiben fontos szerepet kap a helymeghatározás, hiszen a mért adatok megkívánják, hogy a mérési helyzetüket ismerjük és képesek legyünk kalkulálni velük. A pozícionálásra a háromszögelés módszerét alkalmaztam, amelyet matematikai egyenletekkel és geometriai ábrákkal szemléltettem.

Munkám során kialakítottam több olyan algoritmust, amelyet követve az szenzorhálózat tagjai képesek egymáshoz viszonyított helyzetüket meghatározni. A becslési eljárások azonban nem adnak pontos eredményeket, hiszen mérési és számítási hibákkal mindig kell számolnunk. A mérések során a hibák meglétére és torlódására fókuszáltam, ami azt eredményezte, hogy meg tudtam állapítani egyes összefüggéseket az alkalmazás hatékonyságának terén, illetőleg alapot adott az algoritmusok összehasonlítására. A hibatorlódás többlépcsős helymeghatározás esetén egyre nagyobb mértékben van jelen és egyre inkább eltérő eredményeket ad a valós helyhez viszonyítva. Ez olyan problémákat vet fel, amelyek veszteségekkel is járhatnak. A bolytól leszakadó szenzorok számának, az eredmények pontosságának, a számítási hibák nagyságának és a pozíciók kiszámíthatóságának változásait vizsgáltam.

Az eredményeim elemzése egy általam írt szimulációs programmal történt. A kapott számszerű adatokat nem csak szövegesen értelmeztem, hanem grafikonokon is szemléltettem, így részletesebb és kézzelfoghatóbb képet kaptam a választott szenzorhálózat helymeghatározó algoritmusainak működéséről, hatékonyságáról. A szimulációs mérések azt bizonyították, hogy a javasolt többlépcsős helymeghatározás során a 2. algoritmus bizonyult a leghatékonyabbnak. A kapott eredmények felhasználásával könnyen lehet növelni egy idegen égitesten üzemeltetett szenzorhálózat hatékonyságát.

Nemcsak a szenzorhálózat által adott helyzeteket vizsgáltam, hanem a távoli égitesten felléphető jelenségre is kitértem. Olyan általános jellemzőt választottam, amelyek a legtöbb égitesten előfordulhat. A jelenség következményei között szerepel a szenzorok közötti kommunikáció megszakadása, az egymás hatósugarától való távolodás és ahhoz való közeledés, illetve a bolytól leszakadó szenzorok megléte is.

Dolgozatomban igyekeztem törekedni arra, hogy aktuális problémát közelítsek meg és ennek a problémának minél részletesebb és hatékonyabb vizsgálatára törekedjek.

Ábrajegyzék

1. ábra: A Mars felszíne egy 2012. szeptember 28.-án készült felvételen
2. ábra: Fresnel zóna
3. ábra: Deygout modell
4. ábra: A háromszögelés geometriája
5. ábra: A hopszám számítása
6. ábra: Példa a szimuláció során használt domborzati térképre
7. ábra: Példa a szimuláció során használt talajtérképre
8. ábra: A szenzorhálózat kiinduló környezete
9. ábra: Egy szenzor mozgása a meghatározott területen belül x, y koordináta az elmozdulás koordinátái
10. ábra: Útvonalválasztás szemléltetése
11. ábra: Hibatávolságok változása az algoritmusok függvényében
12. ábra: Hibatávolságok változása a hopszámok függvényében
13. ábra: Hibatávolság változása a szenzorok számának függvényében
14. ábra: Hibatávolság változása a szenzorok hatótávolságának függvényében
15. ábra: Leszakadó szenzorok száma az algoritmusok függvényében
16. ábra: Leszakadó szenzorok száma sík terepen az algoritmusok függvényében
17. ábra: Leszakadó szenzorok számának változása a porvihar időtartamának függvényében
18. ábra: Szenzorok eloszlása a biztonsági sávon belül és kívül a három általam használt algoritmusra

Irodalomjegyzék

[1] John Broph et al.: "*Asteroid Retrieval Feasibility Study*". Keck Institute for Space Studies, California Institute of Technology, Jet Propulsion Laboratory, 2013.
[2] G. Wang, G. Cao, and T. Porta: *Movement-assisted sensor deployment*, in Proceedings of. IEEE INFOCOM'04, Hong Kong, 2004.
[3] Vincze Zoltán, Vida Rolland, Budapesti Műszaki és Gazdaságtudományi Egyetem, Távközlési és Médiainformatikai Tanszék: *Mobil eszközök alkalmazása szenzorhálózatokban*, http://web.tmit.bme.hu/~vida/cv/hiradastechnika_v6.pdf, 2007. június *(Utolsó látogatás: 2013. október 21.)*
[4] G. Wang, G. Cao, and T. Porta: *Movement-assisted sensor deployment, in Proceedings of. IEEE INFOCOM'04*, Hong Kong, 2004.
[5] Haslett, C.: *Essentials of Radio Wave Propagation*. Cambridge (UK): Cambridge University, Press, 2008.
[6] Horváth Zoltán: *A Terepdomborzat hatása a kis és közepes magasságon feladatot végrehajtó pilóta nélküli repülőgép kommunikációs csatornájának stabilitására*, Hadmérnök: katonai műszaki tudományok online (2006-) IV. évfolyam 3. szám 2009. szeptember
[7] Bori Zoltán: *Beltéri helymeghatározó rendszerek optimalizálása*, Budapesti Műszaki és Gazdaságtudományi Egyetem, 2011, Budapest
[8] http://www.zytrax.com/tech/wireless/fresnel.htm, *(Utolsó látogatás: 2013. október. 22)*
[9] Z. Butler and D. Rus: *Event-based Motion Control for Mobile Sensor Networks*, IEEE Pervasive Computing, Vol. 2, No. 4, pp. 34–42, Oct.-Nov. 2003.