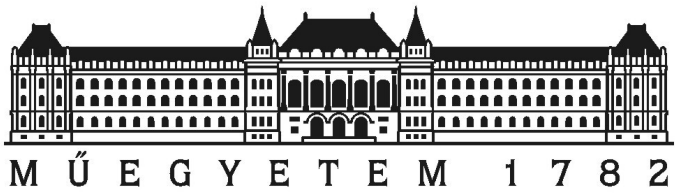


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Szélessávú Hírközlés és Villamosságtan Tanszék

A Föld körüli, űrbéli rádiófrekvenciás szennyezettség ábrázolása a SMOG projekt mérései alapján

TDK-dolgozat

Markotics Boldizsár, Takács Donát

Konzulens: Dr. Dudás Levente

2020. október 26.

Tartalomjegyzék

Kivonat

2019. december 6-án állt pályára a műegyetemi fejlesztésű második és harmadik magyar műhold, a SMOG-P és az ATL-1. Mindkettő fedélzetén helyet kapott egy-egy rádiófrekvenciás spektrumanalizátor, melyek segítségével első alkalommal került megmérésre a Föld körüli pályán érzékelhető, a felszíni digitális TV adók által az űrbe kisugárzott rádiófrekvenciás jelek erőssége. Dolgozatunkban bemutatjuk, hogy hogyan készítettünk a nyers, űrből kapott adatokból két- és háromdimenziós rádiószmog-térképeket.

A műholdak kis mérete és korlátozott teljesítménye miatt azokon nem kapott helyet orientációszabályozó-rendszer, a fedélzeti antenna pedig jelentős iránykarakterisztika fluktuációval rendelkezik, ezért erősen szelektálni kellett a mért adatokat. A mérési pontok térbeli helyzetét, sűrűségét a műhold pályája és a földi állomással való kommunikáció sávszélessége is korlátozta, így jelentős térbeli interpolációra is szükség volt. A távolsággal inverz négyzetesen súlyozott interpolációt a mért RSSI értékek frekvenciatartománybeli átlagából visszaszámolt teljesítmény-értékeken végeztük. Ennél figyelembe kellett venni, hogy a mért pontok gömbfelületen helyezkednek el, így az interpoláció egy megfelelően sűrű ikozahedrális hálón történt.

Továbbiakban az immáron szűrt, interpolált adatok látványos megjelenítése volt a cél. Ezt egy JavaScript technológián alapuló webalkalmazás segítségével értük el. Ez a feldolgozott adathalmazt alapján különféle módokon (diszkrét mérési pontok, folytonos interpolált eloszlás, kontúrvonalak stb.) tudja megjeleníteni. Az oldal interaktív, így a felhasználó képes frekvencia szerint szűrni az eredményeket, valamint képes befolyásolni a megjelenő objektumok egyes tulajdonságait. A megjelenítéshez a webGL alapú THREE.js nevű könyvtárat vettük igénybe.

Munkánk eredményeként egy teljes adatfeldolgozási lánc készült el, amely a Föld körül keringő zseb-műholdak által szolgáltatott nyers mérési eredményekből a saját fejlesztésű program és weboldal segítségével jeleníti meg a Földet körülvevő űrbéli rádiófrekvenciás szennyezettség legjobb jelenlegi becslését. A kifejlesztett környezet bővíthető, de már jelen állapotában is képes a jövőben indítandó, hasonló misszióval rendelkező műholdak adatainak feldolgozására.

Abstract

On December 6, 2019, the second and third Hungarian satellites, SMOG-P and ATL-1 (both having been developed at BME) were launched. They both had a radio frequency spectrum analyzer on board, which was used to measure for the first time the strength of radio frequency signals radiated into space by terrestrial digital TV transmitters – that can be detected in orbit around the Earth. In our paper, we show how we made two- and three-dimensional radiosmog maps from raw data received from space.

Due to the small size and limited power of the satellites, they did not have an orientation control system, and the on-board antenna has significant directional variation in gain, thus the measured data had to be strongly filtered. The spatial position and density of the measurement points were also limited by the orbit of the satellite and the bandwidth of the communication with the ground station, so significant spatial interpolation was also required. Inverse square distance weighted interpolation was performed on power values calculated from the frequency domain average of the measured RSSI values. In doing so, it had to be taken into account that the measured points are located on a spherical surface, thus the interpolation was done on a sufficiently dense icosahedral mesh.

Further, the goal was to display the now filtered, interpolated data in an effective, visually appealing way. This was achieved using a web application based on JavaScript technology. It can display the processed data set in various ways (discrete measurement points, continuous interpolated distribution, contour lines, etc.). The page is interactive, so the user can filter the results by frequency, as well as be able to influence certain properties of the displayed objects. The webGL-based directory THREE.js was used for the display.

As a result of our work, a complete data processing chain was created, which uses a self-developed program and website to display the best current estimate of space radio frequency pollution surrounding the Earth from raw measurement results provided by pocketcube satellites orbiting the Earth. The developed environment can be expanded, but even in its current state it is able to process the data of satellites with a similar mission, to be launched in the future.

1. fejezet

Bevezetés

Jelen dolgozat célja, hogy bemutassuk, hogyan hoztunk létre a műegyetemi fejlesztésű SMOG-P és ATL-1 műholdak által mért adatokból – a világon első alkalommal – egy Föld körüli, űrbéli rádiófrekvenciás szennyezettség-térképet („rádiószmog-térképet”), és egy, azt interaktívan megjelenítő webes alkalmazást. A mérések feldolgozása számos kihívás elé állított minket, melynek egy része a műholdak technikai adottságaiból származik, ezért is érdemes bemutatnunk röviden a két releváns missziót, így kontextusba helyezve a munkánkat.

1.1. PocketQube műholdak

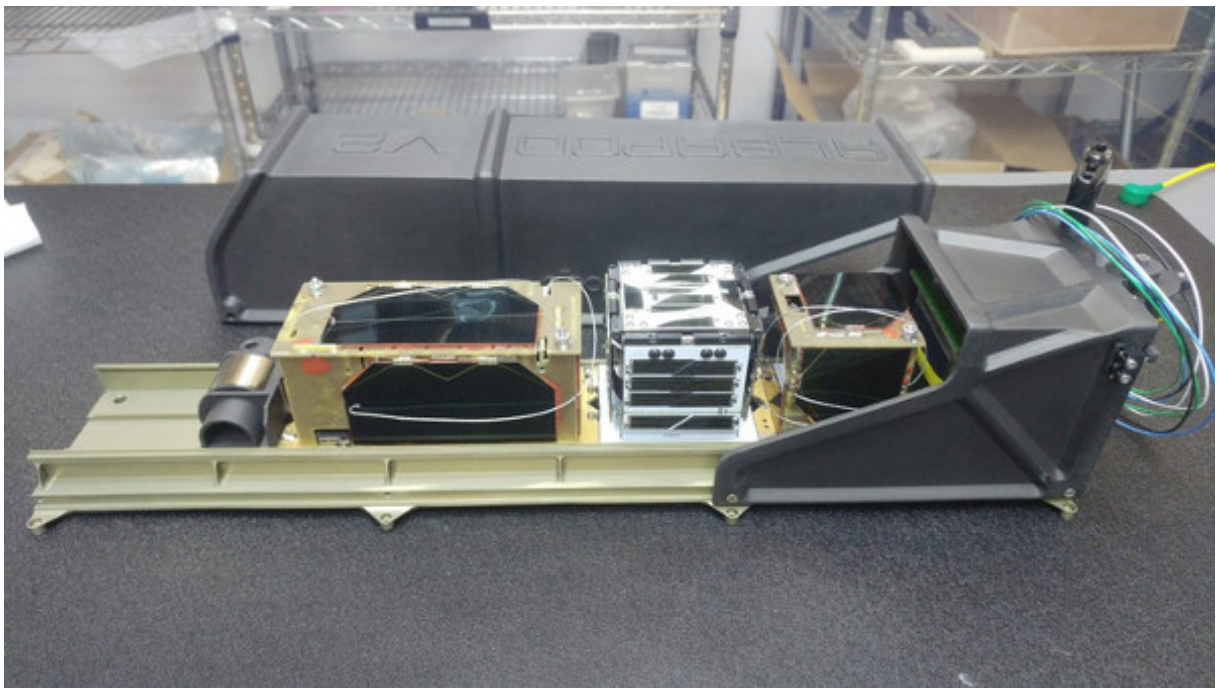
A SMOG-P és az ATL-1 műholdak több tekintetben is újabb jelentős mérföldkőnek számítanak a magyar űrtevékenységben. A SMOG-P a világ első űrben sikeresen pályára állított és működő 1 PQ méretű műholdja [23], egyben a második magyar műhold is.

Az $5 \times 5 \times 5$ cm-es 1 PQ (PocketQube) méretű műholdakat a $10 \times 10 \times 10$ cm-es CubeSat műholdak miniatürizációs törekvéseinek továbbgondolásaként javasolta az amerikai Robert Twiggs professzor [21]. Ezek a nanoműholdak lehetővé teszik, hogy még alacsonyabb költségen léphessenek ki saját műholddal az űrbe egyetemek, kutatóintézetek vagy egyéb érdeklődők; ugyanakkor jelentős kihívásokat is támaszt a fejlesztőkkel szemben: 10 évig tartott, mire Twiggs professzor felvetése után pályára állt a koncepciót először sikeresen megvalósító SMOG-P.

A 2 PQ méretű (azaz $5 \times 5 \times 10$ cm méretű) ATL-1 pedig az első pályára állított magyar kereskedelmi célú műhold.

1.2. Az SMOG-P és az ATL-1 elsődleges küldetése

A SMOG-P elsődleges küldetése a Föld körüli rádiófrekvenciás szennyezettség feltérképezése volt. Számtalan rádiófrekvenciás telekommunikációs eszköz működik a Föld felszínén, melyeket csupán a földi kommunikációban használnak, azonban a kibocsájtott rádióhullámok az űrbe is kijutnak. Ez egyrészt zavarhatja a Föld körüli pályán keringő műholdakkal való kommunikációt; másrészt a földi szolgáltatók irányából is hasznos lehet tudni, hogy hol pazarolnak teljesítményt arra, hogy az űrbe sugározzanak. Ennek ellenére eddig nem



1.1. ábra. Az ATL-1 (balra) és a SMOG-P (jobbra) a start előtt. (Kép forrása: Alba Orbital)

készült még átfogó mérés erről; így a SMOG-P célja az volt, hogy mérései alapján elkészülhessen az első ilyen térkép. Ehhez helyet kapott a SMOG-P fedélzeten egy spektrométer [5], ami a műhold elsődleges hasznos terhe.

Az ATL-1 elsődleges küldetése a magyar ATL (Advanced Technology of Laser) cég által fejlesztett szigetelőanyag tulajdonságainak vizsgálata volt [22]. Több akkumulátor is került a fedélzetre, melyek felületén a vizsgált szigetelőanyag és hőmérők helyezkedtek el. Ezen túl helyet kapott az ATL-1 fedélzetén a SMOG-P-ben használt spektrométer is: így két műhold méréseiből dolgozhattunk.

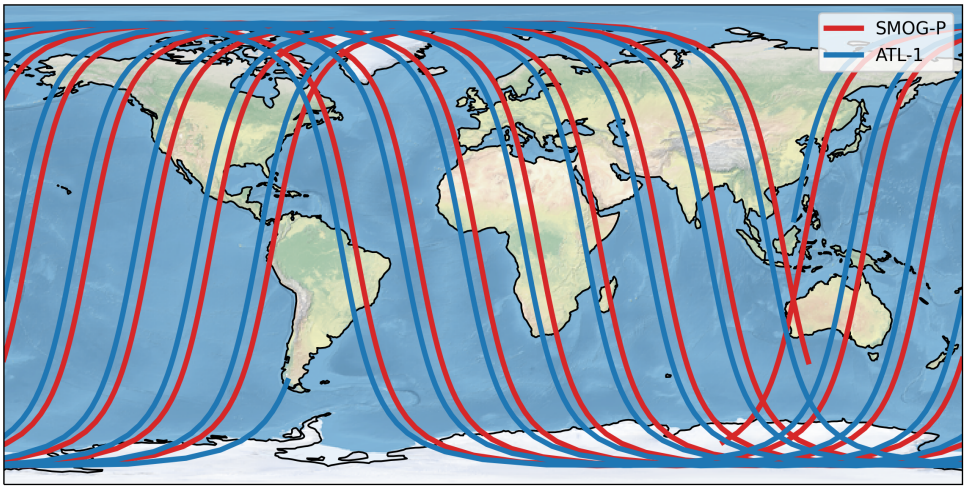
1.3. A missziók lefutása

Mindkét műhold 2019. december 6-án állt Föld körüli pályára. Mindkettő az új-zélandi Rocket Lab saját fejlesztésű Electron rakétáján kapott helyet. A műholdak kezdetben nagyon hasonló, 385 km magasan fekvő napszinkron pályára kerültek. Ez később az eltérő mechanikai tulajdonságaik miatt máshogyan változott, így egyre eltérőbb pályán haladt tovább a két műhold.

A napszinkron pálya (Sun-Synchronous Orbit, SSO) sajátossága, hogy az ilyen pályán haladó műholdak azonos földi helyzet felett ugyanazon időpontban[1] halad el. Másképp fogalmazva: a Nap-Föld irány és a keringési sík által bezárt szög állandó. Mivel ez egy kvázi-poláris pálya, idővel a Föld teljes felszíne feletti áthaladást biztosítja [4], ahogyan azt az 1.2. ábra is mutatja. Ez kulcsfontosságú abból a szempontból, hogy a teljes Földet körülvevő rádiószmog-térkép készülhessen.

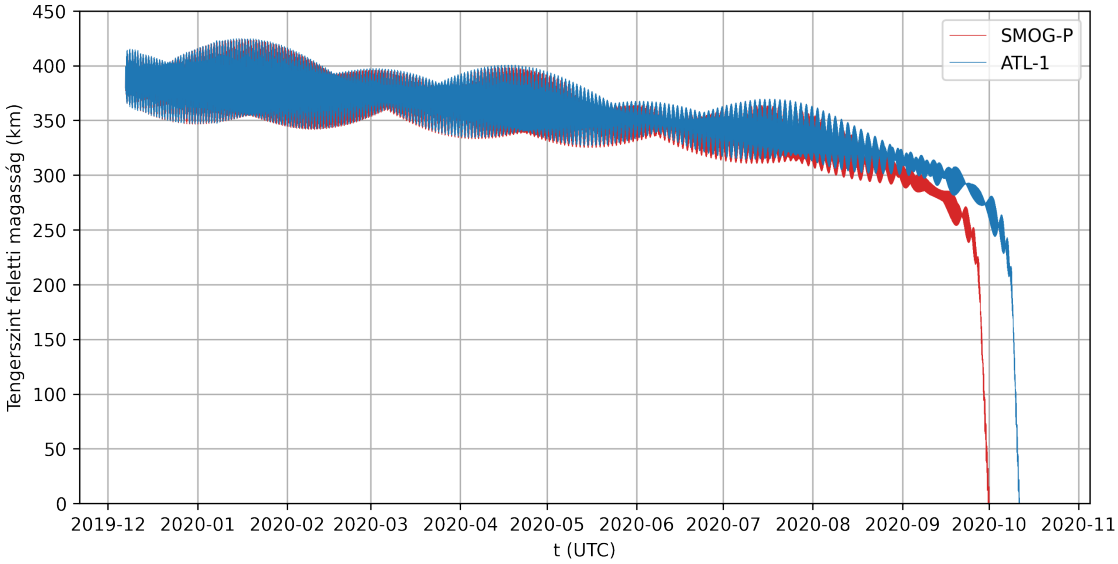
A műholdak pályamagassága az idő előrehaladtával folyamatosan változott, ahogyan azt

[1]Pontosabban: helyi napi középidő szerinti azonos időben.



1.2. ábra. A SMOG-P és az ATL-1 12 órányi pályája 2020. március 9-én. Ekkor már láthatóan elvált a két műhold pályája, bár hasonlóak maradtak.

az 1.3. ábra is mutatja.



1.3. ábra. A két műhold pályamagasságának alakulása az idő függvényében.

Ahogyan az a pályamagasság-adatokból is látszik, a dolgozat írásának idejében már egyik műhold sem keringett a Föld körül: a SMOG-P 2020. szeptember 28-án, az ATL-1 október 18-án semmisült meg a légkörben. Mindkét műhold túlteljesítette tervezett célját: 10 hónapig, a tervezett időnél jelentősen tovább működtek.

1.4. Hardver

A már említett spektrumanalizátoron túl a műholdak fedélzetén számos (hőmérséklet-, mozgás-, mágneses térerősség-, teljes ionizációs dózismérő stb.) szenzor került elhelyezésre, ezekről többek között [6] ad bővebb tájékoztatást. Jelen dolgozat szempontjából a felsoroltak közül az mozgás-szenzor, valamint az antenna [16] érdemel figyelmet.

Mivel a műholdakon nem kapott helyet orientáció-szabályozó rendszer, és az antenna iránykarakterisztikája nem egyenletes, jelentősen függ a mért RSSI-értékek nagysága a műhold aktuális orientációjától. Mivel azonban pl. a SMOG-P összesen 300 mW teljesítményből gazdálkodhatott, valamint a fedélzeti számítógép memóriája [20], továbbá a leküldhető adatok mennyisége is korlátozott volt [5], gondosan meg kellett választani, hogy ez mire fordítódik: ebbe nem fért bele az, hogy a spektrum-értékek mellett folyamatosan, megfelelő felbontással a műhold orientációja is rögzítésre kerüljön. Ez jelentős kihívást jelentett a mérések későbbi kiértékelése során.

1.5. Földi állomások

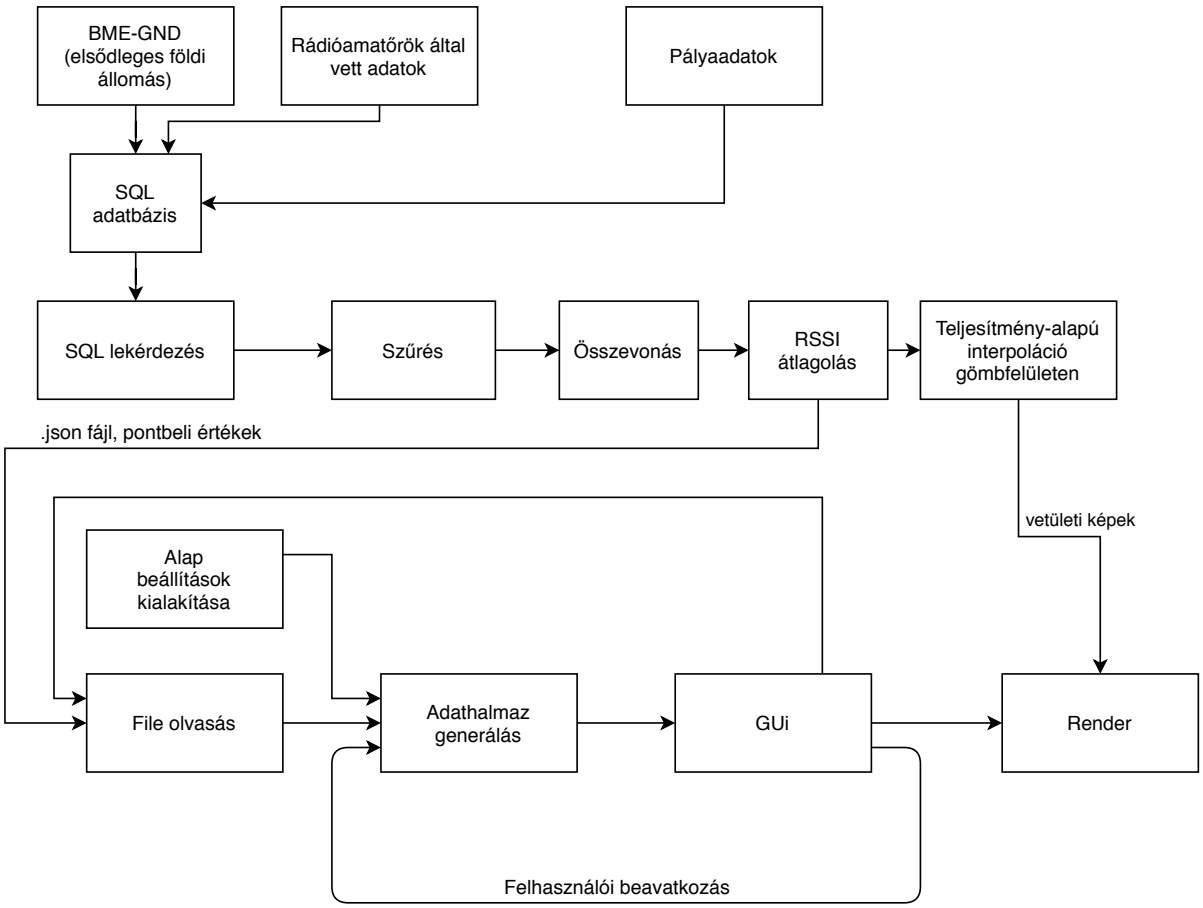
A műholdak a mért adatokat kétféle módon juttatták vissza a Földre. Bizonyos mérési eredményeket (köztük spektrométer-adatokat is) folyamatosan sugároztak, így egy földi vevőállomás a hozzá közel eső műhold-pozíciókban mért értékeket tudta venni. Ezen felül a műhold képes volt előre beprogramozott méréseket megadott időpontban elvégezni, memóriájában eltárolni, majd azt egy megbízható földi állomás felett elfogadva oda lesugározni.

Mivel az első típusú adatokat rádióamatőr-frekvenciákon mindenki szabadon vehette, és a vételhez szükséges program a dekódolt adatokat egy központi szerverre automatikusan továbbította [9], elegendő lelkes rádióamatőr és amatőr műholdakra specializálódott állomás a Föld nagy része felett sugárzott telemetriát tudta venni. Az olyan helyek felett pedig, ahol nem voltak ilyen rádióamatőrök, vagy több mérésre volt szükség, a műholdak operátorai az előre programozott mérésekkel tudták kitölteni a hiányzó helyeket.

Az előre programozott mérések eredményeinek vételének elsődleges földi állomása a BME E épületének tetején található műholdvezérlő állomás, a BME GND. Az itt található, automatikus forgatórendszerrel rendelkező, 437 MHz-en működő 4,5 m-es parabolaantenna folyamatosan követte az áthaladásokat. Ez természetesen jelentősen több összegyűjtött mérési eredményt eredményezett az antenna által észlelhető áthaladások alatt, mint a rádióamatőrök által üzemeltetett állomások, vagy az időzített mérések: ez egy jelentősen inhomogén térbeli eloszlást eredményezett a mérési pontokban (ld. később a 2.1. ábrát).

1.6. A dolgozat felépítése

A dolgozat következő két fejezete ennek megfelelően, és a két szerző munkájának megoszlása szerint van tagolva. Takács Donát készítette az adatfeldolgozáshoz, interpolációhoz szükséges modelleket és azok implementációját, valamint a térképi vetületeket (2. fejezet, kivéve 2.1. alfejezet); Markotics Boldizsár pedig az SQL-adatbázissal való kapcsolatot hozta létre, majd az előbbi eredményekre támaszkodva készítette el a mérési eredményeket interaktívan megjelenítő, háromdimenziós webalkalmazást (2.1. alfejezet, 3. fejezet). Az így létrehozott jelfeldolgozási láncot az 1.4. ábra szemlélteti.



1.4. ábra. A dolgozat tárgyát képező jelfeldolgozási láncot ábrázoló diagram.

2. fejezet

Adatfeldolgozás

Mint minden jelfeldolgozási feladat esetében, a nyers mérési adatokon ebben az esetben is sok műveletet kellett elvégezni, hogy kiértékelhető eredményeket kapjunk. Szükség volt a mérések szűrésére és összevonására, hogy minél nagyobb százalékban hiteles mérési eredményekkel dolgozzunk (2.2-2.3. alfejezet), majd az így kapott, diszkrét pontokban vett értékeket interpoláltuk (2.4. alfejezet), ezután a különféle adathalmazokat különféle céloknak megfelelően ábrázolnunk is kellett, hogy vizuálisan befogadható, valamint a 3D-s ábrázolásban is használható eredményeket kapjunk (2.5. alfejezet).

Az adatfeldolgozás Python nyelven történt, saját fejlesztésű kóddal, melyhez természetesen számos nyílt forráskódú, ingyenesen elérhető Python-könyvtárat [7, 8, 13, 24] használtunk. Ezt a kódot a B függelék tartalmazza. A lényegesebb metódusok és programrészletek nevét a megfelelő részeknél megemlítjük, hogy az érdeklődő olvasó az implementáció részleteibe is betekintést nyerhessen.

Az adatfeldolgozás során számos eljárást paraméterezni kellett. A paramétereket, választott értékeiket és az implementációban használt nevüket a fejezet végén, a 2.6. alfejezetben közöljük.

Az itt ábrázolt grafikonok mindegyikéhez a 2019. 12. 06-tól 2020. 10. 09-ig végzett mérések adatait használtuk fel, a vizsgált frekvenciasáv pedig, ha a szövegben ez külön nincs jelezve, a mérések teljes frekvenciasávja: 119 MHz–960 MHz.

2.1. Adatok letöltése az SQL szerverről

Ahhoz, hogy elkezdjünk dolgozni a központi szerveren összegyűjtött mérési adatokkal, megfelelő formában le kell kérnünk azokat. Műholdanként öt-öt adatbázistáblába rendezve férhetünk hozzá a mérési adatokhoz. Kettő tartalmazza a mérések paramétereit, kettő a mérések eredményeit, egy pedig a műhold pozícióit a mérés időpontjában (lásd 2.4).

Műhold_spa(_measurement)
id
timestamp
start_frequency
step_size
rbw
packet_count
spectrum_lenght
measurement_id
packet_id

2.1. táblázat. A mérések körülményeit tartalmazó tábla oszlopai

Műhold_spa_data(_measurement)
id
spa_id
frequency
rssi

2.2. táblázat. A mérések eredményeit tartalmazó tábla oszlopai

Műhold_position
datetime
latitude
longitude
altitude

2.3. táblázat. A műholdak pozícióját tartalmazó tábla

2.4. táblázat. A mérési adattáblák

Mindkét műhold esetén a `measurement` végződéssel ellátott táblák az előre betervezett mérési helyekről származnak, a `measurement` végződés nélküli táblák pedig a BME-GND földi állomás és rádióamatőrök által vett adatokat tartalmazzák.

Az `spa` és `spa_data`, valamint a `pozíció` megfelelő egymáshoz rendelése után megalkotunk két-két összevont táblát. Ezen táblák kapcsán fontos megjegyezni azt, hogy a földrajzi szélességi és hosszúsági körök értékén kerekítést hajtunk végre az adatbázis méretének későbbi csökkentése érdekében, azonban ez a későbbi számítások pontosságát nem befolyásolja.

Műhold összevont mérés
timestamp
latitude
longitude
altitude
frequency
rssi

2.5. táblázat. Az egyesített adattábla oszlopai

Az így kapott egyesített adattáblák (a pályaadatok és a földi vevőállomások fölött vett táblák) oszlopai megegyeznek, ezért ezeknek unióját képezhetjük. Ezáltal csökkentjük a feldolgozandó fájlok számait. A két műhold táblázatainak összevonásával az adatfeldolgozási lánc későbbi szakaszán foglalkozunk a bővíthetőség fenntartása érdekében, mivel így az adatfeldolgozó program kis módosítással tetszőleges számú műholdból származó adatot képes kezelni. A lekérdezés végeredményét egy .csv fájlba exportáljuk ki. A végső lekérdezést az A. függelék tartalmazza.

2.2. Előszűrés, összevonás

Az SQL adatbázisból lekért adatokat megfelelően szelektálni kell, hogy csak a hasznos adatokkal dolgozzunk tovább. Az SQL lekérés után rendelkezésre álló adatok mérési pontonként: a mérés időpontja másodpercre pontosan UTC időzónában (t, s); az ebből a TLE adatok alapján a szerver által számított mérési pozíció szélességi (φ, deg), hosszúsági (λ, deg) és magassági (h, km) koordinátái; a mérés frekvenciája (f, Hz); valamint a mérés eredményének RSSI értéke (RSSI, dBm). A továbbiakban ezen adatok indexelt változatai konkrét méréseket jelentenek, azaz a teljes mérési adathalmaz

$$\{(t_i, \mathrm{RSSI}_i, \varphi_i, \lambda_i, h_i, f_i)\}, \quad i = 1...N \tag{2.1}$$

alakú, ahol N a mérések száma. Az ebben az alfejezetben tárgyalt számításokat a `select_usable_data` metódus implementálja. Mivel két `.csv` fájlt olvasunk be a két műhold mérési adataiból, azok egyesített értékeire kell lefuttatnunk a metódust, ezt az `import_usable` metódus implementálja.

2.2.1. Térbeli megfontolások

Értékelhető mérések kiválasztása

A műholdak az 1. fejezetben említett tulajdonságai miatt nem lehet tudni, hogy egy mérés során éppen milyen orientációban volt a műhold a Földhöz képest, és merre mutatott az antennája. Telemetria-adatok alapján ismert, hogy a műholdak folyamatosan forgásban voltak, így a nagy számok törvénye alapján feltételezhető, hogy ha elegendően sok alkalommal mérünk a műholddal egy adott pozíció felett, előbb-utóbb lesz olyan mérés, amely során az antenna az optimális (azaz maximális nyereségű) orientációban van a Földhöz képest. Értelemszerűen ez a mérés adja majd a legmagasabb mért értéket is, ha állandónak tekintjük a Földről jövő rádióhullámok teljesítményét, ami jó közelítésnek tekinthető.

Így ha a Földet reprezentáló gömbfelületet régiókra osztjuk, és ezeken a régiókon belül kiválasztjuk bizonyos frekvenciasávokban a legmagasabb mért értéket, elegendő számú mérés esetén nagy valószínűséggel minden ilyen régióban rendelkezünk majd olyan adattal, ami egy közel optimális antenna-orientációnak felel meg. Így ezeket a mért RSSI értékeket valósnak tekinthetjük: azaz ezen RSSI értékek térbeli változását nem a műhold forgásának tudjuk be, hanem a Föld körüli rádiószennyezettség térbeli fluktuációjának.

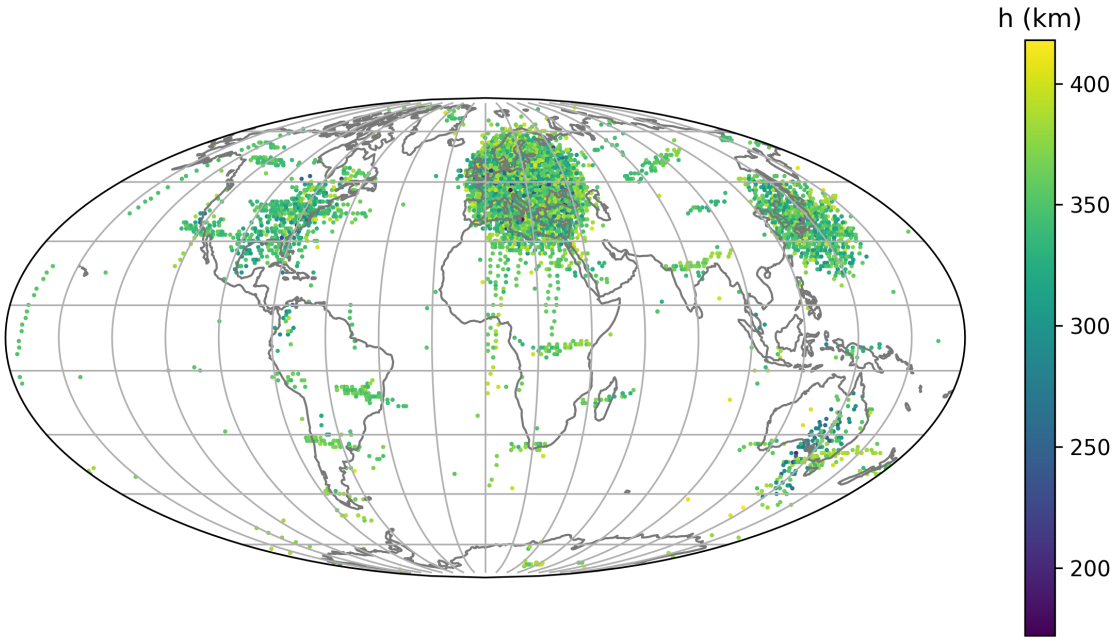
Fontos szempont továbbá, hogy a mérési pontok eloszlását nagyban meghatározta az, hogy a legtöbb mérést aktív földi állomás környékén lehetett elvégezni a műholdak véges memória-kapacitása miatt, ahogy azt a 2.1. ábra is szemlélteti. Éppen ezért fontos szempont az adatok feldolgozása során, hogy ez az egyenlőtlen eloszlás ne torzítsa az eredményt.

A térbeli csoportosítás a szélességi és hosszúsági koordináták kerekített értéke alapján történt: akkor esett ugyanabba a régióba két (φ_i, λ_i) és (φ_j, λ_j) pont (azaz $\varphi_i \equiv \varphi_j$ ill. $\lambda_i \equiv \lambda_j$), ha

$$\mathrm{round}(\varphi_i/\tilde{\varphi}) = \mathrm{round}(\varphi_j/\tilde{\varphi}) \tag{2.2}$$

$$\mathrm{round}(\lambda_i/\tilde{\lambda}) = \mathrm{round}(\lambda_j/\tilde{\lambda}) \tag{2.3}$$

ahol round a hagyományos egészre kerekítési függvény, $\tilde{\varphi}$ és $\tilde{\lambda}$ pedig a régió földrajzi



2.1. ábra. A két műhold által végzett mérések térbeli eloszlása.

szélességi ill. hosszúsági mérete. Lényeges megjegyezni, hogy ez a módszer nem ad egyenlő területű régiókat. Ezt az eljárást a `select_usable_data` metódus tartalmazza.

A magasság figyelembevétele

Mivel a műholdak mérései a pályájuk excentricitása és időbeli változása miatt nem azonos magasságban történtek, szükséges figyelembe venni minden méréshez tartozó magassági értéket is. Egy lehetőség lenne, hogy magasság alapján is sávokra ill. régiókra osztjuk a mérési adatokat, azonban ehhez nem áll rendelkezésre elegendő adat, valamint a mérési időtartam nagy részében nem olyan jelentős a magassági ingadozás. Jó közelítésnek tekinthető tehát, ha a mért értékeket egy közös középfelületre redukáljuk annak feltételezésével, hogy a teljesítmény-források a Föld felszínén találhatóak, és a vett teljesítmény a forrástól távolodva a távolsággal inverz négyzetes törvény szerint változik. Ez nem pontos, hiszen nem csak a sugárra merőleges forrásból érkeznek a műholdak felé a rádióhullámok; azonban ezek a források tekinthetőek dominánsnak a távolság miatt.

A választott középfelület egyszerűen a mérési pontokhoz tartozó magasságértékek számtani közepe, $\bar{h}$. Ezt a 2.2. ábra szemlélteti.

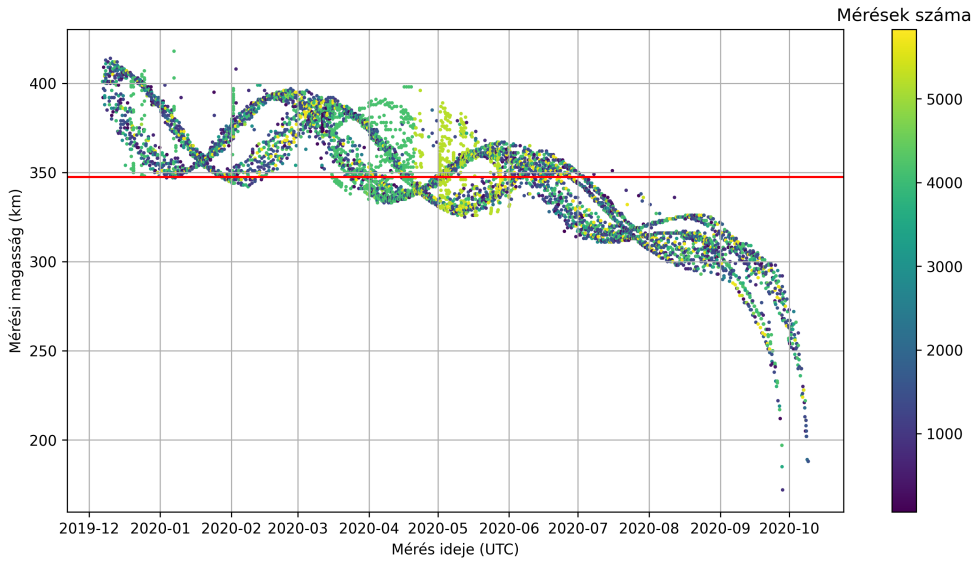
Ezt felhasználva a redukált RSSI értékek:

$$\mathrm{RSSI}'_i = \rho\left(\pi\left(\mathrm{RSSI}_i\right)\frac{h_i^2}{\bar{h}^2}\right) \tag{2.4}$$

Ahol

$$\rho(P) = 10 \cdot \log_{10} P, \quad \pi(\mathrm{RSSI}) = 10^{\mathrm{RSSI}/10} \tag{2.5}$$

az RSSI- és teljesítményértékek közötti átszámítást segítő függvények, melyek megfelelői a kódban a `power_to_rssi` és `rsssi_to_power` metódusok.



2.2. ábra. A mérési magasság és gyakoriság a két műhold adataiban, az idő függvényében. A piros vonal a középfelület magasságát jelöli.

2.2.2. Frekvenciatartománybeli megfontolások

A spektrumanalizátoral végzett mérések frekvenciatartománybeli paraméterei sem ugyanazok az egyes mérések között. Van, hogy a mérés kisebb, van, hogy nagyobb frekvenciatartományra vonatkozik, ez attól is függ, hogy éppen az adatcsomagok hányadrészét sikeresen vennie az adatokat szolgáltató földi állomásnak. Ennek ellenére szeretnénk, ha frekvenciasávonként tudnánk különböző térképeket készíteni, illetve, ha a teljes frekvenciasávra tudnánk egy effektív átlagértéket számítani. Így a térbeli csoportosításhoz hasonlóan egy frekvenciatartománybeli csoportosítást is alkalmaztunk a mérési adatokon.

Két adatpont f_i és f_j frekvenciája ugyanabba a frekvencia-régióba esik (azaz $f_i \equiv f_j$), ha

$$\mathrm{round}(f_i/\tilde{f}) = \mathrm{round}(f_j/\tilde{f}) \tag{2.6}$$

teljesül, ahol $\tilde{f}$ a frekvencia-régió mérete.

2.2.3. Érvényesnek tekintett mérések kiválasztása, összevonás

A fentebb definiált (2.2), (2.3) és (2.6) ekvivalencia-relációk a mérési adatok halmazát ezen három érték alapján egyértelműen ekvivalencia-osztályokba sorolják. Ezen osztályokból a fentebb ismertetett megfontolásoknak megfelelően azokat az méréseket tartjuk csak meg, amelyeknek az ekvivalencia-osztályon belül maximális az RSSI értéke. Ha több ilyen mérés is van, ezeket összevonjuk egyetlen, fiktív méréssé: a fiktív mérés új (φ, λ) koordinátái és f frekvenciája az ekvivalencia-osztálybeli mérések koordinátáinak és frekvenciájának számtani közepei lesznek.

Az eddigiek alapján tehát az előszűrt, összevont méréssel egy

$$\{(\mathrm{RSSI}'_i, \varphi_i, \lambda_i, f_i)\}, \quad i = 1...N' \tag{2.7}$$

adathalmazt kaptunk, ahol N' a szűrt és összevont mérési pontok száma, mely kisebb vagy egyenlő, mint a (2.2), (2.3) és (2.6) által kijelölt ekvivalencia-osztályok száma.

Ezeket a mérési eredményeket a műhold orientációjától már függetlennek, hitelesnek tekintjük a továbbiakban.

2.3. RSSI-átlagérték meghatározása egy frekvenciasávra

A következő lépés, hogy a már érvényesnek tekintett adatokból egy tetszőleges (f_{min}, f_{max}) frekvenciasávra egyetlen jellemző RSSI értéket tudjunk mondani egy (φ_i, λ_i) mérési pontban: ezzel szemléletessé téve azt, hogy milyen erős a rádiófrekvenciás szennyezettség abban a régióban. Más szóval: egy

$$(\mathrm{RSSI}'_i, \varphi_i, \lambda_i, f_{i,j}) \to (\overline{\mathrm{RSSI}}_i, \varphi_i, \lambda_i), \quad f_{min} \leqslant f_{i,j} \leqslant f_{max} \tag{2.8}$$

leképezést keresünk, ahol $\overline{\mathrm{RSSI}}$ a frekvenciasávra vett átlagérték, az i indexelés az azonos (azaz a (2.2) és (2.3) relációk által kijelölt) térbeli régiókba tartozó értékeket jelöli, a j index pedig az ezen régiókon belüli különböző frekvencia-értékek indexe.

Erre a célra a jelfeldolgozásban gyakran használatos négyzetes középértéket használtuk, a teljesítményértékekre értelmezve, majd az eredményt visszakonvertálva RSSI-be:

$$\overline{\mathrm{RSSI}_i} = \rho\left(\sqrt{\frac{\mathrm{trapz}_j(f_{i,j}; \pi\left(\mathrm{RSSI}'_i\right)^2)}{\max_j(f_{i,j}) - \min_j(f_{i,j})}}\right), \quad f_{min} \leqslant f_{i,j} \leqslant f_{max}, \tag{2.9}$$

ahol $\mathrm{trapz}_i(x_i, y_i)$ az (x_i, y_i) értékek numerikus trapéz-integrálja.

Tehát a fenti átlagolással az alábbi jellegű adathalmazokhoz jutottunk:

$$\{(\overline{\mathrm{RSSI}}_i, \varphi_i, \lambda_i)\}_{f_{min}, f_{max}}, \quad i = 1...M, \tag{2.10}$$

ahol M a (2.2) és (2.3) relációk által kijelölt ekvivalencia-osztályok száma. Azaz: minden térbeli régió mérési pontjához számítottunk egy, a frekvenciasávra jellemző, átlag RSSI értéket.

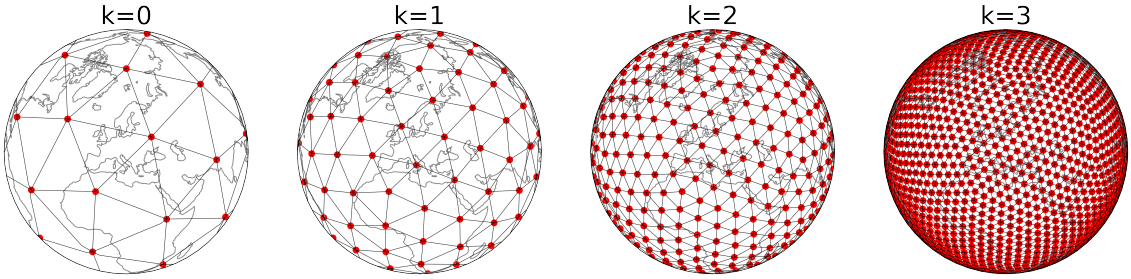
Az ebben az alfejezetben tárgyaltakat az `average_rounded_rssi` metódus implementálja.

2.4. Interpoláció gömbfelületen

Mivel az adatfeldolgozás végső célja az volt, hogy egy folytonos eloszlású térképet hozzunk létre a diszkrét RSSI mérési adatokra, szükség volt valamilyen interpoláció alkalmazására: egy tetszőleges (φ, λ) pontra szeretnénk megbecsülni az ott mérhető RSSI értékét. A középfelületre való redukálással ugyan csökkentettük a ponthalmaz dimenzióját háromról kettőre – így egyszerűbbé téve az interpolációt – azonban a gömbfelület geometriáját nem lehet elhanyagolni. Lényeges jellemzője az interpolálandó felületnek, hogy a gömbi geometria miatt nem lehet egyszerűen a koordináták mentén interpolálni: egyrészt a gömbi koordináta-rendszerben a $\varphi = \pm 90^\circ$ szélesség esetén tetszőleges λ hosszúsági koordinátájú pontok ugyanannak a térbeli pontnak felelnek meg; másrészt a gömbfelületnek topológiai értelemben nincs határa, azaz a $\lambda \equiv \lambda + 2\pi$; harmadrészt a gömb felületén lévő pontok közötti (légvonalbeli és felszíni) távolság is bonyolult függvénye a koordinátáknak. Ezek

a szempontok igen jelentősek a gömbfelületen számított, RSSI-szintvonalak esetében is: szeretnénk, ha azok a dátumválasztónál is folytonosak lennének.

Ezen probléma megoldásához végül a `stripy` python könyvtárat [15] használtuk, mely a STRIPACK [17] és SSRFPACK [18] Fortran könyvtárakkal interfészel. Ezek a könyvtárak igen hatékonyan kezelnek nagy számú, gömbfelületen elhelyezkedő pontot; a pontokat egy háló részeként kezelik, és különféle interpolációs módszereket is tartalmaznak beépítve. A 2.3. ábrán látható az általunk használt ikozahedrális háló előállítása: egy ritka, durva hálóból fokozatos globális sűrítéssel lehet tetszőlegesen finom hálót előállítani, melyet a k paraméter jellemez. A pontok pozíciója egy K-d fa struktúrában van tárolva, így lehetővé téve a gyors térbeli keresést és interpolációt. Az ikozahedrális háló csúcspontjait és lapközéppontjait is használtuk a megfelelően sűrű háló érdekében.



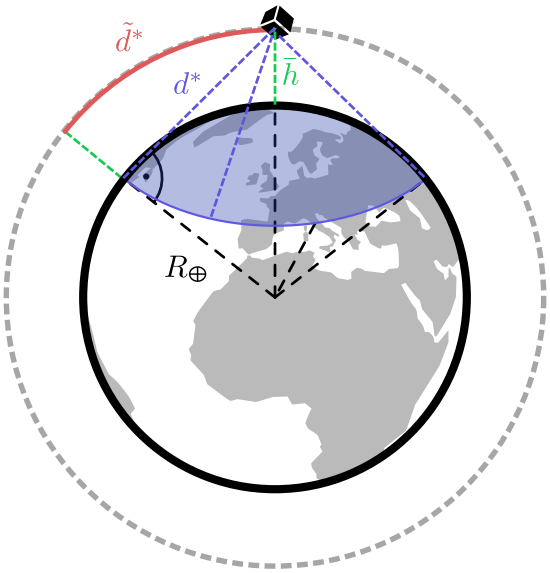
2.3. ábra. A használt ikozahedrális háló finomításának első négy iterációja.

Magához az interpolációs függvényhez saját eljárást kellett definiálnunk, nem voltak elegendőek a beépített lehetőségek. A szakirodalomban számos módszer fellelhető RSSI mérések interpolációjához (például spline-illesztés, távolság-alapú súlyozott átlagolás, krigelés [3, 12]). Mi távolsággal inverzen súlyozott interpolációt (*inverse distance weighting*, IDW) alkalmaztunk az RSSI értékekből számított teljesítmény-értékekre. Egyrészt azért, mert [3] esettanulmányában ez adta a legpontosabb eredményt egy hasonló feladatban; másrészt az IDW interpoláció egyik fő gyakorlati nehézsége általánosságban a d távolság kitevőjének meghatározása a w súlyokhoz, azonban ebben az esetben űrbéli terjedésről van szó, tehát meglehetősen pontosnak tekinthető az ideális, $w \propto d^{-2}$ arányosság; harmadrészt az IDW interpoláció a legmegengedőbb a pontatlan mérési adatokkal szemben, melyeknek a fentebb tárgyaltak nyomán a gondos előszűrés ellenére is megvan az esélye.

Figyelembe kell venni az interpoláció során, hogy egy adott környezetben csak azokat a pontokat érdemes interpolálni, melyek olyan terület felett helyezkednek el, melyek a műhold a horizontjánál közelebb vannak. Másképp fogalmazva ezek azok a mérési pontok, ahonnan még látszik a Föld azon pontja, mely felett az interpolációt végezzük.

Ezek vetülete egy bizonyos d^* távolságon belül van: ez a műhold látóköre, melynek mérete a $\bar{h}$ átlagos pályamagasságból és a Föld közepes sugarából számítható. Ennek geometriáját a 2.4. ábra szemlélteti. Az interpolációs könyvtár azonban az *interpolációs* gömbfelületen mért geodetikus távolság alapján adja meg a legközelebbi pontokat, így a d^* látótávolság helyett a $\tilde{d}^*$, középfelületen mért ívhosszra van szükségünk. Ezek kiszámítása a 2.4 alapján:

$$d^* = \sqrt{\left(R_\oplus + \bar{h}\right)^2 - R_\oplus^2}, \quad \tilde{d}^* = \left(R_\oplus + \bar{h}\right) \arcsin\left(\frac{d^*}{R_\oplus + \bar{h}}\right) \tag{2.11}$$



2.4. ábra. A látókör paramétereinek kiszámítása gömb alakú Föld közelítéssel, $\bar{h}$ közepes magasságú kör alakú pálya esetén. (Az ábra nem méretarányos.)

ahol $R_\oplus$ a Föld közepes sugara.

Az interpolációs függvény:

$$\mathrm{RSSI}(P_0) = \rho\left(\frac{\sum_i \pi\left(\overline{\mathrm{RSSI}_i}\right) w\left(P_0, P_i\right)}{\sum_i w\left(P_0, P_i\right)}\right), \text{ ha } \tilde{d}\left(P_0, P_i\right) < \tilde{d}^* \tag{2.12}$$

valamint az ehhez használt súlyok:

$$w\left(P_0, P_i\right) = \frac{1}{d\left(P_0, P_i\right)^2} \tag{2.13}$$

ahol $d(P_i, P_j)$ két pont euklideszi, míg $\tilde{d}(P_i, P_j)$ két pont gömbfelületi távolságát jelöli. A P_0 pont tetszőleges lehet (ebben az esetben az ikozahedrális háló egy csúcsa vagy lapközéppontja), míg P_i egy mérési pont. Az első távolság-mértéket a saját `xyz_dist_latlon` metódus implementálja, a másodikat a `stripy` könyvtár K-d fa alapján közeli pontokat kiválasztó metódusa tartalmazza.

Ezen interpoláció segítségével a középfelület tetszőleges pontjára ki tudjuk számítani az interpolációval becsült RSSI értéket, figyelembe véve a gömbfelület tulajdonságait. Az ezen alfejezetben tárgyaltakat az `IDW_interpolate_rssi` metódus implementálja.

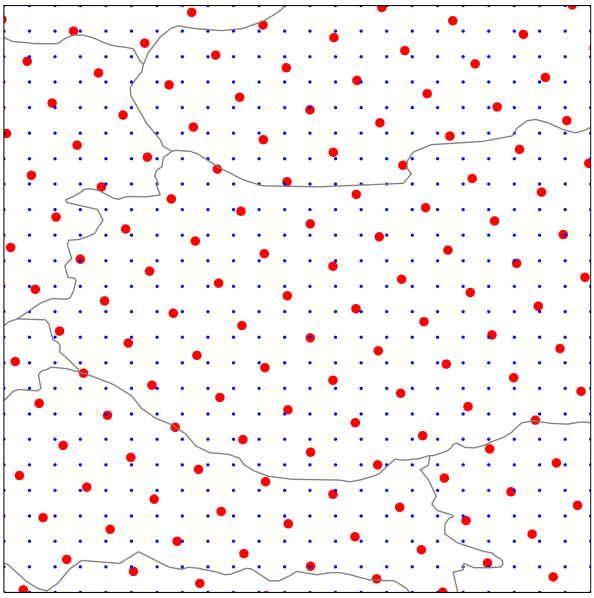
2.5. Ábrázolás

Az interpolált adatokból 2D-s ábrákat szeretnénk készíteni, hiszen egyrészt szeretnénk hagyományos térképi vetületen ábrázolni a mérési eredményeket, valamint a 3D-s térkép felületi textúrái is 2D-s képek formájában adhatóak meg. Számos különböző ábrázolást alkalmaztunk, melyek különböző fokig veszik figyelembe a diszkrét ill. interpolált mérési adatokat.

A térképészet klasszikus problémája, hogy hogyan ábrázoljunk egy gömbfelületet síkba terítve. Elemi differenciálgeometriai ismeretek birtokában könnyen belátható, hogy a feladat lehetetlen: a sík és a gömb Gauss-görbülete különböző, így a nevezetes *Theorema egregium* szerint nem létezik torzításmentes térképi vetület [10]. Így a projekció kiválasztása mindig attól függ, hogy milyen tulajdonságot szeretnénk kevésbé torzítani: így mi is többféle vetületi módszert alkalmaztunk az adott célnak megfelelően. A térképi vetületek alapjai iránt érdeklődő olvasót a műegyetemi geodéta-képzés klasszikus szövegéhez, a Krauter-féle könyvhöz [11] irányítjuk. A vetületi ábrázolás implementálásában nagy segítségünkre volt a `cartopy` [14] könyvtár.

2.5.1. Áttérés ikozahedrális hálóról síkfelületre

A `stripy` használatával kapott interpolált értékek a fentebb tárgyalt ikozahedrális háló csúcspontjaiban és lapközéppontjaiban helyezkednek el. Ezt más alakra kell hozni, ha vetületi térképeket szeretnénk készíteni. Ha elegendően sűrű a háló, egy egyszerű, egyenközű szélesség-hosszúság rácsban lineáris interpolációval is ki lehet számolni a köztes értékeket: elegendően kis környezetben érvényes a síkkal való közelítés a térbeli interpolációnál, ellentétben a globális esettel. Ezen interpoláció implementációját a `plate_carree_interpolation` metódus tartalmazza. Az így kapott szélesség-hosszúsági koordináta-rácsra illesztett RSSI értékeket használjuk fel a további ábrázolások során. A rácsot a $\tilde{k}$ fokonkénti osztások számával paramétereztük. Egy ikozahedrális és egy rektanguláris háló összehasonlítását mutatja a 2.5. ábra.

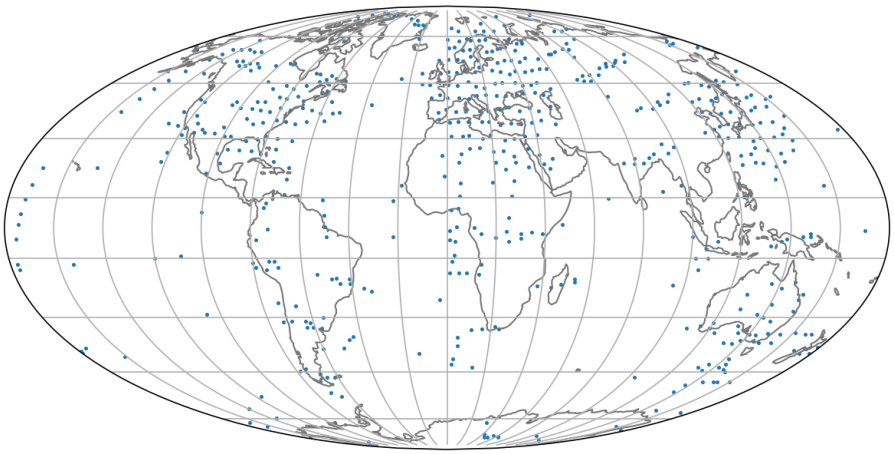


2.5. ábra. Egy $k = 7$ paraméterű ikozahedrális (piros) és egy $\tilde{k} = 6$ paraméterű (kék) rács összehasonlítása Magyarország felett (Mollweide-vetület).

2.5.2. A mérési pontok térbeli eloszlása

Első körben szeretnénk megvizsgálni, hogy a (2.7) szerinti adathalmaznak milyen az eloszlása a Föld felszínén. Ehhez a Mollweide-féle vetítést használtuk, melynek lényeges tulajdonsága, hogy területtartó; így a mérési pontok sűrűségének ábrázolására hasznos. Egy ilyen eloszlást mutat az alábbi a 2.6. ábra, és a fentebbi 2.1. ábra. Látható, hogy

az összevonással sikerült javítani a pontok térbeli eloszlásának sűrűségén, azonban így is maradtak területek, ahol relatíve kevés mérési eredménnyel rendelkezünk.

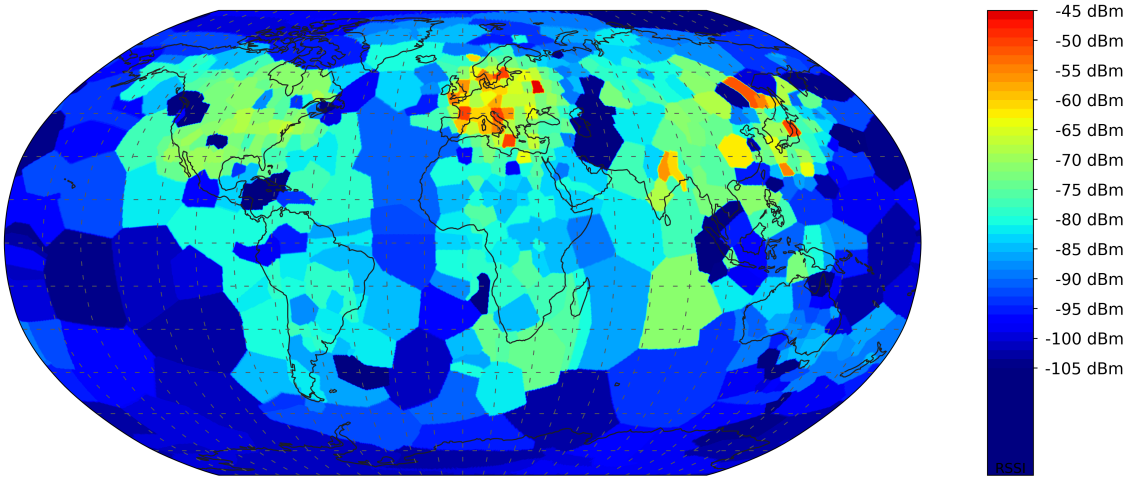


2.6. ábra. Az előszűréssel és összevonással kapott mérési pontok eloszlása, Mollweide vetületen ábrázolva.

Ezt a térképet a 3D-s ábrázoláshoz nem fogjuk használni, hiszen ott ezeket a pontokat térben jelenítjük meg.

2.5.3. RSSI-értékek Voronoj-diagramja

Ha nem szeretnénk interpolációt használni, hanem pusztán a mérési adatok által sugallt RSSI-eloszlást szeretnénk ábrázolni, hasznos a kiátlagolt RSSI-értékek térbeli eloszlása lapján egy Voronoj-diagramot készíteni. Ebben az esetben a gömbfelület minden pontjában a hozzá legközelebb eső mérés eredményét jelenítjük meg, egy a (2.10) jellegű, frekvenciákra átlagolt halmazból. Ez egyszerűen megvalósítható a `stripy` beépített `measurement_triangulation.interpolate` metódusával, ennek részleteit a `voronoi_rssi` metódus tartalmazza. A kapott interpolációt a 2.7 szemlélteti Robinson-féle vetítéssel.



2.7. ábra. Az interpolálatlan mérési eredményekkel kapott Voronoi-cellák ábrázolása Robinson-vetületen.

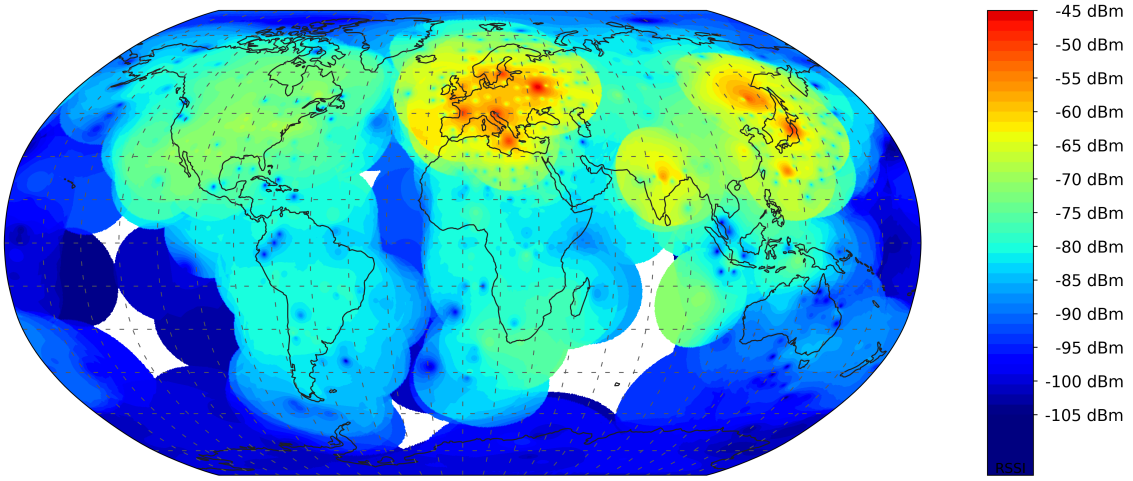
Az ábrát megnézve látható, hogy bizonyos területek felett nem készült elegendő mérés. Egyrészt pl. az Egyesült Államok felett néhány száz km-es alacsony RSSI-jű foltok vannak,

melyeket jelentősen magasabb RSSI-jű tartományok vesznek körbe: ez a pár száz kilométer nem akkora távolság, hogy teljesen megszűnjenek a környező területen érzékelhető rádiójelek: valószínűleg az ezen a területen készült mérések során a műhold mindig rossz orientációban volt. Másrészt az Indiai-óceán felett olyan nagy cellák találhatóak, melyeknek középpontja messzebb van a határoktól, mint 2100 km (azaz a műhold látókörének sugara), így ezeken a területeken a látókör-alapú IDW interpoláció nem fog eredményt adni. (Ennek eredményét a 2.5.4. részben mutatjuk majd be.)

Ezen, valamint több alábbi interpoláció eredményeit is elérhetővé tettük az interaktív felületen, melyhez ekvirektanguláris, *Plate Carree* vetítést kellett alkalmaznunk. Ez egy 2:1 arányú képet generál, melyet a THREE.js megjelenítő textúraként értelmezve gömbfelületen meg tud jeleníteni. Érdemes megjegyezni, hogy a 2.5.1. részben tárgyalt egyenközű koordináta-rács derékszögű-koordináta rendszerben való ábrázolása éppen a Plate Carree vetületet adja: pixelenként egyenközűen változik az azokhoz tartozó szélességi ill. hosszúsági koordináták értéke.

2.5.4. Folytonos eloszlású RSSI

A 2.4. alfejezetben tárgyaltak alapján kapott eredményeket a 2.8. ábra szemlélteti.



2.8. ábra. Az IDW interpolációval kapott eredmények ábrázolása Robinson-vetületen.

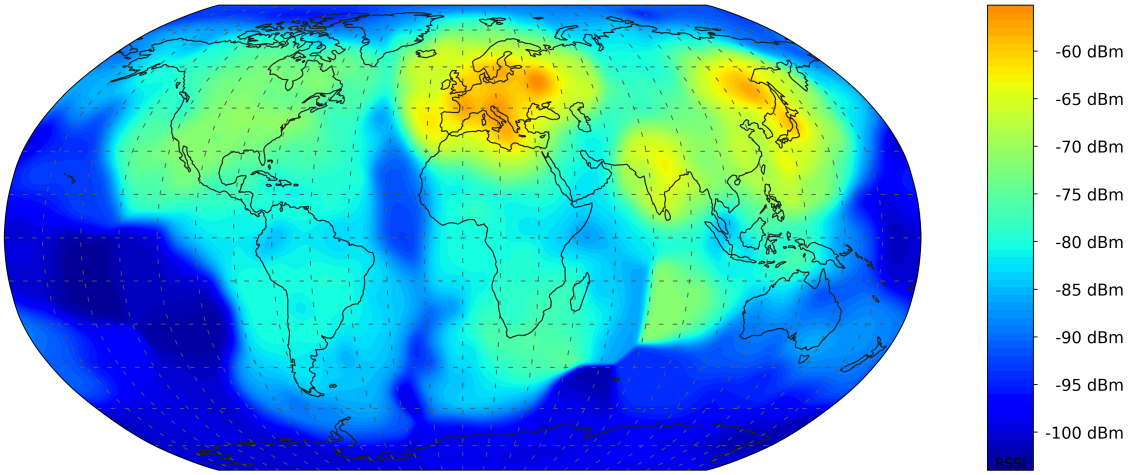
A magas RSSI-jű területekben számos helyen megfigyelhető egy-egy más színű, alacsonyabb RSSI értékű pont. Ez az IDW interpoláció sajátossága: mivel a 2.6. ábrán feltüntetett mérési pontokban kapott értékeket hitelesnek tekintettük, ezen pontok közvetlen közelében az ottani értékeket nem fogják felülírni a környezőek.

Néhány helyen kifejezetten látványos a mérési pontok körüli látókörök széle: ez bizonyára nem felel meg a fizikai valóságnak, hiszen más helyeken viszont kontinenseken keresztül is folytonos a változás. Sőt, a korábban említett módon az Indiai- és a Csendes-óceán felett megfigyelhetőek olyan területek, melyeknek nem adott értéket az IDW interpoláció: egy mérési pont látókörébe sem esnek belel.

Felmerül a kérdés, hogy hogyan kezeljük ezeket az artefaktumokat. Bizonyos értelemben minden ezutáni adatfeldolgozás még távolabb visz minket a valóságtól, ugyanakkor néhány hibáját is kijavítja az IDW interpoláció eredményeinek. Mivel az is szempontunk volt, hogy egy laikusok számára is könnyen értelmezhető térképet készítsünk – ami e dolgozat

írásakor még mindig a legjobb létező közelítése a Föld körüli rádiószmog-eloszlásnak – egy Gauss-kernellel való simítás mellett döntöttünk. Ez megoldja az éles határok, és a láthatóan helytelen RSSI értékű foltok problémáját is, valamint kitölti a mérési eredmény nélküli területeket is, ha azok nem túl nagyok. A Gauss-kernel paramétere a $\tilde{\sigma}$ fokban mért szórás, melyből a $\sigma_x = k\tilde{\sigma}$ összefüggéssel kaphatjuk meg a pixelekben mért szórást, melyből a Gauss-kernel értékét számolhatjuk. Ennek részleteit a `plate_carree_interpolation` eljárás tartalmazza. Érdekesség, hogy az ehhez szükséges konvolúciót az adathalmaz nagy mérete miatt frekvenciatartományban kellett elvégeznünk, hogy reális mennyiségű RAM-ba beleférjünk a futtatáskor. Ehhez az Astropy könyvtárat [1, 2] használtuk.

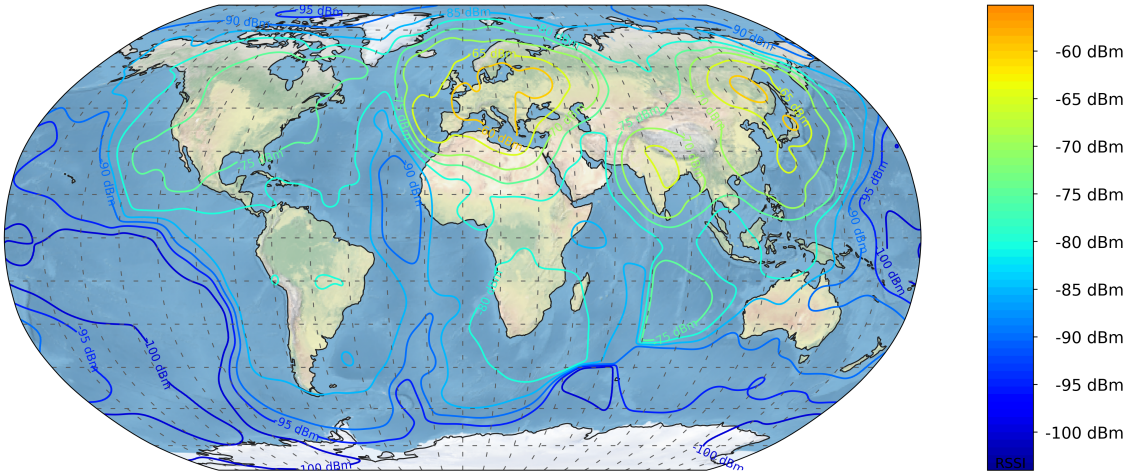
A Gauss-simítással kapott grafikont a 2.9. ábra mutatja. A simítás megoldotta a felsorolt problémákat, azonban behozott egy újat: az RSSI értékek tartománya eltolódott a méréshez képest, így ezeket a számértékeket óvatosabban kell kezelni: inkább az általános tendencia értékelésére érdemes használni ezt a térképet.



2.9. ábra. A Gauss-simítással kapott térkép, Robinson-vetületen.

2.5.5. Szintvonalas ábrázolású RSSI

A simított térkép alapján már könnyen készíthetünk egy szintvonalas ábrázolású térképet is, ahogyan azt a 2.10. ábra is mutatja.



2.10. ábra. A Gauss-simítással kapott térkép azonos RSSI-értékekhez tartozó szintvonalai.

2.6. Az adatfeldolgozás paraméterei

Ebben a fejezetben számos adatfeldolgozással kapcsolatos paramétert bevezettünk. Ezeknek egy része az adathalmaz sajátossága, többi értéke pedig a mi választásunktól függött. Ez utóbbiakat úgy választottuk meg, hogy egyensúlyt találjunk az eredmények pontossága, értelmezhetősége és a számítások megvalósíthatósága között. A paraméterek jelöléseit, értelmezését, a programkódban használt nevüket és a használt vagy kapott értéküket a 2.6. táblázat tartalmazza.

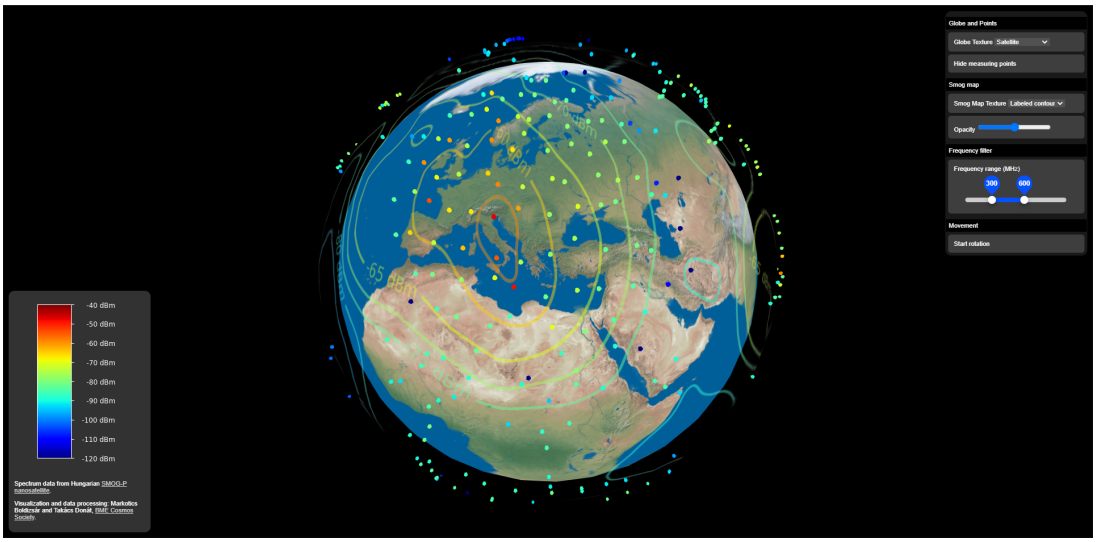
Jelölés	Értelmezés	Változónév	Érték
$\tilde{\varphi}$	Összevonási régió mérete szélességi fokban	`spatial_res_deg_lat`	5
$\tilde{\lambda}$	Összevonási régió mérete hosszúsági fokban	`spatial_res_deg_lon`	5
$\tilde{f}$	Frekvencia-régió mérete (MHz)	`frequency_res_mhz`	50
k	Ikozahedrális gömbháló sűrítési foka	`icosa_mesh_ref_level`	7
$\tilde{k}$	Plate Carree vetületi rács osztása (1/fok)	`points_per_deg`	6
$\tilde{\sigma}$	Gauss-kernel szórása (1/fok)	`gauss_sigma_deg`	3
N	Nyers mérési adatsorok száma	(nincs)	19 115 754
N'	Szűrt és összevont mérési pontok száma	(nincs)	5 904
$\bar{h}$	Középfelület tengerszint feletti magassága (km)	`mean_altitude`	347

2.6. táblázat. Az adatfeldolgozás paraméterei.

3. fejezet

Webalkalmazás

A webalkalmazásunk célja, hogy a szűrt, valamint interpolált adatokat bármilyen felhasználó számára könnyen elérhetővé tegyük. A dolgozat írásának idejében az elkészült alkalmazás megtalálható a https://gnd.bme.hu/mb/site oldalon.

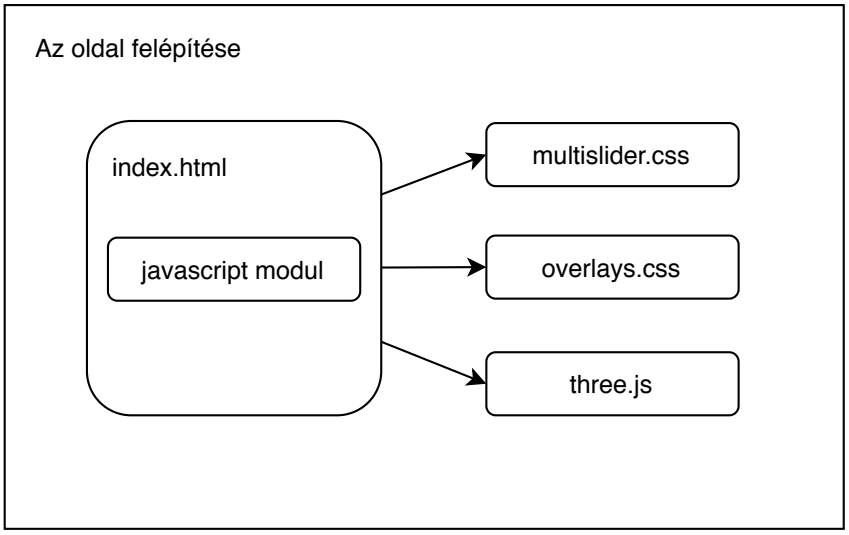


3.1. ábra. Az elkészült weboldal

A rengeteg adat könnyebb befogadása érdekében egy háromdimenziós és interaktív megjelenítő felület mellett döntöttünk, melyet a 3.1 ábra szemléltet. Itt megfigyelhető az interaktív kezelőfelület, a Föld, a különböző (szűrt, összevont) mérőspontok helyei pontos RSSI értékekkel, valamint a bolygó körüli űrbéli rádiófrekvenciás szennyezettséget (továbbiakban: elektorszmog) reprezentáló interpolált értékek szintvonalas ábrázolása.

3.1. Az alkalmazás felépítése

Az alkalmazás alapját egy HTML (*Hypertext Markup Language*) fájl képezi. Ez tartalmazza a megjelenő főbb elemeket, az oldal alapvető funkcionalitását tartalmazó javascript modult, hivatkozásokat a stíluslapokra és a háromdimenziós környezetet támogató THREE.js könyvtárra.



3.2. ábra. Az oldal vázlatos felépítése

3.1.1. Stíluslapok

- **overlay.css**
  Az oldalon megjelenő különböző kiegészítő elemek formázásáért felel, például a jelmagyarázat helyes megjelenítéséért, a menü dizájnjáért és az oldalon megjelenő egységes betűtípusért.

- **multislider.css**
  Ez a fájl felelős a később bemutatott menü *Frekvenciaszűrés* (lásd 3.4) funkciójának megvalósításáért. Érdekessége, hogy mivel a HTML nem rendelkezik tartományszűrő beviteli csúszkával, így két hagyományos csúszka egymásra helyezésével kellett létrehozni ezt az elemet. A megoldás bonyolultsága tette indokolttá a külön stíluslapba való helyezést.

3.1.2. THREE.js

A THREE.js egy WebGL alapú JavaScript könyvtár [19]. A WebGL a JavaScript nyelvet egészíti ki háromdimenziós render lehetőségével, mindezt úgy, hogy a megjelenítéshez a számítógép videokártyáját is használja, ezzel szétosztva a webalkalmazások erőforrásigényét. A THREE.js tovább egyszerűsíti a háromdimenziós objektumokkal való munkát. A térben könnyedén elhelyezhetünk objektumokat, általunk választott látószögű kamerával pedig megjeleníthetjük azokat.

Az oldal létrehozásához használt, az oldal alapvető működésének megértéséhez szükséges főbb objektumok a könyvtáron belül a következők:

- **THREE.Layers();**
  Különböző megjelenítési rétegeket lehet a segítségével előállítani, ez a funkció főleg az interakció növelésére használható. A továbbiakban *rétegként* hivatkozunk rá.
  Előnyei: Az objektumot el tudjuk rejteni, ha nem akarjuk megjeleníteni. Nem kell azt törölni, majd újra betölteni, ha ismét szükség van rá. Ez nagyságrendekkel gyorsítja a gyakran használt objektumok ki- és bekapcsolását, ezzel növelve futás gördülékenységét.
  Hátrányai: Mivel a nem használt objektumot nem töröljük, csak elrejtjük, így nem szabadítunk fel erőforrást: bár videokártyát nem terhelünk, azonban a memóriát fogyasztjuk. Ha túl sok objektum létezik, az alacsony teljesítményű eszközök felhasználói élménye az elvártnál rosszabb lehet.

- **THREE.Group();**
  Csoportosítási lehetőség, sok hasonló típusú objektum kezelhető ezzel metódussal, aránylag nagy számosságú adathalmazok esetén is. Főként az objektumok létrehozásánál és eltávolításánál van nagy jelentősége a csoport használatának. Egyszerűen és átlátható módon lehet ez imént említett műveletet végrehajtani. Továbbiakban *csoportként* hivatkozunk rá.

- **THREE.Scene();**
  Színtér, tartalmazza a megjelenítendő objektumokat, fényeket és az ezeket rögzítő kamerát.

- **THREE.PerspectiveCamera(fov, aspect, near, far);**
  Perspektív vetítést használó kamera. Ezt a vetítési módot úgy tervezték, hogy utánozza az emberi szem látását. Ez az ortografikus mellett a leggyakoribb vetítési mód, melyet 3D jelenetek rendereléséhez használnak. A konstruktorában megadható paraméterek:

  - `fov`
    A kamera látószögének a meghatározása. Értéke jelen esetben 75 személyes preferencia alapján.

  - `aspect` – szélesség/magasság formátumban
    A reszponzivitás növelése érdekében a szélesség illetve magasság mindig egy arányszám. Az oldal lefutásakor az aktuális kijelző képarányának felel meg.

  - `near, far`
    A kamerától mért távolság értékek. A „near” a közel-síkját még, a „far” a távolsíkját határozza meg. A két érték között lévő objektumok fognak látszódni a megjelenített képen.

- **THREE.WebGLRenderer(canvas: globecanvas, alpha: true , antialias: true);**
  Az elkészített objektumokat jeleníti meg a WebGL segítségével. A konstruktorban meg kell adni a megfelelő HTML objektumot amibe megjelenik a háromdimenziós színtér. A többi paraméter esetén pedig az alapbeállításokat használjuk, kettő kivétellel, az elsimítást és az átlátszóságot engedélyezzük. Az élsimítás bár erőforrás igényesebbé teszi az alkalmazást, azonban nagy felbontású monitorokon sem jelennek meg éles határpixelek. Az átlátszóság pedig későbbiekben nélkülözhetetlen az interaktív oldal kezeléséhez, hisz egyes objektumok átlátszóságát a felhasználó

állíthatja.

- **OrbitControls(camera, renderer.domElement);**
  A kamera bolygó körüli keringetéséhez szükséges. Ha a kamerát mozgatjuk a Föld körül, azt a hatást kelti, mintha a kamera statikus lenne és a bolygót forgatnánk. Ez azért fontos, mert így tudunk rögzített koordináta-rendszert használni, ami leegyszerűsíti az objektumok helyzetének a számontartását. A natív THREE.js könyvtár ezt nem tartalmazza, így ezt külön hozzá kellett adni. A konstruktorában megadható paraméterek:

  - camera
    Az irányítani kívánt kamera.

  - domElement
    Az esemény figyelők által használt HTML elem. Ugyanaz, mint a WebGLRendererben beállított (lásd: 3.1.2).

A fent említett objektumokat, metódusokat az alkalmazás inicializációjakor elég futtatni, mivel a továbbiakban nem változnak.

Azonban az oldal interaktivitása miatt előfordul, hogy objektumok vagy objektum-csoport valamely tulajdonságát kívánjuk megváltoztatni. Ezen műveleteket a felhasználói felületről lehet végrehajtani, amit a 3.4. részben tárgyalunk. Ezek az objektumok és objektum-csoport a Föld, az ezt körülvevő interpolált elektroszmog térkép gömbfelülete és a mérési pontokat reprezentáló gömb-halmaz. Ezekről a 3.3 részben írunk.

3.2. Adatfájl beolvasása

Az előkészített adatbázist egy JSON fájl tartalmazza. Az alkalmazás ezt minden oldalbetöltéskor beolvassa. Ezután pedig egy JSON tömbbé alakítja, amit továbbiakban az oldal bármikor elér, ezen képes megfelelő lekérdezéseket végrehajtani.

A tömb egy eleme tartalmazza a mérési pont φ, valamint λ földrajzi koordinátáját (lásd 2.2), az ehhez a méréshez tartózó RSSI értéket egy hozzárendelt színnel,végül egy frekvencia tartomány alsó és felső határát. Lásd 3.1 táblázat.

Jelölés	**JSON Attribútum**	**Lehetséges érték**
$\boldsymbol{\varphi}$	X	-123.0
$\boldsymbol{\lambda}$	Y	44.0
RSSI	Rssi	-97.42
$\boldsymbol{f_{min}}$	freq_bin_lower	0.0
$\boldsymbol{f_{max}}$	freq_bin_upper	200.0
Szín	color	0xFEE2D5

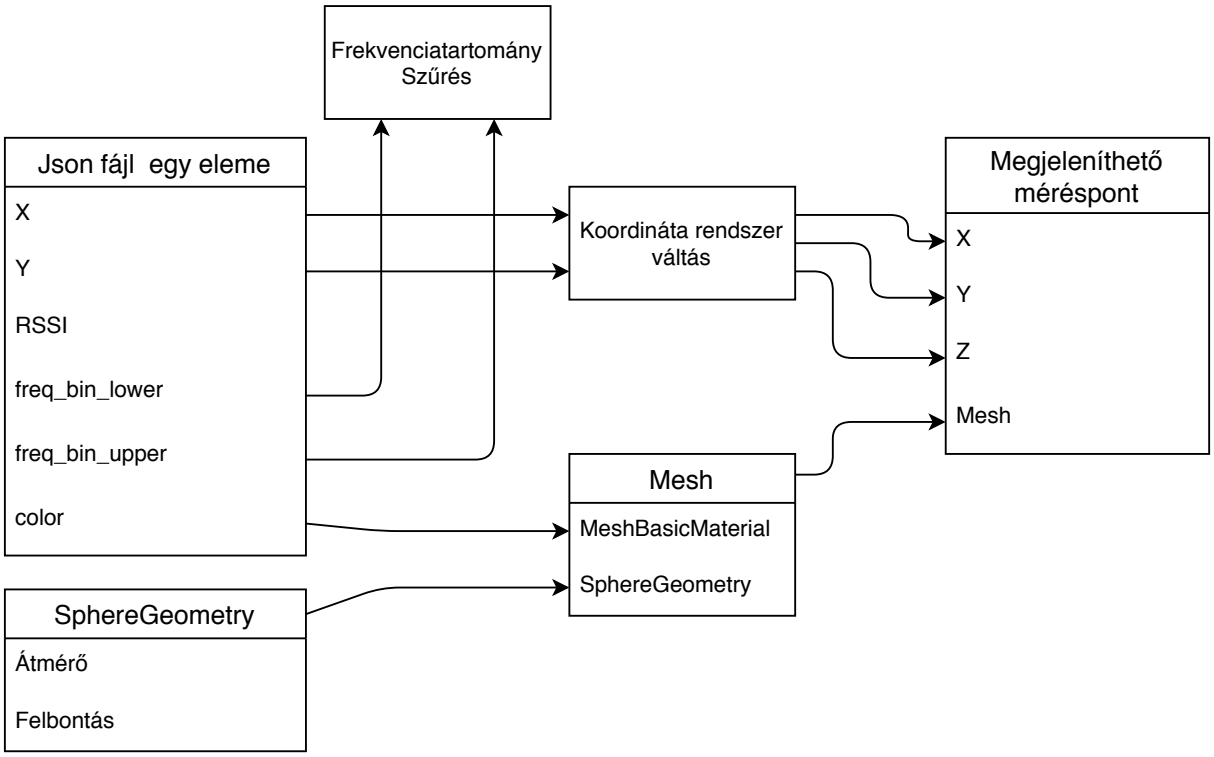
3.1. táblázat. A beolvasott JSON fájl kapcsolata a dolgozat jelöléseivel

3.3. Adathalmaz generálása

Az inicializáció közben, még felhasználói interakció előtt meg kell jeleníteni a Földet, az azt körülvevő elektroszmog térképet és a méréspontok helyét reprezentáló gömb halmazt. Ehhez beállítunk alapértelmezettnek egy-egy textúrát a Föld és elektroszmog gömböknek, továbbá egy 300 MHz és 600 MHz közötti frekvenciatartomány szűrést. Átlagosan világszerte ezen a tartományon működnek a digitális műsorszóró adók. A világűrbe jutó elektroszmog szennyezettség nagy százaléka származik innét, és a műholdak fő mérési tartománya is ez volt, tehát ez a tartomány szemléletes adatokat tartalmaz.

3.3.1. Méréspontok

A 3.2 pontban leírt módon beolvasott .JSON fájlon az imént említett alapértelmezett frekvenciaszűrést lefuttatva létrehozunk egy szűrt JSON adattömböt. Ezt pedig megkapja egy függvény, ami minden sorból létrehoz egy "méréspont" háromdimenziós gömb objektumot. Ezt pontban mért RSSI értéknek megfelelő színnel látja el. Végül pedig pedig annak helyére állítja azt. Az egész folyamat a 3.3 ábrán látható. Miután helyére kerül a pont, belekerül a méréspontokat tartalmazó csoportba. Az elkészült csoport felkerül a megjelenítésért felelős rétegre is. A felhasználói felületen egyszerre ki- és bekapcsolható lesz az egész csoport ezen tulajdonsága miatt. A frekvencia szűrés módosítása esetén a csoport összes eleme törlésre kerül, az új frekvenciatartományi beállítással pedig újra lefut a leírt függvény.



3.3. ábra. A szűrt adatbázis sorából egy méréspontig

3.3.2. Koordináta rendszer váltás

Azonban beérkező méréspontok helyi földrajzi koordináta-rendszernek megfelelően vannak tárolva, nekünk viszont háromdimenziós koordinátákra van szükségünk, hogy megfelelően tudjuk megjeleníteni a méréspontokat. Ehhez előbb gömbi koordináta-rendszerbe kell átalakítani őket a következőző képlet alapján:

$$\phi = (90 - \varphi) \cdot \left(\frac{\pi}{180}\right) + \frac{\pi}{2}, \tag{3.1}$$

$$\theta = (\lambda + 180) \cdot \frac{\pi}{180}, \tag{3.2}$$

ahol ϕ az azimut és θ a magassághoz tartozó szög. Lásd 3.4 ábra. A φ illetve λ pedig a méréspontok GCS (geographic coordinate system) koordinátái.

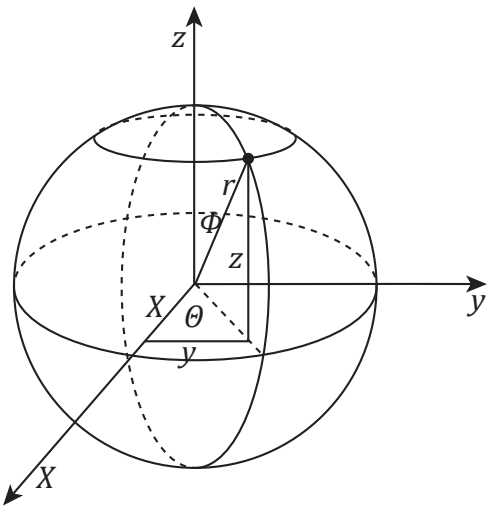
Innét pedig már könnyen átírható háromdimenziós Descartes-féle koordináta-rendszerbe, melynek origója a Föld középpontjában található:

$$X = r\cos(\theta) \cdot \cos(\phi) \tag{3.3}$$

$$Y = -r\sin(\theta) \tag{3.4}$$

$$Z = r\cos(\theta)\sin(\phi) \tag{3.5}$$

ahol r a műholdak átlagos pályamagasságát ($\equiv \bar{h}$) szimbolizáló konstans, ϕ és θ pedig a fentebb kiszámolt (3.1) és (3.2) gömbi-koordináták.



3.4. ábra. A koordináta-rendszerek közötti kapcsolat

3.3.3. Föld és szmogtérkép gömbfelület

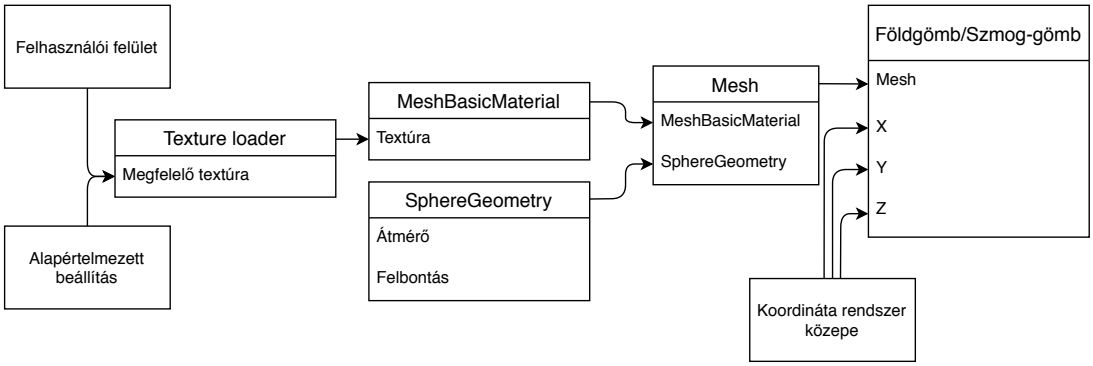
A 3.3 szekció elején olvasható módon a Föld és az azt körülvevő elektroszmog térkép is rendelkezik egy alapértelmezett textúrával az inicializációt követően. A Föld textúrázása hasonló, mint az elektroszmog térképé, annyi különbséggel, hogy az előbbi textúrái frekvenciafüggetlenek, míg az utóbbi frekvenciafüggő. Azonban mindkét esetben a folyamat hasonló. Először kiválasztjuk egy „TextureLoader" segítségével a megfelelő textúrát a neve alapján.

- Föld
  A későbbiekben (3.4. alfejezetben) leírt módon, a felhasználói felület segítségével, négy elem közül kiválasztott elem nevével rendelkező fájlt választjuk, a kiválasztott elem változtatását továbbiakban figyeljük.

- Elektroszmog térkép
  Az előző ponthoz képest az okoz nehézséget, hogy ez a textúra frekvenciafüggő. Tehát nem elég a textúra típusok közül a jót kiválasztani, a megfelelő intervallumra is figyelni kell. A felhasználói felületről három paraméter változását kell számon tartani, a minimum és maximum frekvencia, valamint a textúra típusát. Ezekből az adatokból áll össze a keresett textúra neve.

A helyes textúra betöltése után a méréspontokat modellező objektumok létrehozásához hasonló módon kell létrehozni a gömböket. Annyi könnyítés azonban van az imént említett metódushoz képest, hogy itt az X, Y és Z koordináták mind zérus értékűek, hisz ezek a koordináta rendszer közepében helyezkednek el. (Lásd 3.5 ábra) Az elkészült objektumokat külön rétegekre helyezzük, így könnyedén megjeleníthetőek vagy elrejthetőek lesznek.

Amennyiben a fenti felsorolásban említett attribútum változik, úgy a „TextureLoader"-rel kezdve az egész függvényláncot újra lefuttatjuk. Így érjük el az objektumok interaktivitását.



3.5. ábra. A Föld illetve a szmog „gömbök” textúrázása

3.4. Felhasználói felület

A felhasználó egy egész oldalt kitöltő weblappal találkozik, ahol megjelenik a jelmagyarázat az oldal bal alsó sarkában, középen a renderelt bolygót láthatja a mérés eredményeivel, míg a jobb felső sarokban az oldal vezérlésere szolgáló menüt találja. Az oldalon képes a bolygót forgatni, ha azt megragadja és elhúzza. Emellett a görgő segítségével képes a megjelenő képbe nagyítani vagy kicsinyíteni.

3.4.1. A menü

1. **Bolygó és méréspontok** (*Globe and Points*)

   - Bolygó textúra (*Globe Texture*)
     A Föld felszínén megjelenő textúrát lehet itt állítani. Négy opció közül lehet választani. Három textúra jeleníthető meg, egy műholdkép (*Satellite*) és két kontinenshatárokat tartalmazó térkép, világos (*Light coastlines*) és sötét (*Dark coastlines*). Az utóbbi kettő egyenletesebb színe miatt pontosabb RSSI érték leolvasást eredményez. Továbbá egy lehetőség a bolygó elrejtésére.

   - Méréspontok eltávolítása/megjelenítése (*Hide/Show measuring points*)
     A föld körül megjelenő méréspontokat lehet eltüntetni, illetve megjeleníteni.

2. **Elektroszmog térkép** (*Smog map*)

   - Elektroszmog térkép (*Smog map texture*)
     A bolygó körül megjelenő interpolált szmogtextúrát lehet itt állítani

     Öt opció közül lehet választani. Négy textúra. Továbbá egy lehetőség a textúra elrejtésére. Első lehetőségünk mérési adatok Vornoj (*Voronoi*) diagramjából készült textúra, ezt akkor érdemes használni, ha nem interpolált értékekre kíváncsi a felhasználó (lásd a 2.5.3 részt).

     Amennyiben azonban egy folytonos eloszlású adatok textúráját kívánja megjeleníteni a felhasználó, akkor lehetősége van egy Gauss-simítással elkészített folytonos (*Continous*) textúrát választani (lásd a 2.5.4 részt).

     Kiválasztható még, az ebből készített szintvonalas térkép is, szintvonal-felirattal (*Labeled contour*) vagy anélkül (*Contour*) (lásd a 2.5.5 részt).

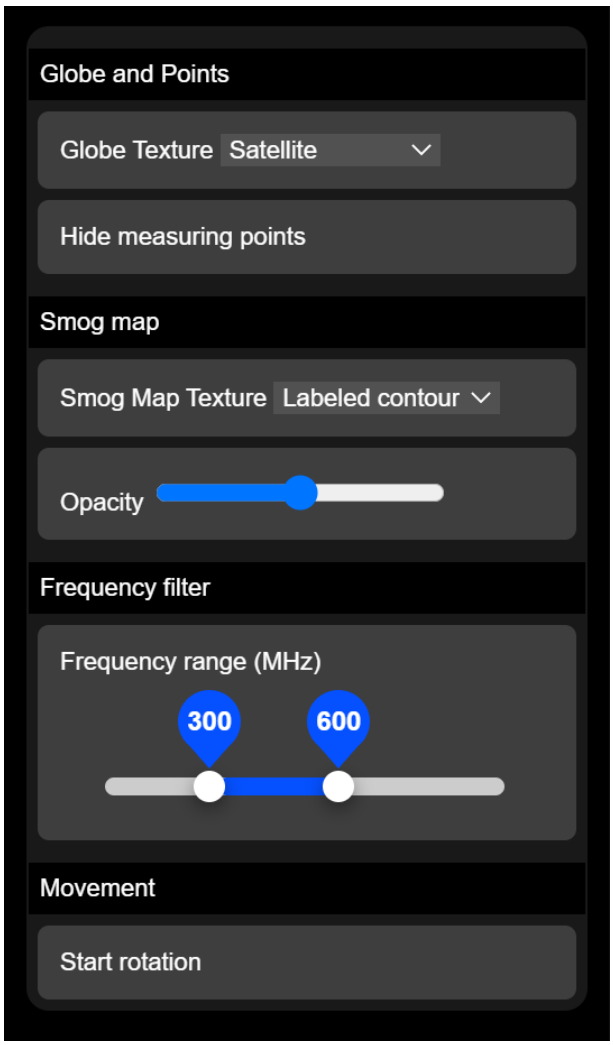
   - Átlátszóság (*Opacity*)
     A megjelenített térkép átlátszóságát lehet állítani pontosan.

3. **Frekvencia szűrés** (*Frequency filter*)

   - Frekvencia sáv (*Frequency range (MHz)*)
     100 MHz-es felbontással szűrhető a megjelenített adathalmaz. Ez hatással van a Föld körüli szmogtextúrára, valamint a méréspontokra is.

4. **Mozgás** (*Movement*)

   - Föld forgás be-/kikapcsolása (*Start/Stop rotation*)



3.6. ábra. A menü

4. fejezet

Kitekintés

Munkánk végeztével elkészítettünk egy teljes, működő adatfeldolgozási láncot. Ez képes a SMOG-P és ATL-1 műholdak által szolgáltatott nyers űrbéli rádiófrekvenciás adatokat előszűrni, feldolgozni és megjeleníteni, saját fejlesztésű programjaink segítségével. Ennek eredményeképpen többféle módon interpolált és szűrt térképeket tudunk előállítani tetszőleges térképi vetületen, valamint egy háromdimenziós, interaktív felületen is hozzáférhetővé tettük az adatfeldolgozás eredményét.

Jelen dolgozat célját nem képezte ezen eredmények értelmezése: egy olyan feldolgozási rendszert szerettünk volna létrehozni, mely a későbbi tudományos értelmezések alapjául szolgálhat.

4.1. Az adatfeldolgozási lánc jövője

A feldolgozási és megjelenítési lánc az előreláthatóan 2021 első felében pályára állított SMOG-1 műhold adatait is képes lesz majd kezelni. Ezen túl természetesen bármilyen más hasonló misszióval rendelkező műhold adatfeldolgozására is képes lehet kis módosítások után. Mivel a SMOG-1 a tervek szerint egy jelentősen magasabb pályán fog keringeni, fontos, hogy a teljes jelfeldolgozási lánc paramétere a pályamagasság, így különböző keringési magasságú műholdak adatainak feldolgozása sem fog gondot jelenteni.

Rövidtávú célunk az adatbázis bővítése, hiszen minél több adat áll rendelkezésre, annál pontosabb lesz a kapott térkép, a becslés annál közelebb lesz a valósághoz.

4.1.1. Adatfeldolgozás fejlesztési lehetőségei

Az adatfeldolgozás jelenleg aszinkron módon működik, a megfelelő szkriptek manuális indításával történik. Ennek az az elsődleges oka, hogy a feldolgozáshoz szükséges idő- és memóriaigény viszonylag nagy, a műholdak által végzett adatgyűjtés pedig nem volt olyan gyors, hogy sűrű frissítést igényeltek volna az eredmények. Több műhold esetében azonban érdemes lenne automatizált módon, a központi szerveren rendszeres adatfeldolgozást futtatni pl. naponta egyszer. Ehhez a szerver SQL adatbázisával való automatikus kommunikációt kellene csak implementálni, minden más készen áll ehhez a szoftveres oldalról.

Hosszú távú célként az is felmerülhet, hogy az adatfeldolgozás időigényesebb részeit SQL

lekérések formájában valósítsuk meg direktben az adatbázison, akár valós időben. Ezt a jelen dolgozatban végzett munka során azért vetettük el, mert a kezdeti fejlesztést, kísérletezést jelentősen könnyebbé tette a rugalmas szkriptelés. Ha azonban az eredményeink alapján a módszereink hosszú távon is használhatónak bizonyulnak, érdemes lenne ezt a lehetőséget újra megvizsgálni.

Az adatfeldolgozás elméleti oldaláról is számos fejlesztési lehetőség kínálkozik. Az elsődleges fejlesztési lehetőség a térbeli interpoláció: megfelelően sok, különböző pályamagasságon végzett méréssel megfigyelhető lenne, hogy magasság függvényében hogyan változik a rádiófrekvenciás szennyezettség ábrázolása, illetve egy adott magasságra is pontosabb térképek készülhetnének, mint a jelenlegi redukciós közelítéssel.

Más interpolációs módszereket is érdemes lenne kripóbálni: a krigelés pl. kifejezetten ígéretes lehetőség, hiszen egy térbeli mező mérések alapján legvalószínűbb eloszlásának meghatározására alkalmas. Ehhez megítélésünk szerint vagy több adatra lenne szükség, hogy minél több mérési ponthoz rendelkezzünk megfelelő műhold-orientációban végzett méréssel, vagy a műhold orientációjának jobb ismeretére ill. jobb térbeli szűrési módszerre lenne szükség.

4.1.2. Weboldal fejlesztési lehetőségei

A weboldal továbbfejleszthető lenne:

- Backend alkalmazásával,
  Egy korszerű backend használatával lehetőség nyílna adatbázis kezelésre. Ez tovább egyszerűsíthetné létrehozott objektumok kezelését, számontartását.

- A méréspontok adatainak tárolása
  A szűréssel és összevonással számos nyers mérési eredményt elrejtünk a felhasználó elől. Ezen lehetne javítani, ha az egyes méréspontok helyein lévő objektumok is interaktívak lennének, és azokkal interakcióba lépve lekérhetőek lennének a mérés rendelkezésre álló további körülményei, eredményei.

- A mérési adatok térbeli kezelése
  Hosszabb távú fejlesztési lehetőség, azonban tanulságos következtetéseket lehetne levonni, ha nem csak földrajzi koordináták szerint készülne el a térkép, hanem a mérések magasságát is figyelembe vennénk.

- Más típusú mérési eredmények megjelenítése
  A webalkalmazás sok szempontból független az adatok tartalmával: más, térbeli helyhez kötődő mérés eredményeinek megjelenítésére is lehetne alkalmazni.

4.2. Az eredmények alkalmazhatósága

A műholdak tervezői és üzemeltetői által létrehozott platformra építve, munkánk eredményeképp a világon először készült több, a Földet körülvevő rádiófrekvenciás, elektromágneses szennyezettséget ábrázoló térkép.

Ennek felhasználási lehetőségei igen sokrétűek. Rávilágíthat arra, hogy mennyire pazarlóak a jelenleg használt infokommunikációs eljárásaink: a földi telekommunikációs szolgál-

tatók által kibocsájtott jelek valóban észlelhetőek a Föld körüli pályán, annak ellenére, hogy erre valójában semmi szükség. Különféle műholdak tervezőinek, üzemeltetőinek is hasznára válhat egy ilyen térkép, hiszen ha az eddigi tendencia folytatódik, egyre rosszabb jel-zajviszonnyal kell számolni jövőbeli műholdjaink esetén. Ezt le lehet küzdeni nagyobb adóteljesítménnyel a földi állomás-műhold viszonylatban, de még jobb lenne az igazán sok elektromágneses szennyezést kibocsájtó területek adásait optimalizálni: ebben is segíthet egy ilyen térkép. A térkép azonban önmagában csak egy eszköz azok közül, melyek egy ilyen munkát segíthetnek. A részletes spektrum-adatok nagyobb felbontású feldolgozása is szükséges lenne a legnagyobb szennyezők meghatározásához.

Az általunk készített térképek nem csak tudományos, hanem oktatási és ismeretterjesztő célokra is alkalmazhatóak. Az elektromágneses tér és műszaki felhasználása meglehetősen absztrakt dolog, azonban a mi térképeink látványosan és könnyen értelmezhető módon jelenítik meg a Föld lakóinak ilyen vetületű tevékenységeit. Az űrbéli perspektíva más nézőpontba helyezheti a TV-adók, mobiltelefonok mindennapi, mégis különös világát. A térképekre ránézve egyértelmű, hogy az emberiség – sok más dologhoz hasonlóan – egyetlen elektromágneses teren osztozik, ennek átgondolt, pazarlás nélküli felhasználása mindannyiunk közös érdeke.

Köszönetnyilvánítás

Köszönettel tartozunk Dr. Dudás Levente konzulensünknek támogatásáért, hasznos meglátásaiért és végtelen lelkesedéséért; Dr. Gschwindt András Bandi bácsinak, hogy megkereste a BME Kozmosz Kört ezzel a lehetőséggel, majd rendszeres tanácsokkal segítette a munkánk; Hödl Emil Viktornak, hogy a műholddal és üzemeltetésével, valamint a honlap elérhetővé tételével kapcsolatos kérdéseinkre mindig türelmesen és tartalmasan válaszolt.

Hálásak vagyunk és gratulálunk a *Zsebi* projekt minden tagjának, akik sok éves kemény munkájukkal lehetővé tették, hogy két hibátlanul működő magyar műegyetemi műhold adataira építhessük ezt a dolgozatot.

Markotics Boldizsár köszöni továbbá Kovács Marcellnek, hogy kérdéseivel fordulhatott hozzá véleményéért, projekt kezdetétől a végéig.

Végül Takács Donát köszönettel tartozik az Építőmérnöki Kar Általános- és Felsőgeodézia Tanszékéről Dr. Rózsa Szabolcs és Dr. Toronyi Bence tanár uraknak, akik nem csak lehetővé tették számára a karok és szakok közötti rendhagyó áthallgatást, de el is mélyítették benne a geodézia iránti érdeklődést, és átadott tudásukkal közvetve stabilabb elméleti alapokra helyezték egy gépész- és egy villamosmérnök hallgató által írt dolgozat földrajzi gondolatait, számításait.

Irodalom

[1] Astropy Collaboration és tsai. „Astropy: A community Python package for astronomy”. *\aap* 558 (2013. okt.). _eprint: 1307.6212, A33. DOI: 10.1051/0004-6361/201322068.

[2] Astropy Collaboration és tsai. „The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package”. *aj* 156.3 (2018. szept.). _eprint: 1801.02634, 123. old. DOI: 10.3847/1538-3881/aabc4f.

[3] Marco A. Azpurua és K. Dos Ramos. „A comparison of spatial interpolation methods for estimation of average electromagnetic field magnitude”. *Progress In Electromagnetics Research M* 14 (2010), 135–145. old. ISSN: 1937-8726. DOI: 10.2528/PIERM10083103. URL: http://www.jpier.org/PIERM/pier.php?paper=10083103 (elérés dátuma 2020. 10. 22.).

[4] *Catalog of Earth Satellite Orbits*. Publisher: NASA Earth Observatory. 2009. szept. 4. URL: https://earthobservatory.nasa.gov/features/OrbitsCatalog/page2.php (elérés dátuma 2020. 10. 23.).

[5] Levente Dudás, László Szűcs és András Gschwindt. „The spectrum monitoring system by Smog-1 satellite”. *2015 Conference on Microwave Techniques (COMITE)*. 2015 Conference on Microwave Techniques (COMITE). Pardubice, Czech Republic: IEEE, 2015. ápr., 1–4. old. ISBN: 978-1-4799-8121-2 978-1-4799-8123-6. DOI: 10.1109/COMITE.2015.7120316. URL: http://ieeexplore.ieee.org/document/7120316/ (elérés dátuma 2020. 10. 23.).

[6] Gábor Géczy. „A SMOG-1 PocketQube másodlagos energiaellátó rendszere”. MSc diplomaterv. Budapest: Budapesti Műszaki és Gazdaságtudományi Egyetem, 2016. URL: http://gnd.bme.hu/smog/files/publikaciok/gabor_eps2/Geczy_Gabor_MSc_Diplomaterv_3_v0_t.pdf.

[7] Charles R. Harris és tsai. „Array programming with NumPy”. *Nature* 585.7825 (2020. szept. 17.), 357–362. old. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-020-2649-2. URL: http://www.nature.com/articles/s41586-020-2649-2 (elérés dátuma 2020. 10. 22.).

[8] John D. Hunter. „Matplotlib: A 2D Graphics Environment”. *Computing in Science & Engineering* 9.3 (2007), 90–95. old. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.55. URL: http://ieeexplore.ieee.org/document/4160265/ (elérés dátuma 2020. 10. 22.).

[9] Tibor Kálmán. „SMOG-1 műhold földi állomás kliensszoftver fejlesztése”. BSc szakdolgozat. Budapest: Budapesti Műszaki és Gazdaságtudományi Egyetem, 2016. URL: http://gnd.bme.hu/smog/files/publikaciok/kalman_tibor/kalman_tibor_szakdolgozat.pdf.

[10] László Kozma és Zoltán Kovács. *Görbék és felületek elemi differenciálgeometriája*. Debrecen-Nyíregyháza, 2011. URL: http://zeus.nyf.hu/~kovacsz/dghome/index.html.

[11] András Krauter. *Geodézia*. Budapest: Műegyetemi Kiadó, 1995. 346 old.

[12] B. Li és tsai. „Method for yielding a database of location fingerprints in WLAN". *IEE Proceedings - Communications* 152.5 (2005), 580. old. ISSN: 13502425. DOI: 10.1049/ip-com:20050078. URL: https://digital-library.theiet.org/content/journals/10.1049/ip-com_20050078 (elérés dátuma 2020. 10. 22.).

[13] Wes McKinney. „Data Structures for Statistical Computing in Python". Python in Science Conference. Austin, Texas, 2010, 56–61. old. DOI: 10.25080/Majora-92bf1922-00a. URL: https://conference.scipy.org/proceedings/scipy2010/mckinney.html (elérés dátuma 2020. 10. 22.).

[14] Met Office. *Cartopy: a cartographic python library with a matplotlib interface*. Exeter, Devon, 2010. URL: https://scitools.org.uk/cartopy.

[15] Louis Moresi és Ben Mather. „Stripy: A Python module for (constrained) triangulation in Cartesian coordinates and on a sphere." *Journal of Open Source Software* 4.38 (2019. jún. 13.), 1410. old. ISSN: 2475-9066. DOI: 10.21105/joss.01410. URL: http://joss.theoj.org/papers/10.21105/joss.01410 (elérés dátuma 2020. 10. 22.).

[16] Levente Pápay. *SMOG-1 Antennanyitás*. URL: http://gnd.bme.hu/smog/files/publikaciok/levi_ant/SMOG_1_antennanyitas.pdf.

[17] Robert J. Renka. „Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere". *ACM Transactions on Mathematical Software* 23.3 (1997. szept.), 416–434. old. ISSN: 0098-3500, 1557-7295. DOI: 10.1145/275323.275329. URL: https://dl.acm.org/doi/10.1145/275323.275329 (elérés dátuma 2020. 10. 22.).

[18] Robert J. Renka. „Algorithm 773: SSRFPACK: interpolation of scattered data on the surface of a sphere with a surface under tension". *ACM Transactions on Mathematical Software* 23.3 (1997. szept.), 435–442. old. ISSN: 0098-3500, 1557-7295. DOI: 10.1145/275323.275330. URL: https://dl.acm.org/doi/10.1145/275323.275330 (elérés dátuma 2020. 10. 22.).

[19] *Three.js documentation*. Three.js Authors. 2020. URL: https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene (elérés dátuma 2020. 10. 20.).

[20] Kristóf Timur. „SMOG-1 fedélzeti számítógép fejlesztése". BSc szakdolgozat. Budapest: Budapesti Műszaki és Gazdaságtudományi Egyetem, 2017. URL: http://gnd.bme.hu/smog/files/publikaciok/timur_obc/SMOG1-fedelzeti-szamitogep-fejlesztese-Dolgozat-5.pdf.

[21] RJ Twiggs. „Making it small". *Cal Poly Developers' Workshop*. 2009.

[22] *Űrvilág.hu - A SMOG és az ATL-1*. URL: http://www.urvilag.hu/velemenyek/20201011_a_smog_es_az_atl1 (elérés dátuma 2020. 10. 23.).

[23] *Űrvilág.hu - Magyar világrekord az űrben*. URL: http://www.urvilag.hu/hazai_kutatohelyek_es_uripar/20191216_magyar_vilagrekord_az_urben (elérés dátuma 2020. 10. 23.).

[24] Pauli Virtanen és tsai. „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". *Nature Methods* 17 (2020), 261–272. old. DOI: 10.1038/s41592-019-0686-2.

Ábrák jegyzéke

Táblázatok jegyzéke

A. függelék

SQL lekérdezés

```
SELECT
  smogp_spa.timestamp,
  smogp_spa_data.frequency,
  smogp_spa_data.rssi,
  ROUND(smogp_position.longitude,0) AS longitude,
  ROUND(smogp_position.latitude,0) AS latitude,
  smogp_position.altitude
FROM
  smogp_spa INNER JOIN smogp_spa_data ON smogp_spa_data.spa_id =
   smogp_spa.id INNER JOIN smogp_position ON    smogp_position.datetime
   = smogp_spa.timestamp
UNION ALL
SELECT
  smogp_spa_measurement.timestamp,
  smogp_spa_data_measurement.frequency,
  smogp_spa_data_measurement.rssi,
  ROUND(smogp_position.longitude,0)  AS longitude,
  ROUND(smogp_position.latitude,0) AS latitude,
  smogp_position.altitude
FROM
  smogp_spa_measurement INNER JOIN smogp_spa_data_measurement ON
   smogp_spa_data_measurement.spa_id = smogp_spa_measurement.id INNER
   JOIN smogp_position ON smogp_position.datetime =
   smogp_spa_measurement.timestamp
```

B. függelék

Adatfeldolgozás implementációja

Az SQL-adatbázisból letöltött adatokat Python nyelven írt szkriptekkel dolgoztuk fel. A fő szkript a `final_generator.py` (B.2), melynek megadhatóak a 2.6. táblázat szerinti paraméterek. Ez két eljárást hív meg, melyeket a `generate_texture.py` (B.3) és `generate_texture_freq.py` (B.4) fájlok tartalmaznak. Előbbi a teljes mérési frekvencia-tartományra készíti el a textúrákat és a szűrt-interpolált adatpontokat, míg az utóbbi az interaktív felületen kiválasztható frekvencia-tartományokra generálja a megfelelő textúrákat a (2.10) szerinti adathalmazokból. A gyakran használt eljárásokat az `rssi_processing.py` (B.1) fájl tartalmazza. Mivel az olvasót elsősorban ezek érdekelhetik, ezt közöljük először, azonban a teljesség kedvéért a többi részt is megadjuk lentebb.

B.1. Gyakran használt metódusok

```
# rssi_processing.py

import pandas as pd
import matplotlib.pyplot as plt

import numpy as np
from numpy.linalg import norm

from datetime import datetime, timedelta, timezone

from scipy.integrate import trapz
from scipy.interpolate import griddata
from scipy.ndimage import gaussian_filter

import stripy as stripy
from stripy.spherical import lonlat2xyz

from math import pi, cos, sin

import cartopy
import cartopy.crs as ccrs

from astropy.convolution import convolve_fft, Gaussian2DKernel

import matplotlib as mpl


```

```
def select_usable_data(
    csv, spatial_res_deg=1, frequency_res_mhz=100, reduce_by_altitude=True
):
    # Magasság-redukció
    mean_altitude = csv["altitude"].mean()
    if reduce_by_altitude:
        csv["rssi"] = power_to_rssi(
            rssi_to_power(csv["rssi"]) * csv["altitude"] ** 2 / mean_altitude ** 2
        )
    rssi_df = csv.drop(columns=["timestamp", "altitude"])

    # Hz->MHz
    rssi_df["frequency_mhz"] = rssi_df["frequency"] / 1000000
    rssi_df.drop(columns=["frequency"], inplace=True)
    print("Kiinduló adathalmaz mérete: {0} sor".format(len(rssi_df)))

    # Csoportosítások térbeli és frekvencia-régiók szerint
    spatial_res_deg_lat = spatial_res_deg
    spatial_res_deg_lon = spatial_res_deg

    rssi_df["lat_rounded"] = (
        spatial_res_deg_lat * (rssi_df["latitude"] / spatial_res_deg_lat).round()
    )
    rssi_df["long_rounded"] = (
        spatial_res_deg_lon * (rssi_df["longitude"] / spatial_res_deg_lon).round()
    )
    rssi_df["freq_rounded"] = frequency_res_mhz * (
        (rssi_df["frequency_mhz"] / frequency_res_mhz).round()
    )

    grouped_rssi = rssi_df.groupby(["lat_rounded", "long_rounded", "freq_rounded"])
    max_idx = grouped_rssi["rssi"].transform(max) == rssi_df["rssi"]
    max_rssi = rssi_df[max_idx].copy()
    max_grouped = max_rssi.groupby(["lat_rounded", "long_rounded", "freq_rounded"])

    # Összevonás
    rssi_ret = pd.DataFrame(
        columns=["long_rounded", "lat_rounded", "long", "lat", "freq_mhz", "rssi"]
    )
    rssi_ret["lat"] = max_grouped["latitude"].mean().values
    rssi_ret["long"] = max_grouped["longitude"].mean().values
    rssi_ret["freq_mhz"] = max_grouped["frequency_mhz"].mean().values
    rssi_ret["rssi"] = max_grouped["rssi"].max().values.astype("float")

    max_grouped_reset = max_grouped["rssi"].max().reset_index()
    rssi_ret["lat_rounded"] = max_grouped_reset["lat_rounded"]
    rssi_ret["long_rounded"] = max_grouped_reset["long_rounded"]

    # Kerekítés
    rssi_ret["lat"] = np.round(rssi_ret["lat"], decimals=2)
    rssi_ret["long"] = np.round(rssi_ret["long"], decimals=2)

```

```
80      print("Összevont adathalmaz mérete: {0} sor".format(len(rssi_ret)))
81
82      return mean_altitude, rssi_ret.sort_values(by=["long", "lat", "
        freq_mhz"])
83
84
85  def import_usable(csv_files, spatial_res_deg=1, frequency_res_mhz=100):
86      # Több, RSSI-méréseket tartalmazó CSV fájlt olvas be, melyeket egyes
        ít és
87      # térben és frekvencia szerint csoportosít, majd szűr.
88
89      if not isinstance(csv_files, (list,)):
90          raise ValueError("Input must be a list.")
91          return
92      else:
93          pass
94
95      spa_data = []
96
97      for csv_file in csv_files:
98          spa_data.append(pd.read_csv(csv_file))
99
100     concat = pd.concat(spa_data, ignore_index=True)
101
102     del spa_data  # Manuális GC, hogy memóriát spóroljunk
103
104     return select_usable_data(concat, spatial_res_deg, frequency_res_mhz
        )
105
106
107 def rssi_to_power(rssi):
108     return 10 ** (rssi / 10)
109
110
111 def power_to_rssi(power):
112     return 10 * np.log10(power)
113
114
115 def clean_x_gaps(x, threshold=1):
116     # Cleans gaps in x (e.g. frequency or time) arrays
117     # longer than threshold
118     diff = np.ediff1d(x)
119     diff2 = diff[diff > threshold] - 1
120     indices = np.indices(diff.shape)[
121         0,
122     ][diff > threshold]
123     subtract = np.zeros((len(diff2), len(x)))
124     i1, i2 = np.indices(subtract.shape)
125     subtract[i2 > np.repeat(np.atleast_2d(indices), len(x), axis=0).T] =
        1
126     subtract = np.sum(subtract * np.atleast_2d(diff2).T, axis=0)
127     return x - subtract
128
129
130 def rssi_power_rms(frequency, rssi, gap_threshold_mhz=10):
131     # Bemenet: rssi
132     # Kimenet: RSSI-ből számított teljesítmény RMS-e, RSSI-be visszaszá
        molva
133
```

```
    if len(frequency) > 1:
        return power_to_rssi(
            np.sqrt(
                trapz(x=frequency, y=rssi_to_power(rssi) ** 2)
                / (frequency.max() - frequency.min())
            )
        )
    else:
        return rssi[0]


def rssi_power_rms_fp(frequency, rssi, gap_threshold_mhz=10):
    # Bemenet: rssi
    # Kimenet: RSSI-ből számított teljesítmény RMS-e, RSSI-be visszaszámolva
    # Kiterjeszti az rssi_power_rms eljárást úgy, hogy a nagy réseket
    # kihagyja az integrálásból

    if len(frequency) > 1:
        freq_compact = clean_x_gaps(frequency, gap_threshold_mhz)
        return power_to_rssi(
            np.sqrt(
                trapz(x=freq_compact, y=rssi_to_power(rssi) ** 2)
                / (freq_compact.max() - freq_compact.min())
            )
        )
    else:
        return rssi[0]


def average_grouped(grouped, averaging_method="power_rms", gap_threshold_mhz=10):
    # Bemenet: (n,7) DataFrame, n: frekvenciák
    # Kimenet: (1,7) Series
    if averaging_method == "power_rms_gapless":
        rssi_avg = rssi_power_rms_fp(
            grouped["freq_mhz"].values,
            grouped["rssi"].values,
            gap_threshold_mhz
        )
    elif averaging_method == "power_rms":
        rssi_avg = rssi_power_rms(
            grouped["freq_mhz"].values,
            grouped["rssi"].values)
    elif averaging_method == "rssi_mean":
        rssi_avg = grouped["rssi"].mean()

    long = grouped["long"].mean()
    lat = grouped["lat"].mean()

    return pd.Series(
        data={
            "long": long,
            "lat": lat,
            "rssi": rssi_avg,
        }
    )


```

```
def average_rounded_rssi(rssi_rounded, gap_threshold_mhz=None):
    return (
        rssi_rounded.sort_values(by="freq_mhz")
        .groupby(by=["long_rounded", "lat_rounded"])
        .apply(
            lambda grouped: average_grouped(
                grouped, gap_threshold_mhz=gap_threshold_mhz
            )
        )
        .reset_index(drop=True)
    )


def xyz_dist_latlon(lons1, lats1, lons2, lats2):
    x1, y1, z1 = lonlat2xyz(lons1, lats1)
    x2, y2, z2 = lonlat2xyz(lons2, lats2)
    return np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2 + (z2 - z1) ** 2)


def IDW_interpolate_rssi(
    lons_rad, lats_rad, rssi, mesh, los_distance_limit, mean_altitude
):

    power = rssi_to_power(rssi)

    Re = 6378  # Föld sugara, km
    Re += mean_altitude  # Erre a magasságra redukáltuk a méréseket
    # Gömbi távolság, \tilde{d}*
    distance_limit = np.arcsin(los_distance_limit / Re) * Re

    # Meg kell adni, hogy hány pont lehet maximum a kívánt távolságban
    # Ez egy elegendően jó becslés lesz, hogy ne indexeljünk ki
    A_cap = 2 * pi * Re ** 2 * (1 - cos(distance_limit / Re))
    A_sphere = 4 * Re ** 2 * pi
    npoints = np.round(0.9 * A_cap / A_sphere * len(mesh.lons))

    # Interpolációban részt vevő pont-környezetek kiválasztása
    # A stripy szerint a gömb sugara egységnyi!
    distances, vertices = mesh.nearest_vertices(
        lons_rad, lats_rad, k=npoints, max_distance=distance_limit / Re
    )

    # Maga az interpoláció
    numerators = np.zeros_like(mesh.lons)
    denominators = np.zeros_like(mesh.lons)

    for i in range(0, vertices.shape[0]):
        vi = vertices[i, :]
        di = xyz_dist_latlon(lons_rad[i], lats_rad[i], mesh.lons[vi],
   mesh.lats[vi])
        wi = 1 / di ** 2
        numerators[vi] += wi * power[i]
        denominators[vi] += wi

    power_idw_mesh = numerators / denominators
    rssi_idw_mesh = power_to_rssi(power_idw_mesh)

    return rssi_idw_mesh

```

```
249
250 def voronoi_rssi(lons_rad, lats_rad, rssi, fmesh):
251     measurement_triangulation = stripy.sTriangulation(
252         lons=lons_rad, lats=lats_rad, permute=True
253     )
254
255     interp_fine, err = measurement_triangulation.interpolate(
256         fmesh.lons, fmesh.lats, order=0, zdata=rssi
257     )
258
259     return interp_fine
260
261
262 def plate_carree_interpolation(
263     lons_rad,
264     lats_rad,
265     rssi,
266     points_per_deg=12,
267     gauss_sigma_deg=None,
268     ext=20,
269     wraparound=True,
270     method="linear",
271 ):
272
273     # Ez a vetítés szükséges a 3D-s vizualizáció textúráihoz
274
275     ext = 20
276     ngridx = (360 + 2 * ext) * points_per_deg
277     ngridy = (180 + 2 * ext) * points_per_deg
278
279     xi = np.linspace(-180 - ext, 180 + ext, ngridx, dtype="float16")
280     yi = np.linspace(-90 - ext, 90 + ext, ngridy, dtype="float16")
281
282     lons0 = np.degrees(lons_rad)
283     lats0 = np.degrees(lats_rad)
284
285     lons_rad = None
286     lats_rad = None
287     del lons_rad
288     del lats_rad
289
290     # wraparound
291     right_wrap = lons0 < (-180 + ext)
292     left_wrap = lons0 > (180 - ext)
293     lons0 = np.append(
294         lons0, np.concatenate((lons0[right_wrap] + 360, lons0[left_wrap]
    - 360))
295     )
296     lats0 = np.append(lats0, np.concatenate((lats0[right_wrap], lats0[
    left_wrap])))
297     rssi = np.append(rssi, np.concatenate((rssi[right_wrap], rssi[
    left_wrap])))
298
299     # vertical mirror
300     top_mirror = lats0 > (90 - ext)
301     bottom_mirror = lats0 < (-90 + ext)
302     lons0 = np.append(lons0, np.concatenate((lons0[top_mirror], lons0[
    bottom_mirror])))
303     lats0 = np.append(
```

```
        lats0, np.concatenate((-lats0[top_mirror] + 180, -lats0[
    bottom_mirror] - 180))
    )
    rssi = np.append(rssi, np.concatenate((rssi[top_mirror], rssi[
    bottom_mirror])))

    zi = griddata((lons0, lats0), rssi, (xi[None, :], yi[:, None]),
    method=method)

    lons0 = None
    lats0 = None
    del lons0
    del lats0
    if gauss_sigma_deg is not None:
        kernel = Gaussian2DKernel(x_stddev=gauss_sigma_deg *
    points_per_deg)
        zi = convolve_fft(zi, kernel, allow_huge="True", complex_dtype=
    np.complex64)

    edge = ext * points_per_deg
    zi = zi[edge:-edge, edge:-edge]
    return xi[edge:-edge], yi[edge:-edge], zi


def plot_texture(zi, filename, cmap="viridis", **kwargs):
    # Textúrák a 3D-s térképhez
    fig = plt.figure(frameon=False, figsize=(16, 8))
    fig.subplots_adjust(bottom=0)
    fig.subplots_adjust(top=1)
    fig.subplots_adjust(right=1)
    fig.subplots_adjust(left=0)

    plt.axis("off")
    plt.imshow(
        zi, cmap=cmap, origin="lower", aspect="auto", interpolation="
    lanczos", **kwargs
    )
    plt.savefig(filename, dpi=2048 / 4, transparent=True)
    plt.close(fig)


def plot_contour_level_texture(
    xi,
    yi,
    zi,
    filename,
    levels=9,
    vmin=-120,
    vmax=-40,
    cmap="viridis",
    transparent=True,
    labels=True,
):
    fig = plt.figure(frameon=False, figsize=(16, 8))
    fig.subplots_adjust(bottom=0)
    fig.subplots_adjust(top=1)
    fig.subplots_adjust(right=1)
    fig.subplots_adjust(left=0)

```

```
    plt.axis("off")
    CS = plt.contour(xi, yi, zi, levels=levels, vmin=vmin, vmax=vmax,
    cmap=cmap)
    if labels is True:
        CS.levels = ["{0:.0f} dBm".format(l) for l in CS.levels]
        plt.clabel(CS, CS.levels, inline=True, fontsize=10)
    plt.savefig(filename, dpi=2048 / 4, transparent=transparent)
    plt.close(fig)


def plot_mollweide(
    lons_rad,
    lats_rad,
    rssi,
    filename,
    cmap="viridis",
    coastline_color="#777777",
    plot_measurement_points=False,
    lons_meas_rad=None,
    lats_meas_rad=None,
    figsize=(10, 5),
    s=5,
    transparent=False,
    colorbar=False,
    color_by_rssi=True,
    **kwargs
):

    fig = plt.figure(figsize=figsize, facecolor="none", dpi=150)
    ax = plt.subplot(111, projection=ccrs.Mollweide())

    ax.coastlines(color=coastline_color)
    ax.set_global()

    lons0 = np.degrees(lons_rad)
    lats0 = np.degrees(lats_rad)

    if not color_by_rssi:
        rssi = None

    sc = ax.scatter(
        lons0,
        lats0,
        marker="o",
        s=s,
        alpha=1,
        transform=ccrs.PlateCarree(),
        c=rssi,
        cmap=cmap,
        label="rssi (dBm)",
        **kwargs
    )

    lonticks = np.arange(-180, 180 + 1, 20)
    latticks = np.arange(-90, 90 + 1, 20)
    ax.gridlines(xlocs=lonticks, ylocs=latticks, crs=ccrs.PlateCarree())
    ax.grid(linewidth=2, color="black", alpha=0.5, linestyle="--")

    if colorbar:
```

```
        plt.colorbar(sc)

    if plot_measurement_points:
        ax.scatter(
            np.degrees(lons_meas_rad),
            np.degrees(lats_meas_rad),
            marker="o",
            s=5.0,
            alpha=0.9,
            transform=ccrs.Geodetic(),
            c="purple",
        )

    plt.savefig(
        filename, transparent=transparent,
        facecolor="white", bbox_inches="tight"
    )
    plt.close(fig)


def dB_to_hex_color(values, cmap, vmin, vmax):
    norm = mpl.colors.Normalize(vmin=vmin, vmax=vmax)
    colormap = plt.get_cmap(cmap)
    mapped = colormap(norm(values))
    rgb255 = np.round(mapped[:, :3] * 255)
    vhex = np.vectorize(hex)
    return vhex(
        np.asarray(np.sum(rgb255 * np.array([256 ** 2, 256, 1]), axis=1)
    , "int")
    )
```

B.2. Fő szkript

```python
# final_generator.py

from generate_texture import create_textures
from generate_texture_freq import create_textures_freq
import warnings

kwargs = {
    "cmap": "nipy_spectral",
    "spatial_res_deg": 5,
    "frequency_res_mhz": 50,
    "gap_threshold_mhz": 50,
    "los_distance_limit": 2100,
    "icosahedral_mesh_refinement_level": 7,
    "points_per_deg": 6,
    "gauss_sigma_deg": 3,
    "vmin": -105,
    "vmax": -40,
    "files": ["smogp_spa_1022.csv", "atl1_spa_1022.csv"],
}
warnings.filterwarnings("ignore", message="RuntimeWarning: invalid value encountered in true_divide")

print("Textúrák a teljes frekvencia-tartományra...")
create_textures(**kwargs)
print("Textúrák az egyes frekvencia-tartományokra...")
create_textures_freq(**kwargs)
```

B.3. Általános textúrák generálása

```python
# generate_texture.py

from rssi_processing import *


def create_textures(
    files,
    spatial_res_deg,
    frequency_res_mhz,
    gap_threshold_mhz,
    los_distance_limit,
    icosahedral_mesh_refinement_level,
    points_per_deg,
    gauss_sigma_deg,
    vmin,
    vmax,
    cmap,
):

    # Előszűrés
    mean_altitude, rssi_rounded = import_usable(
        files,
        spatial_res_deg=spatial_res_deg,
        frequency_res_mhz=frequency_res_mhz
    )

    # RSSI-átlagolás
    rssi_averaged = average_rounded_rssi(rssi_rounded)
    print('RSSI-átlagolt adatsorok száma: {0}'.format(len(rssi_averaged)
    ))

    # Exportálás a 3D-s térképhez
    rssi_averaged_renamed = rssi_averaged.copy()
    rssi_averaged_renamed.rename(
        columns={"long": "X", "lat": "Y", "rssi": "Rssi"}, inplace=True
    )

    rssi_averaged_renamed["color"] = dB_to_hex_color(
        rssi_averaged_renamed["Rssi"].values, cmap=cmap, vmin=vmin, vmax
    =vmax
    )
    rssi_averaged_renamed.to_json("all_frequencies.json", orient="
    records")

    # Interpoláció
    mesh = stripy.spherical_meshes.icosahedral_mesh(
        refinement_levels=icosahedral_mesh_refinement_level,
        include_face_points=True,
        tree=True,
    )

    lons_rad = np.radians(rssi_averaged["long"].values)
    lats_rad = np.radians(rssi_averaged["lat"].values)
    rssi = rssi_averaged["rssi"].values

    base_filename = "all_frequencies"
    ### Ábrázolás ###

```

```
    # Selected points
    plot_mollweide(
        lons_rad,
        lats_rad,
        rssi,
        "mollweide_selected_" + base_filename + ".png",
        cmap="Reds",
        color_by_rssi=False,
        s=2,
        vmin=vmin,
        vmax=vmax,
    )
    # IDW
    rssi_idw_mesh = IDW_interpolate_rssi(
        lons_rad, lats_rad, rssi, mesh, los_distance_limit=
    los_distance_limit,
        mean_altitude=mean_altitude
    )

    xi, yi, zi = plate_carree_interpolation(
        mesh.lons,
        mesh.lats,
        rssi_idw_mesh,
        points_per_deg=points_per_deg,
        gauss_sigma_deg=gauss_sigma_deg,
    )

    plot_texture(
        zi,
        filename="textures/continous_" + base_filename + ".png",
        cmap=cmap,
        vmin=vmin,
        vmax=vmax,
    )

    levels = np.arange(-120, -40 + 1, 5)

    plot_contour_level_texture(
        xi,
        yi,
        zi,
        "textures/contour_labeled_" + base_filename + ".png",
        vmin=vmin,
        vmax=vmax,
        cmap=cmap,
        labels=True,
        levels=levels,
    )

    plot_contour_level_texture(
        xi,
        yi,
        zi,
        "textures/contour_" + base_filename + ".png",
        vmin=vmin,
        vmax=vmax,
        cmap=cmap,
        labels=False,
        levels=levels,
```

```
    )

    # Voronoi
    gauss_sigma_deg = 0.1
    rssi_voronoi_mesh = voronoi_rssi(lons_rad, lats_rad, rssi, mesh)
    xi, yi, zi = plate_carree_interpolation(
        mesh.lons,
        mesh.lats,
        rssi_voronoi_mesh,
        points_per_deg=points_per_deg,
        gauss_sigma_deg=gauss_sigma_deg,
    )

    plot_texture(
        zi,
        filename="textures/voronoi_" + base_filename + ".png",
        cmap=cmap,
        vmin=vmin,
        vmax=vmax,
    )
    print("Done.")
```

B.4. Frekvencia-tartományra vonatkozó textúrák generálása

```
# generate_texture_freq.py

from rssi_processing import *
import matplotlib as mpl
import itertools
from tqdm import tqdm


def create_textures_freq(
    files,
    spatial_res_deg,
    frequency_res_mhz,
    gap_threshold_mhz,
    los_distance_limit,
    icosahedral_mesh_refinement_level,
    points_per_deg,
    gauss_sigma_deg,
    vmin,
    vmax,
    cmap,
):
    points_mollweide = False

    # Colormap
    norm = mpl.colors.Normalize(vmin=vmin, vmax=vmax)
    colormap = plt.get_cmap(cmap)
    colors_df = pd.DataFrame(columns=["value", "r", "g", "b", "a"])
    for i in range(vmin, vmax + 1, 1):
        r, g, b, a = colormap(norm(i))
        colors_df = colors_df.append(
            pd.Series(data={"value": i, "r": r, "g": g, "b": b, "a": a})
,
            ignore_index=True,
        )
    colors_df.to_csv("colormap.csv", header=True, index=False)

    # Textúrák
    # Előszűrés
    mean_altitude, rssi_rounded = import_usable(
        files, spatial_res_deg=spatial_res_deg, frequency_res_mhz=
frequency_res_mhz
    )
    rssi_freq_binned = pd.DataFrame(columns=["long", "lat", "rssi", "
freq_bin"])

    # Minden kiválasztható kombinációra más textúra kell
    freq_bins = list(
        itertools.combinations(
            [0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000], 2
        )
    )
    for lower, upper in tqdm(freq_bins):
        rssi_filtered = rssi_rounded[
            (lower <= rssi_rounded["freq_mhz"]) & (rssi_rounded["
freq_mhz"] <= upper)
        ]
```

```
        if rssi_filtered.empty:
            print("{0}--{1}MHz empty".format(lower, upper))
            continue

        print("{0}--{1}MHz not empty".format(lower, upper))

        # RSSI-átlagolás
        rssi_averaged = average_rounded_rssi(
            rssi_filtered, gap_threshold_mhz=gap_threshold_mhz
        )

        rssi_averaged["freq_bin_lower"] = np.ones(len(rssi_averaged)) *
    lower
        rssi_averaged["freq_bin_upper"] = np.ones(len(rssi_averaged)) *
    upper
        rssi_freq_binned = rssi_freq_binned.append(rssi_averaged)

        # Interpoláció
        mesh = stripy.spherical_meshes.icosahedral_mesh(
            refinement_levels=icosahedral_mesh_refinement_level,
            include_face_points=True,
            tree=True,
        )

        lons_rad = np.radians(rssi_averaged["long"].values)
        lats_rad = np.radians(rssi_averaged["lat"].values)
        rssi = rssi_averaged["rssi"].values

        base_filename = "{0:03d}-{1:03d}MHz".format(lower, upper)

        # Selected points
        if points_mollweide is True:
            plot_mollweide(
                lons_rad,
                lats_rad,
                rssi,
                "mollweide_selected_" + base_filename + ".png",
                cmap="Reds",
                s=2,
                vmin=vmin,
                vmax=vmax,
            )
        # IDW
        rssi_idw_mesh = IDW_interpolate_rssi(
            lons_rad, lats_rad, rssi, mesh,
            los_distance_limit=los_distance_limit,
            mean_altitude=mean_altitude
        )

        xi, yi, zi = plate_carree_interpolation(
            mesh.lons,
            mesh.lats,
            rssi_idw_mesh,
            points_per_deg=points_per_deg,
            gauss_sigma_deg=gauss_sigma_deg,
        )
        plot_texture(
            zi,
            filename="textures/continous_" + base_filename + ".png",
```

```
            cmap=cmap,
            vmin=vmin,
            vmax=vmax,
        )

        levels = np.arange(-120, -40 + 1, 5)

        plot_contour_level_texture(
            xi,
            yi,
            zi,
            "textures/contour_labeled_" + base_filename + ".png",
            vmin=vmin,
            vmax=vmax,
            cmap=cmap,
            labels=True,
            levels=levels,
        )

        plot_contour_level_texture(
            xi,
            yi,
            zi,
            "textures/contour_" + base_filename + ".png",
            vmin=vmin,
            vmax=vmax,
            cmap=cmap,
            labels=False,
            levels=levels,
        )

        # Voronoi
        rssi_voronoi_mesh = voronoi_rssi(lons_rad, lats_rad, rssi, mesh)
        xi, yi, zi = plate_carree_interpolation(
            mesh.lons,
            mesh.lats,
            rssi_voronoi_mesh,
            points_per_deg=points_per_deg,
            gauss_sigma_deg=0.1,
        )

        plot_texture(
            zi,
            filename="textures/voronoi_" + base_filename + ".png",
            cmap=cmap,
            vmin=vmin,
            vmax=vmax,
        )
        print("{0}-{1}MHz done.".format(lower, upper - 1))

    # Értékek exportálása 3D-s térképhez
    rssi_freq_binned.rename(
        columns={"long": "X", "lat": "Y", "freq_bin": "Freq", "rssi": "
    Rssi"},
        inplace=True,
    )
    rssi_freq_binned["color"] = dB_to_hex_color(
        rssi_freq_binned["Rssi"].values, cmap=cmap, vmin=vmin, vmax=vmax
    )
```

```
168     rssi_freq_binned.to_json("rssi_binned.json", orient="records")
```

C. függelék

Weboldal

C.1. Weboldal forráskódja

```
<html>
  <head>
    <link rel="shortcut icon" href="#" />
    <meta charset=utf-8>
    <title>Smog-P adatvizualizáció</title>
    <link rel="stylesheet" href="css/multislider.css">
    <link rel="stylesheet" href="css/overlays.css">
    <style>
    </style>
  </head>

  <body>
    <div>
    <div id="loading_div">
      <div id="loading_text">Loading..</div>
    </div>

    <div id="overlay">
        <img id="legend_pic" src="textures/colorbar.png">
        <div id="text">
          <p>Spectrum data from Hungarian <a href="http://gnd.bme.hu/
   smog">SMOG-P nanosatellite</a>.</p>
          <p>Visualization and data processing: Markotics Boldizsár and
   Takács Donát, <a href="https://kozmosz.space/">BME Cosmos Society</a>
   .</p>
        </div>
    </div>

    <div id="jobbmenu">
      <div class="placeholder"style="margin-top: 10px;">Globe and Points
   </div>

      <div class='gomb'>
        <label>Globe Texture</label>
          <select id='menuGlobemap'>
            <option value='none'>None</option>
            <option value='satellite'>Satellite</option>
            <option value='coastlines_dark' selected="selected">Dark
   coastlines</option>
```

```
35            <option value='coastlines_light'>Light coastlines</option>
36          </select>
37      </div>
38
39      <div class="gomb" id='menuTogglePoints'>
40        Hide measuring points
41      </div>
42
43      <div class="placeholder"style="margin-top: 10px;">Smog map</div>
44
45      <div class='gomb'>
46        <label>Smog Map Texture</label>
47          <select id='menuSmogMap'>
48            <option value='none'>None</option>
49            <option value='continous' selected="selected">Continous</
    option>
50            <option value='contour'>Contour</option>
51            <option value='contour_labeled'>Labeled contour</option>
52            <option value='voronoi'>Voronoi</option>
53          </select>
54      </div>
55
56      <div class="gomb">
57        <label>Opacity</label>
58        <input type='range' id='menuOpacity'>
59      </div>
60
61      <div class="placeholder"style="margin-top: 10px;">Frequency filter
    </div>
62
63      <div class="gomb">
64        <label>Frequency range (MHz)</label>
65        <div slider id="slider-distance">
66          <div>
67            <div inverse-left style="width:70%;"></div>
68            <div inverse-right style="width:70%;"></div>
69            <div range style="left:30%;right:40%;"></div>
70            <span thumb style="left:30%;"></span>
71            <span thumb style="left:60%;"></span>
72            <div sign style="left:30%;">
73              <span id="value">300</span>
74            </div>
75            <div sign style="left:60%;">
76              <span id="value">600</span>
77            </div>
78          </div>
79          <input id='menuLowFreqSlider' type="range" tabindex="0" value=
    "30" max="100" min="0" step="1" oninput="
80          this.value=Math.min(this.value,this.parentNode.childNodes[5].
    value-1);
81          var value=(100/(parseInt(this.max)-parseInt(this.min)))*
    parseInt(this.value)-(100/(parseInt(this.max)-parseInt(this.min)))*
    parseInt(this.min);
82          var children = this.parentNode.childNodes[1].childNodes;
83          children[1].style.width=value+'%';
84          children[5].style.left=value+'%';
85          children[7].style.left=value+'%';children[11].style.left=value
    +'%';
86          children[11].childNodes[1].innerHTML=this.value*10;" />
```

```
87
88          <input id='menuHighFreqSlider' type="range" tabindex="0" value
    ="60" max="100" min="0" step="1" oninput="
89          this.value=Math.max(this.value,this.parentNode.childNodes[3].
    value-(-1));
90          var value=(100/(parseInt(this.max)-parseInt(this.min)))*
    parseInt(this.value)-(100/(parseInt(this.max)-parseInt(this.min)))*
    parseInt(this.min);
91          var children = this.parentNode.childNodes[1].childNodes;
92          children[3].style.width=(100-value)+'%';
93          children[5].style.right=(100-value)+'%';
94          children[9].style.left=value+'%';children[13].style.left=value
    +'%';
95          children[13].childNodes[1].innerHTML=this.value*10;" />
96        </div>
97      </div>
98
99      <div class="placeholder"style="margin-top: 10px;">Movement</div>
100
101     <div class="gomb" id="menuRotation">
102       Stop rotation
103     </div>
104   </div>
105
106
107   <canvas id='globecanvas'></canvas>
108   <div id="labels"></div>
109     <script type="module">
110       var db_json;
111       function readTextFile(file,callback)
112       {
113         var rawFile = new XMLHttpRequest();
114         rawFile.overrideMimeType("application/json");
115         rawFile.open("GET",file, false);
116         rawFile.onreadystatechange = function(){
117           if(rawFile.readyState === 4 && rawFile.status == "200"){
118             callback(rawFile.responseText);
119           }
120         }
121         rawFile.send(null);
122       }
123
124       readTextFile("./db.json",function(text){
125         var tacsi = JSON.parse(text);
126         db_json =tacsi;
127       });
128
129       import * as THREE from './node_modules/three/build/three.module.
    js';
130       import { OrbitControls } from './node_modules/three/examples/jsm
    /controls/OrbitControls.js';
131
132       //Föld sugara
133       const EarthRadius = 300;
134       //műhold keringésének átlagos sugara
135       const SateliteRadius = 320;
136       //mérőpont sugara
137       const MeasPointRadius = 2;
138       //kamera magasság
```

```
139         const CameraHeight = 800;
140         //Földgömb felbontás
141         const EarthRes = 40;
142         //mérőpont felbontás
143         const MeasPointRes = 5;
144         //mérőpont átlátszóság
145         const golyo_opacity = 1;
146         //smog átlátszóság
147         const smog_opacity=0.6;
148 
149         var ObjGroup = new THREE.Group();
150         var golyotomb = new THREE.Group();
151         var Earth = new THREE.Layers();
152         Earth = 1;
153         var SmogMap = new THREE.Layers();
154         SmogMap = 2;
155         var SmogPoint = new THREE.Layers();
156         SmogPoint = 3;
157 
158         var scene, renderer, foldgeom,foldtexture,foldmaterial,foldglobe
    ,smoggeom,smogtexture,smogmaterial,smogglobe;
159         let camera = new THREE.PerspectiveCamera( 75, window.innerWidth/
    window.innerHeight, 0.1, 1000 );
160         var SmogPointArray = [];
161 
162         //menü állapot változók
163         var Forgas=true;
164         var PointsOn=true;
165         var minFreq=300;
166         var maxFreq=600;
167         var SmogMap = document.getElementById("menuSmogMap").value;;
168         var GlobeMap;
169         var DBsorted=editdb();
170 
171         //Eventek
172         document.getElementById("menuOpacity").addEventListener("input",
     OpacityEvent);
173         document.getElementById("menuRotation").addEventListener("click"
    , MenuRotationEvent);
174         document.getElementById("menuSmogMap").addEventListener("change"
    , SmogMapEvent);
175         document.getElementById("menuGlobemap").addEventListener("change
    ", GlobeMapEvent);
176         document.getElementById("menuTogglePoints").addEventListener("
    click", MenuTogglePointsEvent);
177         document.getElementById("menuLowFreqSlider").addEventListener("
    change", LowFreqEvent);
178         document.getElementById("menuHighFreqSlider").addEventListener("
    change", HighFreqEvent);
179 
180         function LowFreqEvent()
181         {
182           minFreq=document.getElementById("menuLowFreqSlider").value*10;
183           DBsorted=editdb();
184           NewDataToGlobe();
185           SmogMapEvent();
186           return minFreq;
187         }
188 
```

```
189         function HighFreqEvent()
190         {
191           maxFreq=document.getElementById("menuHighFreqSlider").value
    *10;
192           DBsorted=editdb();
193           NewDataToGlobe();
194           SmogMapEvent();
195           return maxFreq;
196         }
197 
198         function OpacityEvent()
199         {
200           smogglobe.material.opacity=(document.getElementById("
    menuOpacity").value)/100;
201         }
202 
203         function MenuRotationEvent()
204         {
205           if(!Forgas) document.getElementById("menuRotation").innerHTML
    ='Stop rotation';
206           else document.getElementById("menuRotation").innerHTML='Start
    rotation';
207           Forgas=!Forgas;
208         }
209 
210         function SmogMapEvent()
211         {
212           SmogMap=document.getElementById("menuSmogMap").value;
213           if(SmogMap=='none')
214             {
215               camera.layers.disable(SmogMap);
216               smogglobe.material.opacity=0;
217             }
218           else
219           {
220             camera.layers.enable(SmogMap);
221             ChangeTexture(SmogMap, Math.round(minFreq/100)*100 ,Math.
    round(maxFreq/100)*100);
222             OpacityEvent();
223           }
224         }
225 
226         function GlobeMapEvent()
227         {
228           GlobeMap=document.getElementById("menuGlobemap").value
229           if(GlobeMap=='none')
230           {
231             camera.layers.disable(Earth);
232           }
233           else
234           {
235             camera.layers.enable(Earth);
236             ChangeTexture_world(GlobeMap);
237           }
238         }
239 
240         function MenuTogglePointsEvent()
241         {
242           if(!PointsOn) {
```

```
243             document.getElementById("menuTogglePoints").innerHTML='Show
    measuring points';
244             camera.layers.enable(SmogPoint);
245           }
246           else{
247             document.getElementById("menuTogglePoints").innerHTML='Hide
    measuring points';
248             camera.layers.disable(SmogPoint);
249           }
250           PointsOn=!PointsOn;
251         }
252
253         init();
254         NewDataToGlobe();
255         animate();
256
257         function editdb()
258         {
259           var sorteddb = new Array();
260           for(var i=0; i<db_json.length;i++){
261               if(db_json[i].freq_bin_lower == Math.round(minFreq/100)
    *100 && db_json[i].freq_bin_upper == Math.round(maxFreq/100)*100){
262                 sorteddb.push(db_json[i]);
263               }
264             }
265           return sorteddb;
266         }
267
268         function SiteLoaded(){
269           document.getElementById("loading_div").style.display = "none";
270         }
271
272         function ChangeTexture_world(newTexture)
273         {
274           ObjGroup.remove(foldglobe);
275
276           foldtexture =new THREE.TextureLoader().load('textures/'+
    newTexture+'.png');
277           foldmaterial = new THREE.MeshBasicMaterial( {map: foldtexture
    });
278
279           foldglobe = new THREE.Mesh( foldgeom, foldmaterial);
280           foldglobe.layers.set(Earth);
281           ObjGroup.add( foldglobe );
282           scene.add(ObjGroup);
283           render();
284         }
285
286         function ChangeTexture(newTexture, min_freq=300, max_freq=600)
287         {
288           ObjGroup.remove(smogglobe);
289           if(min_freq==0)min_freq='000';
290           smogtexture =new THREE.TextureLoader().load('textures/
    smog_texture/'+newTexture+'_'+min_freq+'-'+max_freq+'MHz'+'.png');
291           smogmaterial = new THREE.MeshBasicMaterial( {map: smogtexture,
     opacity: smog_opacity, transparent: true});
292           smogglobe = new THREE.Mesh( smoggeom, smogmaterial);
293           smogglobe.layers.set(SmogMap);
294           ObjGroup.add( smogglobe );
```

```
295           scene.add(ObjGroup);
296           render();
297         }
298
299         function NewDataToGlobe()
300         {
301           if(SmogPointArray!= null)
302           {
303             for(var i=0;i<SmogPointArray.length;i++)
304             {
305               ObjGroup.remove( SmogPointArray[i]);
306               SmogPointArray[i].geometry.dispose();
307               SmogPointArray[i].material.dispose();
308             }
309             SmogPointArray=[];
310           }
311
312           for(var i=0;i<DBsorted.length;i++)
313           {
314             var aktcolor =0xffffff;
315             aktcolor=DBsorted[i].color;
316             var hexcol = new THREE.Color(parseInt(aktcolor, 16));
317             var tombgolygeom = new THREE.SphereGeometry(MeasPointRadius,
    MeasPointRes, MeasPointRes);
318             var tombgolyotexture = new THREE.MeshBasicMaterial( {color:
   hexcol } );
319             var tombgolyo =new THREE.Mesh(tombgolygeom,tombgolyotexture)
   ;
320             tombgolyo.material.transparent= true;
321             tombgolyo.material.opacity=golyo_opacity;
322
323             //Szög Setup
324             var lng= DBsorted[i].Y;
325             var lat= DBsorted[i].X;
326
327             var phi   = (90-lat)*(Math.PI/180)+Math.PI/2;
328             var theta = (lng+180)*(Math.PI/180);
329
330             var r = SateliteRadius+5;
331             tombgolyo.position.x = r * Math.cos(theta) * Math.cos(phi);
332             tombgolyo.position.y = -r * Math.sin(theta);
333             tombgolyo.position.z = r * Math.cos(theta) * Math.sin(phi);
334
335             SmogPointArray.push(tombgolyo);
336           }
337           for(var i=0; i<SmogPointArray.length;i++)
338           {
339             ObjGroup.add( SmogPointArray[i] );
340             SmogPointArray[i].layers.set(SmogPoint);
341           }
342         }
343
344         function init()
345         {
346           scene = new THREE.Scene();
347
348           //Kezdőállapotok
349           camera.layers.enable(Earth);
350           camera.layers.enable(SmogMap);
```

```
351         renderer = new THREE.WebGLRenderer({canvas: globecanvas, alpha
    : true , antialias: true});
352         renderer.setSize( window.innerWidth, window.innerHeight );
353         document.body.appendChild( renderer.domElement );
354 
355         //Orbit
356         var controls =  new OrbitControls( camera, renderer.domElement
     );
357         controls.enablePan= false;
358         controls.enableDamping = true;
359         controls.dampingFactor = 0.1;
360         controls.rotateSpeed = 0.3;
361         controls.enableZoom = true;
362         controls.zoomSpeed = 0.1;
363         controls.minDistance = SateliteRadius*1.2;
364         controls.maxDistance = CameraHeight;
365         controls.maxPolarAngle = Math.PI ;
366 
367         //Földgömb
368         foldgeom = new THREE.SphereGeometry(EarthRadius, EarthRes,
    EarthRes);
369         foldtexture =new THREE.TextureLoader().load('textures/
    coastlines_dark.png');
370         foldmaterial = new THREE.MeshBasicMaterial( {map: foldtexture
    });
371         foldglobe = new THREE.Mesh( foldgeom, foldmaterial);
372         foldglobe.name="foldgomb";
373         foldglobe.layers.set(Earth);
374         ObjGroup.add( foldglobe );
375 
376         //Elektrosmog
377         smoggeom = new THREE.SphereGeometry(SateliteRadius, EarthRes,
    EarthRes);
378         smogtexture =new THREE.TextureLoader().load('textures/
    smog_texture/continous_300-600MHz.png');
379         smogmaterial = new THREE.MeshBasicMaterial( {map: smogtexture,
     opacity: smog_opacity, transparent: true});
380         smogglobe = new THREE.Mesh( smoggeom, smogmaterial );
381         smogglobe.layers.set(SmogMap);
382         camera.layers.enable(SmogPoint);
383         ObjGroup.add( smogglobe );
384 
385         //Scene
386         scene.add(ObjGroup);
387         camera.position.z = CameraHeight;
388 
389         window.addEventListener( 'resize', onWindowResize, false );
390         SiteLoaded();
391       }
392 
393       function animate()
394       {
395         requestAnimationFrame( animate );
396         render();
397       };
398 
399       function render()
400       {
401         if(Forgas)
```

```
            ObjGroup.rotation.y += 0.001;
          renderer.render( scene, camera );
        }

        function onWindowResize()
        {
          camera.aspect = window.innerWidth / window.innerHeight;
          camera.updateProjectionMatrix();
          renderer.setSize( window.innerWidth, window.innerHeight );
        }

        document.getElementById("loading_div").style.display = "none";
      </script>
    </div>
  </body>

</html>
```

C.2. Weboldalhoz tartozó stíluslapok

```
[slider] {
    position: relative;
    height: 8px;
    border-radius: 8px;
    text-align: left;
    margin: 45px 0 10px 0;
  }

  [slider] > div {
    position: absolute;
    left: 13px;
    right: 15px;
    height: 8px;
  }

  [slider] > div > [inverse-left] {
    position: absolute;
    left: 0;
    height: 8px;
    border-radius: 8px;
    background-color: #CCC;
    margin: 0 7px;
  }

  [slider] > div > [inverse-right] {
    position: absolute;
    right: 0;
    height: 8px;
    border-radius: 8px;
    background-color: #CCC;
    margin: 0 7px;
  }

  [slider] > div > [range] {
    position: absolute;
    left: 0;
    height: 8px;
    border-radius: 8px;
    background-color: #0551ff;
  }
  [slider] > div > [range]:hover {
    background-color: #0041cb;
  }

  [slider] > div > [thumb] {
    position: absolute;
    top: -3px;
    z-index: 3;
    height: 14px;
    width: 14px;
    text-align: left;
    margin-left: -11px;
    cursor: pointer;
    box-shadow: 0 3px 8px rgba(0, 0, 0, 0.4);
    background-color: #FFF;
    border-radius: 50%;
    outline: none;
  }
```

```
59
60  [slider] > input[type=range] {
61    position: absolute;
62    pointer-events: none;
63    -webkit-appearance: none;
64    z-index: 4;
65    height: 8px;
66    top: -2px;
67    width: 100%;
68    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=0)";
69    filter: alpha(opacity=0);
70    -moz-opacity: 0;
71    -khtml-opacity: 0;
72    opacity: 0;
73  }
74
75  div[slider] > input[type=range]::-ms-track {
76    -webkit-appearance: none;
77    background: transparent;
78    color: transparent;
79  }
80
81  div[slider] > input[type=range]::-moz-range-track {
82    -moz-appearance: none;
83    background: transparent;
84    color: transparent;
85  }
86
87  div[slider] > input[type=range]:focus::-webkit-slider-runnable-track {
88    background: transparent;
89    border: transparent;
90  }
91
92  div[slider] > input[type=range]:focus {
93    outline: none;
94  }
95
96  div[slider] > input[type=range]::-ms-thumb {
97    pointer-events: all;
98    width: 28px;
99    height: 28px;
100   border-radius: 0px;
101   border: 0 none;
102   background: red;
103 }
104
105 div[slider] > input[type=range]::-moz-range-thumb {
106   pointer-events: all;
107   width: 28px;
108   height: 28px;
109   border-radius: 0px;
110   border: 0 none;
111   background: red;
112 }
113
114 div[slider] > input[type=range]::-webkit-slider-thumb {
115   pointer-events: all;
116   width: 28px;
117   height: 28px;
```

```
    border-radius: 0px;
    border: 0 none;
    background: red;
    -webkit-appearance: none;
  }

  div[slider] > input[type=range]::-ms-fill-lower {
    background: transparent;
    border: 0 none;
  }

  div[slider] > input[type=range]::-ms-fill-upper {
    background: transparent;
    border: 0 none;
  }

  div[slider] > input[type=range]::-ms-tooltip {
    display: none;
  }

  [slider] > div > [sign] {
    position: absolute;
    margin-left: -18px;
    top: -39px;
    z-index:4;
    background-color: #0551ff;
    color: #fff;
    width: 28px;
    height: 28px;
    border-radius: 28px;
    -webkit-border-radius: 28px;
    align-items: center;
    -webkit-justify-content: center;
    justify-content: center;
    text-align: center;
  }

  [slider] > div > [sign]:after {
    position: absolute;
    content: '';
    left: 0;
    border-radius: 16px;
    top: 19px;
    border-left: 14px solid transparent;
    border-right: 14px solid transparent;
    border-top-width: 16px;
    border-top-style: solid;
    border-top-color: #0551ff;
  }

  [slider] > div > [sign] > span {
    font-size: 12px;
    font-weight: 700;
    line-height: 28px;
  }
```

```
html{
  font: 11px 'Lucida Grande', sans-serif;
  font-style: normal;
  font-variant-ligatures: normal;
  font-variant-caps: normal;
  font-variant-numeric: normal;
  font-variant-east-asian: normal;
  font-weight: normal;
  font-stretch: normal;
  font-size: 11px;
  line-height: normal;
  font-family: "Lucida Grande", sans-serif;
  color: white;
}

body{
  margin: 0;
  background-color: black;
  background-size: cover;
  overflow-x: hidden;
}

#globecanvas{
  display: block;
}

#renderdiv{
  position: relative;
  width: 100vw;
  height: 100vh;
  overflow: hidden;
}

#labels{
  position: absolute;
  left: 0;
  top: 0;
  color: white;
}

#labels>div{
  position: absolute;
  left: 0;
  top: 0;
  cursor: pointer;
  font-size: large;
  user-select: none;
  text-shadow:
    -1px -1px 0 #000,
      0   -1px 0 #000,
      1px -1px 0 #000,
      1px  0   0 #000,
      1px  1px 0 #000,
      0    1px 0 #000,
    -1px  1px 0 #000,
    -1px  0   0 #000;
}
#labels>div:hover{
  color: red;
```

```
}

#overlay{
  position: fixed;
  display: block;
  width: 250px;
  height: auto;
  left: 10px;
  bottom: 10px;
  background-color: rgba(100,100,100,0.5);
  z-index: 2;
  border-radius: 10px;
}

#text{
  padding: 10px;
}

#legend_pic{
  display: block;
  padding: 10px;
  padding-left: 40px;
  padding-bottom: 0;
  max-width: 100%;
  height: auto;
}

a{
  color:lightgray;
}

#loading_text{
  position: absolute;
  top: 50%;
  left: 50%;
  font-size: 50px;
  color: white;
  transform: translate(-50%,-50%);
  -ms-transform: translate(-50%,-50%);
  font-family: "Lucida Grande", sans-serif;
}

#loading_div{
  position: fixed;
  display: block;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0,0,0,0.5);
  z-index: 3;
  cursor: pointer;
}

#jobbmenu{
  position: fixed;
  display: block;
```

```
  width: 250px;
  height: auto;
  right: 10px;
  top: 10px;
  background-color: rgba(100,100,100,0.25);
  z-index: 3;
  border-radius: 10px;
}


.gomb{
  width: 220px;
  border: none;
  outline: none;
  cursor: pointer;
  padding: 10px;
  margin: 5px;
  border-radius: 5px;
  right:0px;
  background-color: rgba(100,100,100,0.5);
}

.gomb:hover{
  background-color: rgba(100,100,100,0.7);
}

.gomb select{
  border: none;
  cursor: pointer;
  color: white;
  background: inherit;
  font: inherit;
}

.placeholder{
  width: 238px;
  background-color: black;
  padding: 5px;
  margin: 1px;
}
label{
  width:115px;
}
```