

TDK DOLGOZAT

Pieler Gergely



Nyugat-magyarországi Egyetem

Simonyi Károly Műszaki, Faanyagtudományi és
Művészeti Kar

Informatikai és Gazdasági Intézet

Ionogram automatikus feldolgozását és kiértékelését támogató szoftver implementálása

Szerző: Pieler Gergely, 2. évfolyamos MSc hallgató, pielerg@gain.nyme.hu

Konzulens: Bencsik Gergely, PhD hallgató, bencsikg@inf.nyme.hu

2014. november. 3

TARTALOMJEGYZÉK

Tartalomjegyzék.....	3
1 Bevezetés.....	4
1.1 Probléma bemutatása.....	4
2 Ionoszféra és annak vizsgálata	5
2.1 Ionogram és a kiértékelés menete.....	8
3 Ionogram elemző rendszer bemutatása és implementálása.....	10
3.1 Qt keretrendszer	10
3.2 Alapkoncepció.....	11
3.3 Tervezés és implementáció	12
3.4 Connected-Component Labeling algoritmus, mint szűrés.....	15
3.5 Unió-Holvan adatszerkezet megvalósítása union-by-rank és útössze-nyomás heurisztikával	17
4 Eredmények.....	20
4.1 Függvényillesztéssel megvalósítható távolságalapú klaszterező módszer bemutatása	25
5 Összefoglalás	26
5.1 Tovább lépési lehetőségek.....	27
6 Irodalomjegyzék.....	28

1 BEVEZETÉS

A geofizika által vizsgált jelenségek nagy része nem csak a Föld belsejére és felszínére irányulnak, hanem a magnetoszféra határáig terjednek. Földünk légkörének határa is eddig húzódik, mely elemzésével és megfigyelésével foglalkozik a geofizika és a meteorológia. Az atmoszféra több különböző fizikai tulajdonság szerint rétegekre bontható. Az egyik ilyen az ionoszféra, mely az atmoszféra azon tartományát jelöli, ahol a szabad elektronok és ionok olyan magas koncentrációban vannak jelen, hogy azok a rádióhullámok terjedését jelentősen befolyásolják. Ennek vizsgálatára és mérésére egy speciális radart használnak, melyet ionoszondának hívnak. Ezek az ionoszféra rétegeit szondázzák magas hullámú rádiójelekkel és a különböző magasságokból visszatérő jelek kibocsájtási és beérkezési idődifferenciálját dolgozzák fel. Az ekkor létrejött szöveges adatok feldolgozásából hozható létre az ionogram, amely egy olyan bináris képet eredményez, ami a visszaverődési helyek látszólagos magasságát jelöli (Y tengely), az adott frekvencia függvényében (X tengely). Ezeket az adatokat az ionoszféra tulajdonságainak és változásainak mérésére, valamint ezen változások és az űridőjárás közti összefüggések megállapítására használják. Továbbá a rádiózáshoz szükséges optimális működési frekvencia meghatározására is használják, mely által sokkal nagyobb távolságokat lehet áthidalni[1].

A projektfeladataink során, szoros kapcsolatban álltunk a Magyar Tudományos Akadémia soproni Geodéziai és Geofizikai Intézetével, ahol betekintést nyertünk az ionoszondák által végzett mérésekbe, valamint azok mérési adataiba az ionogramokba. Ekkor merült fel egy igény az intézet részéről, hogy egy olyan szoftvert készítsünk, amely képes ezek utólagos vagy akár azonnali, valós idejű feldolgozására. Ezen szoftver tervezését, fejlesztését és eddig elért eredményeit szeretnénk részletesebben bemutatni TDK dolgozatunkban.

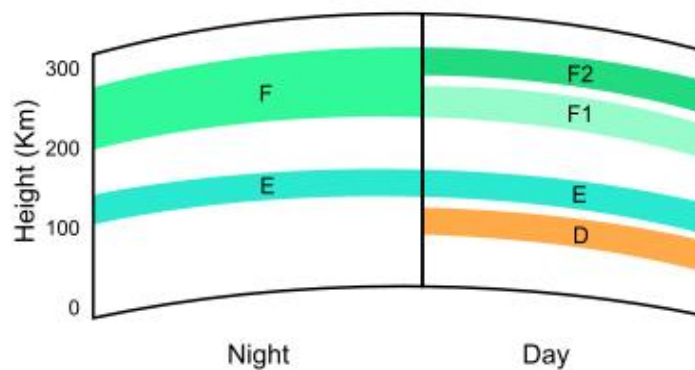
1.1 PROBLÉMA BEMUTATÁSA

A tudományos áttekintés előtt röviden bemutatnánk a problémakört. Az ionogram releváns területeinek meghatározása és elemzése egy jól ismert és fontos problématerület az ionoszférikus kutatásoknak. A feladat nehézsége, hogy a képek kb. 70%-a zaj, így ezek kiszűrése egy sarkalatos pontja az elemzéseknek. Másodrészből az ionoszférában végbemenő véletlen fizikai behatások végett az ionogramok teljesen elérőek lehetnek az elvárttól. Ennek hátránya, hogy az általános esetre elkészített modell nem illeszthető, transzformálható más ionogramra. Továbbá a nappali és éjszakai ionogramok is más felépítésűek a megváltozott szabad elektron és ion koncentrációból adódóan, mely még egy könnyebben kezelhető probléma. Ezek azok az okok, ami miatt az ionogramok automatikus kiértékelését nem lehet teljes mértékben szoftveres feldolgozásra bízni. A mai napig a bevált eljárás, hogy az automatikusan kiértékelte ionogram egy manuális ellenőrzésen is átesik egy hozzáértő személy jelenlétében.

2 IONOSZFÉRA ÉS ANNAK VIZSGÁLATA

Az ionoszférikus rétegek ionizált gázokból jönnek létre, melyeket a napsugárzás és más behatások ionizálnak. E folyamat során az elektromosan semleges atomok vagy molekulák pozitív- vagy negatív töltésűvé ionná változnak, attól függően, hogy elektront veszítenek vagy vesznek fel. Ezután ezek az atomok vagy molekulák ionizált állapotban maradnak, majd idővel más ionnal vagy elektronnal reakcióba lépve rekombinálódnak.

A különböző elektron- és ionkoncentráció miatt az ionoszféra több elkülöníthető rétegre bontható. Ezek a rétegek a különböző sűrűségük végett más és más frekvenciájú rádióhullámot képesek visszaverni vagy csillapítani. Ez alapján az ionoszféra a 1. ábrán látható módon osztható fel rétegekre.



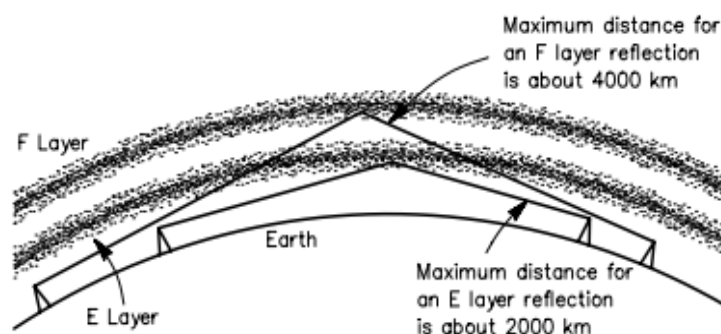
1. ábra - Rétegek előfordulása a napszak függvényében

Az ionoszféra első tartománya az úgynevezett D réteg, mely 50-90 km között helyezkedik el. Itt az ionizáció mértéke a legalacsonyabb a rétegek közül. A folyamat legfőbb kiváltója az UV-sugárzás, de nem elhanyagolható a kozmikus sugárzás hatása sem. Ezeknek köszönhetően a D réteg általában csak nappal létezik és estére feloszlik, de a kozmikus sugárzás hatására este is létrejöhét jelentősebb ionizáció. Ezzel szemben a rekombinációs folyamatok gyorsan zajlanak, mivel a semleges atomok sűrűsége magas ebben a tartományban. A réteg egyik meghatározó tulajdonsága a szabad elektronok magas koncentrációja, mely befolyásolja a rádióhullámok terjedését. A rétegről a magas frekvenciájú rádióhullámok nem verődnek vissza, de nagymértékű energia veszteséget szenvednek, amint áthaladnak rajta. A rétegről való visszaverődés felső határa a 200 kHz körül van, majd egészen 2 MHz-ig erősen elnyeli a rádióhullámokat, a 3 MHz felettiek pedig könnyedén áthatolnak rajta. [2] Ezen tulajdonság végett a D réteg nagy szerepet játszik a rövidhullám rádiós összeköttetésekénél.

Egyre feljebb haladva elérünk az E rétegig, mely a Kennelly-Heaviside nevet viselte. Ez körülbelül a 90 és 150 kilométeres magasságok között helyezkedik el. Az ionizáció a lágy röntgensugarak és az UV-sugárzás által megy végbe, mely nagyrészt az oxigén molekulákat

ionizálja. A rekombináció sebessége itt jóval kisebb, mivel a semleges atomok sűrűség nem olyan nagy, viszont a tartomány elektron sűrűsége kb. 100-szorosa a D rétegnek. Ennek köszönhetően az 1 és 2,5 MHz közötti rádióhullámokat könnyedén visszaveri, de az ennél magasabb frekvenciájúakat csak csillapítja. A napsugárzás hiánya következtében a réteg az éjszaka folyamán meggyengül, illetve feljebb tolódik. Ezen felül az E réteg magasságában van az ionizációnak egy nem periodikus változása is. Ekkor egy részleges ionizációs forrás által megnő az ionkoncentráció, s a környezeténél nagyobb elektronsűrűségű felhőket hoz létre, ezt nevezzük szporadikus E rétegnek. Ezen felhők kiterjedése lehet néhány négyzetkilométer, de akár több millió is. A réteg pár perctől, akár óráig is fennmaradhat. A részleges ionizációs forrás, mely ezeket létrehozza még nem tisztázott. Egyes elméletek szerint a semleges atmoszféra mozgása, az északi fény vagy a légkörbe belépő meteorok okozzák. Hátránya, hogy ezek a felhők könnyen visszaverik a nagyfrekvenciás rádióhullámokat akár 200 MHz-ig is.

A következő ionoszféra régió legjelentősebb része 150 és 400 km közé tehető, de sűrűségének folyamatos csökkenése elhúzódik egészen 1000 kilométeres magasságig. Ez az egyik állandóan megtalálható réteg az ionoszférában, melyet F rétegnek vagy Appleton rétegnek hívunk. Az esti órákban a régióban lecsökken az ionizáció, de nem olyan drasztikusan, mint a D és E rétegbe. Ennek legfőbb oka, hogy ekkora magasságba már jóval kisebb a rekombináció sebessége. Este a réteg 300-500 km közé húzódik, viszont nappal a megemelkedett UV-sugárzás és az általa létrehozott ionizáció hatására két jól elkülöníthető rétegre esik szét. Az alsó F1 rétegre (150-250 km) és a felső F2 rétegre (350-400 km). Az F1 hasonló az E réteghez, melyet nagyban befolyásol a napsugárzás, így éjszaka eltűnik és beleolvad az F2-be. A réteg egyik fő tulajdonsága, hogy itt van a legmagasabb szabad elektron- és ionkoncentráció. E tulajdonság legfőbb előnye, hogy könnyedén visszaveri a magas frekvenciájú rádióhullámokat kb. 10 MHz-ig és részlegesen elnyeli az e fölötteket. Ez fontos szerepet játszik a magas frekvenciájú rádiójelek továbbításában, mivel a réteget kihasználva, akár 4000 kilométeres távolságot is át lehet hidalni, míg az E réteggel 2000 kilométert. [3]



2. ábra - E és F réteg visszaverődésével áthidalható távolságok[3]

A rétegek némely esetben élesen elkülöníthetők egymástól, de sok esetben közbenső rétegek jöhetnek létre, mint például az említett szporadikus E-réteg, valamint ezen kívül még sok más és más zavar is létrejöhethet a különböző behatásoknak köszönhetően. Ezek a behatások lehetnek periodikusak, mint például a Napsugárzás, mely nem egyenletesen éri a Földet, illetve intenzitása és összetétele is más lehet. Valamint lehetnek aperiodikusak is, mint például a Napkitörések, melyek nagyban megváltoztathatják az alsóbb rétege összetételét és viselkedését.

Mint azt már a bevezetésben is említettük az ionoszféra vizsgálatára egy speciális radart, az ionoszondát alkalmazzák. Ez gyakorlatilag két szimmetrikus adó és vevő antennából álló berendezés. Az adó egy magas frekvenciás jeladó, mely széles frekvencia tartományban állítható, akár 0,5-23 MHz vagy 1-40 MHz között is. Mérés során ez végigpásztázza a meghatározott frekvenciatartományt 25 Hz-es léptékben és minden kiküldött frekvenciájú jel elküldése és beérkezése közt eltelt időt méri, mellyel meghatározható a visszaverődés látszólagos magassága. A kibocsájtott jelek, hogy jól elkülöníthetők legyenek rövid időtartamúak és adáskor a vevőoldal kikapcsolt állapotban van.

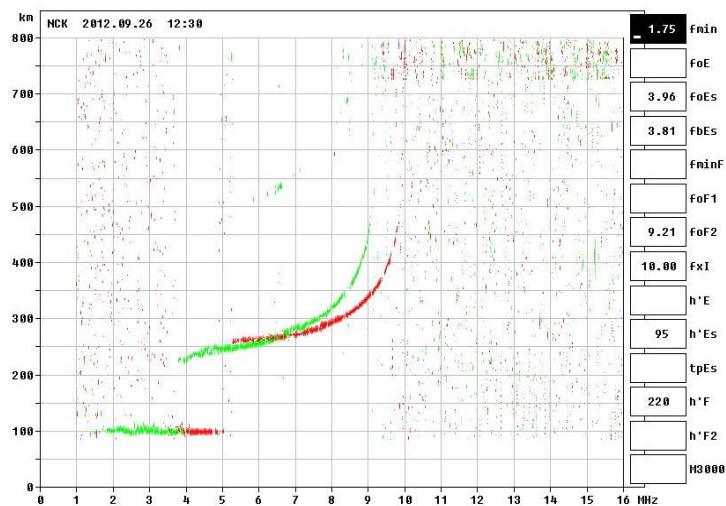
Esetünkben a mérések kimenete egy szöveges fájl, mely soronként tartalmazza a különböző pásztázott frekvenciákhoz tartozó jelvisszaérkezéseket. Egy mérési sor mindig az adott frekvencia értékével kezdődik, majd ezt követi szóközzel elválasztva a kibocsájtott jelek beérkezései. Az egyes beérkezések 4 különböző értéket vehetnek fel, attól függően, hogy milyen információt tartalmaznak. Ezek a következők lehetnek:

- (0) Nem érkezett be adat
- (1) Ordinary komponens
- (2) Extraordinary komponens
- (3) Zaj

Az ordinary és extraordinary komponensek közti különbség lényegében az ionoszféra fizikájából adódik, amit most részletesebben nem fejtenénk ki. A mérés során létrejött szöveges adatok feldolgozásából hozható létre az ionogram. Ez egy olyan bináris képet eredményez, ami a visszaverődési helyek látszólagos magasságát jelöli (Y tengely), az adott frekvencia függvényében (X tengely). Az ionogram részletesebb bemutatására és annak elemzésére a következő fejezetben szeretnénk kitérni.

2.1 IONOGRAM ÉS A KIÉRTÉKELÉS MENETE

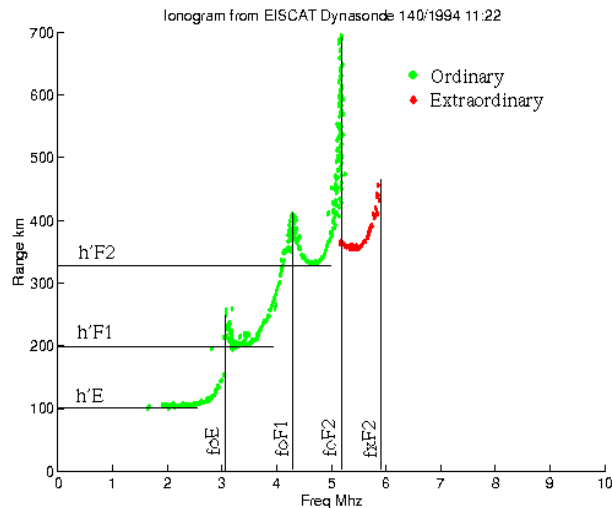
Mint azt már az előzőekben tárgyaltuk a mérési adatainkat ábrázolva, kapjuk meg a tényleges ionogramot. Ez tulajdonképpen egy pontfelhőkből álló kép, melyen különböző kisebb zaj jellegű és nagyobb elemzéshez szükséges foltok láthatók, melyek az egyes rétegekről való visszatéréseket ábrázolják. Egy mérési adatokból létrehozott és már kiértékelt ionogramot láthatunk a 3. ábrán.



3. ábra - Bináris adatokból konvertált és kiértékelt ionogram

Az előző fejezetben említett ordinary (1-es érték bináris fájlban) és extraordinary komponens (2-es érték) itt most színekkel elkülönítve jelenik meg. A zöld jelöli az ordinary komponenst, míg a piros az extraordinary komponenst. A kép ebben az esetben már egy előszűrt verzió, így a kb. 100 km alatti adatok már eleve eldobásra kerültek a mérés feldolgozásakor. Látható, hogy a kép bal és jobb oldala (kb. 10 MHz-től) sok kisebb zaj jellegű foltot tartalmaz, melyek az elemzés szempontjából nem értelmezhetőek. A fennmaradó egybefüggő görbeszerű foltok azok, amik a kiértékeléshez szükségesek. Valamint a jobb oldalon látható értékek a manuális kiértékelés eredményei. A következőkben egy példán keresztül szeretnénk szemléltetni, hogy hogyan kell kiértékelni egy ionogramot, illetve melyek azok az értékek, amik szükségesek az elemzésekhez.

Egy ionogramon minden egyes ionoszférikus réteget reprezentál egy görbeszerű folt. Ezeket a görbét általában elválasztja egymástól egy aszimptotikus érték (ahova a görbe tart), mely egyben a réteg kritikus frekvenciáját is jelöli. A 4. ábrán jól látható, hogy az egyes görbék mely pontjai szükségesek számunkra.



4. ábra - Ionogram jelölt értékekkel [4]

Az Ionogram jelölt értékekkel című ábra bal oldali első görbe az E réteget jelölni. Látható, hogy a visszavert rádióhullámok a 1,5 MHz és 3 MHz között helyezkednek el, melyek megfelelnek a réteg reflexiós tulajdonságának. A függvény aszimptotikus értéke fogja meghatározni a kritikus frekvenciát, ahonnan már nem verődik vissza rádióhullám az adott rétegről. Továbbhaladva a függvény minimuma fogja jelölni a réteg elhelyezkedésének magasságát. A következő görbe az F réteghez tartozik, amely nappali időszakban az ionizáció hatására két részre oszlik, ami két részre bontja a függvényt is. Ebben az esetben egy kicsit bonyolultabb az értékek meghatározása, mivel ezek nem különülnek el egymástól. A függvény első szakasza fogja jelölni az F1 réteget. Az itt felívelő szakasz aszimptotikus értéke, ahol a függvénynek töréspontja van, fogja megadni a kritikus frekvenciát, valamint ennek minimumpontja határozza meg a réteg elhelyezkedésének magasságát. A törésponttól továbbhaladva a függvény ezen szakasza fogja jelölni az F2 réteget, amellyel hasonlóképpen kell eljárni, mint az előzőekben. A grafikon végén látható piros görbe az extraordinary komponenshez tartozik, melynél a kritikus frekvenciát szokták csak meghatározni, de erre most nem térnénk ki részletesen. Ezen értékeken túl általában még a legalacsonyabb frekvenciát is le szokták olvasni, ami az első olyan frekvencia, amelyen visszaverődik a jel az E rétegről. Bár ebben a példában elég egyértelmű az értékek meghatározása, sok olyan eset van, ahol nehéz szabályszerűségeket felírni az ionogram értékeinek leolvasására, így elég nehéz az automatizálás megoldása minden esetre. A világon sok szervezet foglalkozik kutatásokkal és projektekkel ionogramok automatizált kiértékelésére. Az egyik ilyen az olaszországi Rómában lévő Istituto Nazionale di Geofisica e Vulcanologia nevű intézmény. Az általuk fejlesztett AutoScala nevű szoftver jó pár tudományos cikkel rendelkezik, melyek megtekinthetők a tartalomjegyzék [5] bejegyzésében szereplő webcím alatt. Ez képes elektron sűrűséget számolni, felismeri az Es, F1 és F2 rétegeket, valamint meghatározni azok értékeit egész nagy biztonsággal. Érdeemes szem előtt tartani, hogy a rendszer rendelkezik ezen

felül olyan funkcióval, ahol kézzel átállíthatók a különböző értékek, mivel a mai napig az ionogram kiértékelése csak hivatalos, ha azt egy hozzáértő ellenőrizte és jóváhagyta.

A megvizsgált ionogramok után a következő volt az alapkoncepciónk. Mivel az ionogramok kb. 70%-a zaj jellegű információ, így valamilyen képosztályozási algoritmus felhasználásával meg kell határozni a releváns és össze-függő görbeszerű területeket és elhagyni a zajokat. Ehhez egy olyan algoritmus szükséges, amely figyelembe veszi azt, hogy az egyes pixelek kapcsolatban állnak-e egymással vagy sem. Választásunk a bináris képfeldolgozásban jól ismert és könnyen alkalmazható Connected-Component Labeling (CCL) algoritmusra esett, mely lényegében a képen végighaladva felcímkézi a kép vizsgált összefüggő komponenseit. Ennek ismertetése a következő fejezetben megtekinthető.

3 IONOGRAM ELEMZŐ RENDSZER BEMUTATÁSA ÉS IMPLEMENTÁLÁSA

A rendszer egyik alapkövetelménye a platformfüggetlenség volt, mivel a Geodéziai és Geofizikai Intézetnél nem ritka a Linuxos környezet. Emiatt a választásunk a C++ programozási nyelvre esett. Ez a C nyelv továbbfejlesztett változata, amely biztosítja az objektumorientált programozási technika lehetőségét. A nyelv nagy előnye, hogy mindhárom platformon elérhető hozzá fordító program, így könnyedén hozható létre keresztplatformos alkalmazás. Szintaktikáját olyannyira megegyezik a C-vel, hogy egy C-ben megírt szintaktikailag helyes program egyben egy szintaktikailag helyes C++ program is.[6] Több népszerűbb fejlesztői környezetet vizsgálata után a választásunk a Qt keretrendszerre esett, melyet a következő alfejezetben érintőlegesen szeretnénk bemutatni.

3.1 QT KERETRENDSZER

A Qt tulajdonképpen egy olyan keretrendszer, mely egy saját osztálykönyvtárat biztosít számunkra számos operációsrendszeren történő alkalmazás-fejlesztésre alapesetben C++ nyelven. A keretrendszert 1991-ben mutatta be a Trolltech és jelenleg a Digia és Qt Project fejleszti. A Qt rendelkezik mind nyílt forráskódú (LGPL) és kereskedelmi verzióval is. A fejlesztői környezetben olyan szoftvereket fejlesztenek, mint a KDE, AutoCAD, Skype, Adobe Photoshop, VLC és még sok más, valamint olyan cégek használják, mint a Google, Hewlett-Packard, Samsung, Volvo és még folytathatnánk a sort. Elsősorban C++ nyelvet támogatja, de ezen felül még akár C#, Java, Perl, PHP, Ruby nyelven is biztosít fejlesztést. A Qt alapértelmezett fejlesztőeszköze a Qt Creator, de más fejlesztői környezetekben is van lehetőségünk Qt fejlesztésre (például: Code::Blocks, Visual Studio). A keretrendszer egy másik érdekessége a QML (Qt Meta Language), mely egy JavaScript-szerű deklaratív programnyelv, amellyel látványos GUI-kat hozhatunk létre. Ezzel a lehetőséggel továbbá elkülöníthető egymástól a felhasználói felület és a programlogika.

Azok számára, akik még csak ismerkednek a keretrendszerrel, azoknak egy szimpatikus funkciója a Qt Creator-nak, hogy a szoftveren belül egy kattintással lehet példaprogramokat letölteni, így sokkal kényelmesebb a rendszer megismerése. Kényelmes és könnyen használható a rendelkezésre álló sűgő is, mellyel áttekinthető az egész Qt programkönyvtár. A fejlesztői környezet kezelése természetes módon adódik, a kódszerkesztő könnyedén felosztható több kisebb ablakra. Rendelkezik egy Design menüponttal, ahol egyszerűen össze lehet rakni a szoftverünkhöz szükséges felhasználói felületet, mint azt, ahogy a Visual Studio-ban előzőleg megszoktuk. A Debug-olás is egy külön menüt kapott, amivel a hagyományos módon lehet a fejlesztett szoftvert futás közben átvizsgálni. Továbbá van lehetőség Analyze funkcióra is, mely a szoftver erőforrás foglalási statisztikáit állítja elő, de ezt az eddigi munkáink során még nem volt rá szükségünk. Ezen felül a Qt, bár maradéktalan osztálykönyvtárat biztosít a fejlesztésekhez, az Interneten nagyon sok harmadik fél által készített nyílt forráskódú és könnyedén beépíthető programkönyvtár érhető el, mellyel további plusz funkciókkal lehet ellátni a fejlesztői környezetet.

Nagyvonalakban ezek voltak azok a funkciók és tulajdonságok, amik miatt a Qt-ra esett a választásunk. A további előnyöket sorolhatnánk oldalakon keresztül, nem is beszélve a különböző programkönyvtárakhoz tartozó osztályok ismertetéséről, de ezekről csak akkor fogunk említést tenni, ha azt a kontextus megkívánja. A következő alfejezetben a rendszerünk alapkonceptióját szeretnénk részletesen bemutatni, majd a tényleges rendszertervet mutatnánk be.

3.2 ALAPKONCEPCIÓ

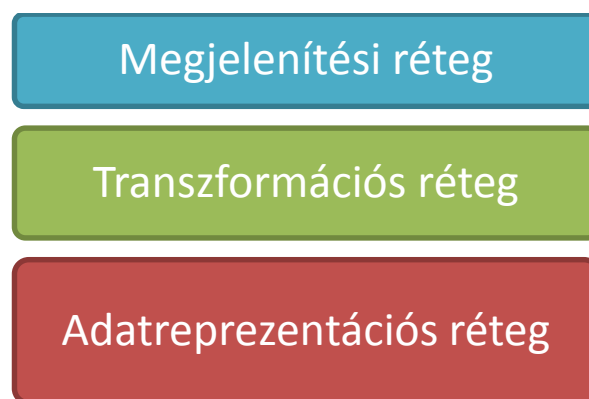
Az ionoszférikus mérések és az ionogramok áttanulmányozása után kialakítottunk egy alapkonceptiót, mely mentén elindíthatóvá vált a tervezési fázis.

Az alapötletünk az volt, hogy az ionogramokat kvázi-bináris képként kezeljük. Ennek előnye, hogy így különféle képelemző algoritmusok futtathatók rajtuk. Ilyenek lehetnek például szűrő-, klaszterező- vagy mintafelismerő eljárások. Az ionogramnak az egyik nagy hátránya, hogy több ezer pontfelhőből áll, amely egyrészt különböző zaj jellegű pontokat, másrészt az elemzéshez szükséges releváns és összefüggő foltokat tartalmaz. Ezekből kb. 70% zaj jellegű foltok, melyek pontos kiszűrése nélkül nehezen meghatározhatóak az egyes aszimptotikus értékek. A képelemző algoritmusok tanulmányozása során akadtunk rá a Connected-Component Labeling (CCL) algoritmusra, mely képes úgy osztályozni egy bináris képet, hogy az összefüggő komponenseket közös osztályba sorolja, ezáltal a számunkra fontos összefüggő részek megmaradnak, míg a kisebb elemszámú osztályok szűrhetővé válnak. Az eljárás pontos leírására a későbbiekben fogunk kitérni.

Ezen öltet alapján a feldolgozó program három részből tevődik össze: (1) egy adatreprézenciós réteg, mely képes bináris képként tárolni az ionogramot, melynél a fő szempont, hogy könnyen kezelhető és rugalmas legyen, valamint a prezentációs réteg használni tudja. (2) Egy olyan transzformációs/logikai réteg, mellyel a memóriába eltárolt képen különböző műveleteket tudjunk végrehajtani, többek közt a CCL algoritmust, valamint további módszereket. (3) Prezentációs réteg, ami a kezelői felületet valósítja meg az elérhető funkciókkal és központjában az ionogram vizualizációja áll.

3.3 TERVEZÉS ÉS IMPLEMENTÁCIÓ

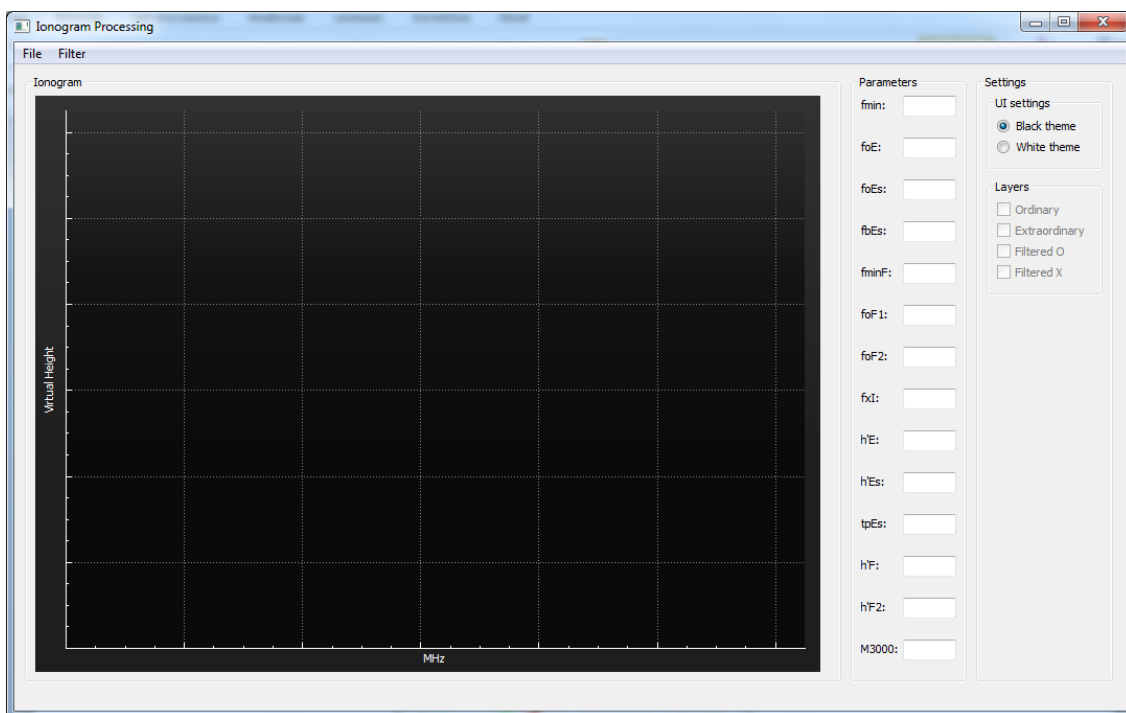
Az előzőekben leírt alapkoncepció alapján megtervezésre került a rendszer, melynél megmaradtunk a háromkomponenses megvalósításánál, amely megtekinthető a következő ábrán:



(1) Első körben megtervezünk az adatreprézenciós réteget, melynek fő funkciója a letárolás és lekérdezés. Az Ionogram osztály megtervezésekor, elsődleges szempontunk az egyszerű tárolás és a rugalmasság volt. A C++ jellemzőiből adódóan egy kétdimenziós vektor volt a legkézenfekvőbb megoldás a kép memóriában való eltárolására, melynek minden egyes eleme egy pont, melyeknek több különféle tulajdonsága is lehet a mérési adatoknak megfelelően. Ehhez megtervezünk egy saját Point struktúrára egy adott pontot reprezentál a képen és a méréshez tartozó jellemzőket tárolja. A Qt keretrendszer tartalmaz egy QPoint nevű struktúrát, de mivel nekünk több információt kell tárolnunk, mint x és y koordináta, így a saját implementációt kellett használnunk. Ezt úgy terveztük, hogy képes legyen a pontok koordinátáján felül a mérési értéket (0, 1, 2 vagy 3) és az adott ponthoz tartozó címkeszámot (szűréshez szükséges) is eltárolni. Továbbá, hogy képes legyen az egyes adattagokat a metódusain keresztül megváltoztatni (setter metódus), valamint azokat lekérdezni (getter metódus). Az Ionogram osztály ezek alapján képes az adatok letárolására, lekérdezésére az egyes metódusokon keresztül, valamint a módosítására (CCL algoritmus futtatása). Mivel a réteg a tároláson kívül az ionogramok szöveges fájlból való beolvasását is el kell, hogy lássa, így szükségünk lesz egy CSV beolvasó osztályra is, melyet az Ionogram osztályunk fel fog tudni használni betöltéskor. Az

osztály kifejezetten az ionoszonda szöveges kimenetére kell optimalizálva, mely könnyedén beolvassa a fájlokat és az adott formátumban adja tovább azt.

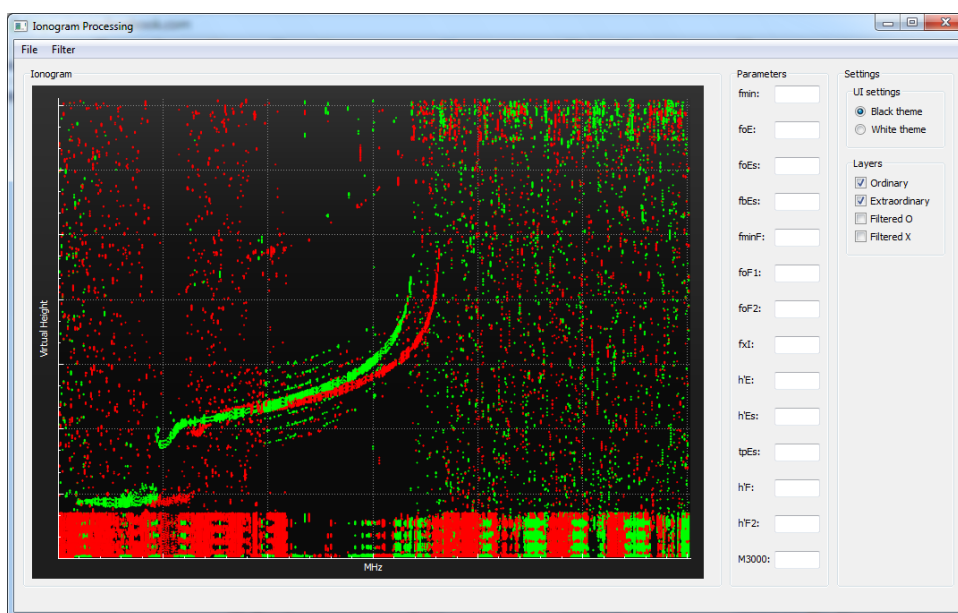
(2) A transzformációs/logikai réteg nem került részletes megtervezésre, mivel előreláthatólag nem tudjuk, hogy milyen eljárásokra lesz szükségünk. Elsőként egy képosztályozó eljárás került bele, melynek alapja a CCL algoritmus, melyet a következő alfejezetben szeretnénk részletesebben bemutatni. Ez adattagja lesz az Ionogram osztálynak, mely közvetlenül memória cím alapján kapja meg az adott képet és azon hajtja végre az algoritmust. Ezáltal nem kell nagy memóriaterületet másolni és újraírni, hanem a meglévő kétdimenziós vektor adataimein állítja át a címkeszámokat, mellyel a kép szűrhető válik. (3) Legvégül a prezentációs réteg került megtervezésre. Ehhez megterveztünk egy kezelői felületet, melynek központjában a QCustomPlot nevű függvény- és diagramkészítő áll, ami egy könnyen használható és rugalmas osztálykönyvtár, melyet az ionogramok megjelenítésére fogunk felhasználni. A felület létrehozásakor próbáltunk arra törekedni, hogy a felhasználónak a legkevesebb billentyűleütést kelljen végrehajtania, ezért minden funkciót, úgy alakítottunk ki, hogy azt egérrel véghez lehessen vinni. Továbbá a réteg képes az Ionogram osztállyal kommunikálni az egyes interface-eken keresztül, és olyan funkciókat ellátni, mint betöltés, megjelenítés és szűrés. Az ehhez elkészített MainForm megtekinthető az 5.ábrán.

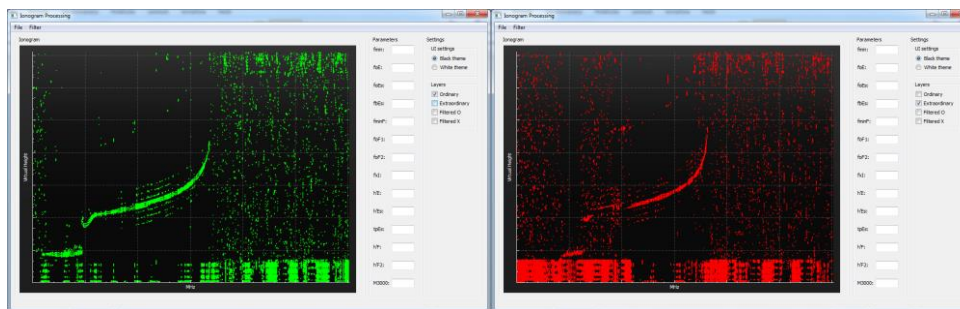


5. ábra - Kezelői felület bemutatása

A Kezelői felület bemutatása című képen látható, hogy a program nagy részét a QCustomPlot foglalja el a baloldalon, mely a vizualizációt látja el. Közvetlenül mellette láthatók felsorolva az elemzésekhez szükséges mezők, melyek az automatikus feldolgozáskor lesznek feltöltve értékekkel. A felület jobb oldalán pedig egy Settings menü található, ahol változtathatunk a UI témáján fekete és fehér színbeállítások közt a jobb átláthatóság végett. Helyet kapott itt egy rétegválasztó is, mellyel a plot-ra kirajzolható 4 ponthalmaz (ordinary-, extraordinary komponens, valamint ezeknek a szűrt változatai) között tudunk váltani, vagy azokat összehasonlítani egymással. A menüsor egyelőre még nincs teljes mértékben feltöltve, az eddig elkészült funkciók elérhetők és használhatók, viszont nem tudjuk, hogy a későbbiekben mivel fog bővülni a rendszer.

A teljesség igénye nélkül kitérnénk az implementálás folyamatára is. A rendszerhez először elkészítettük az előzőekben bemutatott kezelői felületet. Ehhez létrehoztuk a form-ot, valamint a mögötte elhelyezkedő programlogikát. A programlogika feladata, hogy a betöltött és letárolt ionogramot kirajzolja a felületre, valamint azon az egyes műveleteket elindítsa. A QCustomPlot beépített zoomolási funkcióját is ki kellett bővítenünk, mivel nem volt képes a túlgörgetés megakadályozására. Ezt a C++ egyik sajátosságával, a slot és signal összekapcsolással oldottuk meg. Ennek nagyvonalakban az a lényege, hogy a görgetést, mint signalt, hozzácsatoltuk az általunk megírt metódushoz (slot). Így ha a plot-on görgetés történik, az kivált egy jelet és a kapcsoláson keresztül meghívja a metódust, így ha túl akarnánk zoom-olni a felületen, akkor a metódus automatikusan megakadályozza azt. Ezt követte az Ionogram és azt a támogató osztályok implementálása és tesztelése, valamint összehangolása a kezelői felülettel. Ennek szemléltetése látható a 6. ábrán.





6. ábra - (Felül) Betöltött ionogram mindkét komponensének megjelenítése, (Bal alul) csak ordinary komponens, (Jobb alul) csak extraordinary komponens

A program a hardverközeliségének köszönhetően a betöltést és a kirajzolást gyakorlatilag pár ezer milliszekundum alatt képes elvégezni.

1. táblázat – Futási idő a pontok számának függvényében

Pontok száma	Futási idő
13708 db	1126 ms
56882 db	2047 ms
54588 db	2037 ms

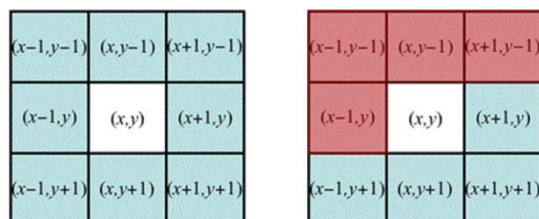
Az 1. táblázatban látható, hogy az egyes betöltésekkor mért futási idők szinte elenyészőek, ahhoz képest, hogy a pontok száma nagy. Továbbá azt is tekintve, hogy a mért idő nem csak a kirajzolásra értendő, hanem az ionogram fájlból való kiolvasására és betöltésére az általunk fejlesztett adatstruktúrába, valamint az abból való lekérdezésre is. Vagyis a teljes betöltési folyamat a fájl beolvasásától a pontok kirajzolásáig nem sokkal több, mint két másodpercet vett igénybe 50 ezer pont esetében, mely több mint elfogadható.

A fejlesztések további szakaszában a Transzformációs réteg létrehozása volt a cél, melybe majd az elemzési technikák kerülnek. Ennek első eleme a Connected-Component Labeling algoritmus volt, mely egy képosztályozási eljárás. Ez lényegében végighalad a képen és felcímkézi a vizsgált összefüggő komponenseket, amit utána könnyedén fel tudunk használni a kép szűrésére. Ennek ismertetése a következő alfejezetben megtekinthető.

3.4 CONNECTED-COMPONENT LABELING ALGORITMUS, MINT SZŰRÉS

Az algoritmus ezen implementációja két fázisból[7] tevődik össze. Az első fázis pixelről pixelre halad végig a képen, majd ha az adott pixel az általunk megadott értékű (pl.: ordinary, azaz 1), akkor megvizsgálja a szomszédjait. Az alapalgoritmus mind a nyolc szomszédot vizsgálja egy adott pontnál, viszont mi ezt lecsökkentettük csak négyre. Ez amiatt lehetséges, hogy az eljárás végeztén egy vizsgált pont körül csak közvetlen előtte lévő egy és felette elhelyezkedő három pont lehet felcímkézve, a többi csak a következő lépések során címkéződne fel. Az 5. ábrán látható

pixelekből az eljárás során mi csak a $(x-1, y-1)$, $(x, y-1)$, $(x+1, y-1)$, $(x-1, y)$ koordinátájú szomszédokat vizsgáltuk.



5. ábra - Vizsgált pixelek számának leszorítása

Az algoritmus, ha ekkor nem talál felcímkézett pixelt a szomszédoknál, akkor az aktuális új címkeszámot (label) kapja meg, amely az eljárás során iterációként növekszik. Ha van körülötte felcímkézett szomszéd, akkor két eset lehetséges:

1. Ha csak egy szomszéd van, akkor annak a címkeszámát kapja meg.
2. Ha nem csak egy szomszédja van címkézve, akkor a kikeresett szomszédokat címke szerinti szám-sorrendbe rakja, úgy hogy csak az egyedi értékek szerepelnek, majd:
 - a. Ha továbbra is csak egy szomszédja van, akkor annak a címkéjét kapja meg.
 - b. Ha nem csak egy van, akkor a legkisebb címkével rendelkező szomszéd címkéjét megkapja és az összes többi szomszéd szülőjévé teszi a legkisebb címkét.

A szülőcímkék eltárolás fa szerkezetben, az Unió-Holvan adatszerkezettel lehetséges, melyre a következő alfejezetben térnénk ki. Az adatszerkezetet lényegében úgy használjuk fel, hogy az egyes címkék felmerülésükkor felvesszük az adathalmazba és az algoritmusban leírtak alapján van beállítva a szülőmutatójuk. Az első fázis után a teljes képünk fel van címkézve és az adatszerkezetben is be vannak állítva a különböző szülőmutatók. Ekkor, viszont még maradnak olyan foltok, ahol több különböző címkével rendelkező és ténylegesen kapcsolódó komponensek vannak. Egy ionogramon lefutott első fázis a következő képen látható címkézést eredményezi egy adott foltnál:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	490	490	490	487	485
0	0	0	0	0	0	0	0	0	0	0	0	492	492	490	490	490	487	485	485
0	0	0	0	0	0	0	0	493	493	492	492	490	490	490	490	487	485	485	485
0	0	0	0	0	0	494	493	493	493	492	492	490	490	490	490	487	485	485	485
0	0	0	0	0	0	494	493	493	493	492	492	490	490	490	490	487	485	485	485
0	0	0	0	494	493	493	493	492	492	490	490	490	487	0	485	485	485	485	485
0	495	495	494	494	493	493	492	492	490	490	490	487	487	485	485	485	485	485	485
495	495	494	493	493	493	492	492	492	490	490	490	487	487	485	485	485	485	485	485
0	494	493	493	0	0	0	0	0	0	0	0	485	0	0	0	0	0	0	0

6. ábra – Első fázis által felcímkézett folt

A második fázis és az Unió-Holvan adatszerkezet együttesen képes ezt a problémát kiküszöbölni. Ez lényegében egy újracímkezés, mely során az adott címkéhez tartozó szülőt kell megkeresni az Unió-Holvan adatszerkezetben, majd felülről a szülő címkéjével az adott címkét. A második fázis lefutása után a 6. ábrán látható folt a következőre módosult:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	485	485	485	485	485
0	0	0	0	0	0	0	0	0	0	0	0	485	485	485	485	485	485	485	485
0	0	0	0	0	0	0	0	0	485	485	485	485	485	485	485	485	485	485	485
0	0	0	0	0	0	485	485	485	485	485	485	485	485	485	485	485	485	485	485
0	0	0	0	0	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485
0	0	0	0	485	485	485	485	485	485	485	485	485	485	0	485	485	485	485	485
0	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485
485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485	485
0	485	485	485	0	0	0	0	0	0	0	485	0	0	0	0	0	0	0	0

7. ábra – Újracímkezés eredménye

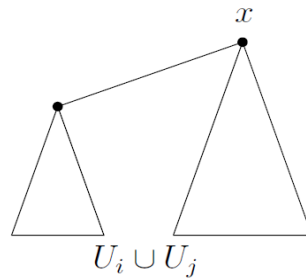
Látható, hogy az újracímkezés és az Unió-Holvan adatszerkezet együttes eredményeként a foltban szereplő pontok egyazon címkét kaptak, mellyel az eljárás teljessé vált. A későbbiekben részletesebben bemutatjuk az itt említett Unió-Holvan adatszerkezetet, amely implementálásánál különböző gyorsítási heurisztikákat használtunk, melyek gyorsabb működést eredményeznek a futtatás során.

Az algoritmus egy kis utómunkával könnyedén átalakítható szűrésre. Ez úgy lehetséges, hogy a CCL futtatása után a különböző foltokban elhelyezkedő pontok számát összegezzük, majd egy küszöbérték meghatározásával, csak azokat a foltokat tartjuk meg, melyek az adott küszöbszám felett helyezkednek el. Ez esetünkben azt teszi lehetővé, hogy a kisebb zaj jellegű foltok eldobásra kerüljenek, viszont az elemzéshez szükséges nagyobb elemszámú, görbeszerű foltok megmaradnak. Mindezt úgy hoztuk létre, hogy a felhasználó tudja megadni, melyik komponenst szeretné szűrni, valamint mekkora küszöbérték alatt kerüljenek eldobásra a foltok. A szűrés eredménye ekkor egy újabb rétegbe kerül rá a QCustomPlot-ra, így a bal oldali Settings menübe a felhasználó képes összevetni a szűrt és az eredeti adatokat is. Ezen eljárás részletesebb bemutatása és szemléltetése az Eredmények című főfejezetben olvasható.

3.5 UNIÓ-HOLVAN ADATSZERKEZET MEGVALÓSÍTÁSA UNION-BY-RANK ÉS ÚTÖSSZENYOMÁS HEURISZTIKÁVAL

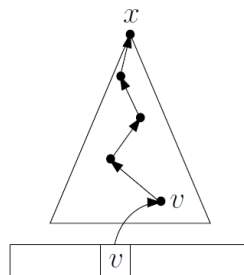
Az Unió-Holvan adatszerkezet az egyik legismertebb tárolási struktúra a számítástudományban. Legfőképp gráfelméleti területen használják, ahol a Kruskal algoritmus megvalósításánál alkalmazzák, mely képes minimális költségű feszítőfák keresésére összefüggő súlyozott élű gráfokban. Az adatszerkezet azt a feladatot képes megoldani, amikor egy véges S halmaznak szeretnénk egy $(U_1, U_2, \dots, U_k)$ részhalmazait adott számú partíciókra osztani. Ennek megvalósításához két műveletet használ:

- 1) Unió (U_i, U_j): Egyesíti a két részhalmazt, egyetlen halmazzá.



8. ábra – Unió, két részhalmaz összekapcsolása[8]

- 2) Holvan (v): Meghatározza az adott elem melyik részhalmazba tartozik.

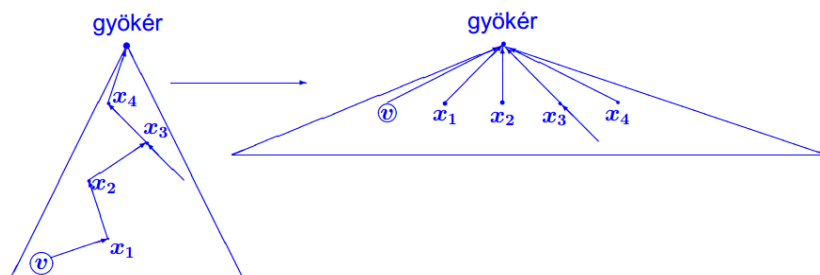


9. ábra – Holvan művelet[8]

Az adatszerkezet ezeken kívül még felhasználja a `MakeSet()` utasítást, mellyel új elemet lehet hozzáadni a meglévő S halmazunkhoz. Az adatszerkezet megvalósítása fákkal történik, ahol egy adott részhalmaz egy gyökérrel rendelkező, felfelé irányított fa. Ennek csúcsaiban tároljuk el a különböző részhalmazokat, melyek szülőmutatóval hivatkoznak a felettük lévő csúcsra. A szülőmutató a fa gyökéréleme esetében saját maga lesz. Az S halmaz elemeit egy vektorban tároltuk, melynek v indexű eleme v . Az adatszerkezet műveleteinek algoritmikus költsége jól ismert számítástudományban. Ez Unió művelet esetén $O(1)$ költség, mivel ekkor csak az egyik fa szülőmutatóját kell átállítani. A Holvan művelet költsége, pedig $O(\log n)$, mivel ha egy v csúcs új szülőmutatót kap, így egy szinttel távolodik a gyökértől és fájának mérete legalább az eredeti fa duplájára nő. Egy csúcs legfeljebb $\log_2 n$ -szer kerülhet új szülő alá (a fa szintjeinek száma legfeljebb $\log_2 n$), amiből adódik az $O(\log n)$ költség[8]. Könnyen belátható, hogy ez a naiv megközelítés egy sima láncolt listának is felfogható. Ennek eredményeként a felépített fák mélysége gyorsan növekedhet egyes esetekben, valamint erősen kiegyensúlyozatlan fák is létrejöhetnek, melynek következménye az Holvan művelet futási idejének növekedése. Ezen problémák kiküszöbölése két heurisztika alkalmazásával optimalizálható.

Az első az ún. Union-by-rank heurisztika, amely végett mindig a kisebb mélységű (rank) fa lesz a nagyobb mélységű fa gyökeréhez kapcsolva. Mivel a fák mélysége nagyban befolyásolja a futási időt, így a (X) kisebb mélységű fa hozzákapcsolása a (Y) nagyobb fa gyökeréhez, mely nem növeli (Y) mélységét csak egy részfat kap. A mélység csak abban az esetben növekedhet, ha (X) és (Y) mélysége egyenlő. Ekkor az egy elemű fák mélysége nulla lesz, és amikor két egyenlő mélységű (n) fa összekapcsolásra kerül, akkor a mélység (n)+1-re változik. Az implementációba ez annyi változást idéz elő, hogy a csomópontok mélységét is tárolni kell, valamint az Unió műveletnél vizsgáljuk a mélységek egymáshoz viszonyulását. A heurisztika önmagába való felhasználása az algoritmikus időt a legrosszabb esetben is lecsorítja $O(\log n)$ költségre.

Az előző heurisztika önmagában is nagy előrelépés az adatstruktúra szempontjából, viszont az általunk alkalmazott második heurisztikával még nagyobb javulást lehet elérni, melyet útösszenyomásnak hívnak. Ez egy további módja annak, hogy az algoritmus műveletidejét csökkentjük, mégpedig úgy, hogy a fákat tovább laposítjuk. Ez abból áll, hogy a gyökérbe vezető utakat összenyomjuk. A heurisztika alkalmazása az Holvan műveletben jelenik meg, amely rekurzívan megkeresi a gyökérelmet és ezt teszi szülőelemévé az adott fának az Unió műveletnél. Ennek szemléltetése megtekinthető a 10. ábrán.

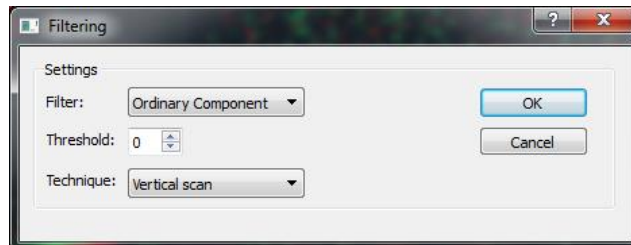


10. ábra - Útösszenyomás

A módszer alkalmazásával a létrehozott fa struktúrák sokkal laposabbak lesznek, ami felgyorsítja a rajtuk végrehajtható műveleteket. A két heurisztika együttes használata végett, ha egy olyan utasítássorozatot hajtunk végre, amelyben $(n - 1)$ darab Unió és $(m \geq n - 1)$ darab Holvan szerepel, akkor a költség $O(m \alpha(m))$ lesz. Ahol az $\alpha: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ függvény az úgynevezett inverz Ackermann-függvény. Melynél, ha m tart végtelenbe, akkor a függvény is végtelenbe tart, de sokkal lassabban, mint a k -szor önmagába helyettesített $\log \log \log \dots \log m$ függvény. Például egy $m < 265536$ esetén $\alpha(m) \leq 4$. Vagyis normálméretű feladatoknál ezt állandónak (≤ 4 -nek) lehet venni[9].

4 EREDMÉNYEK

A következőekben szeretnénk bemutatni az eddig implementációval elért eredményeinket. Egyik ilyen a szűrés funkció, mely a grafikus felület Filter menüjéből érhető el. Mivel az általunk készített szűrő eljárás több állítható paraméterrel is rendelkezik, ezért létre kellett hoznunk egy felugró ablakot, melyen az egyes értékek beállíthatók.

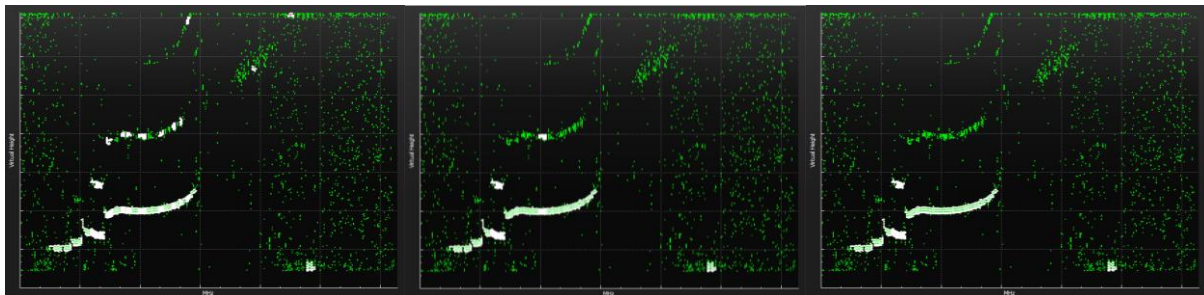


7. ábra - Filtering DialogBox kinézete

A 7. ábrán látható, hogy a szűréshez három paramétert tudunk megadni:

1. Filter: melyik komponenst szűrje
2. Threshold: mekkora küszöbértéket alkalmazzon
3. Technique: milyen technikát használja

A szűrés funkció elkészültével és használatával, arra lettünk figyelmesek, hogy minél nagyobb küszöbszámot választunk meg, annál több irreleváns területet veszítünk a képekből.

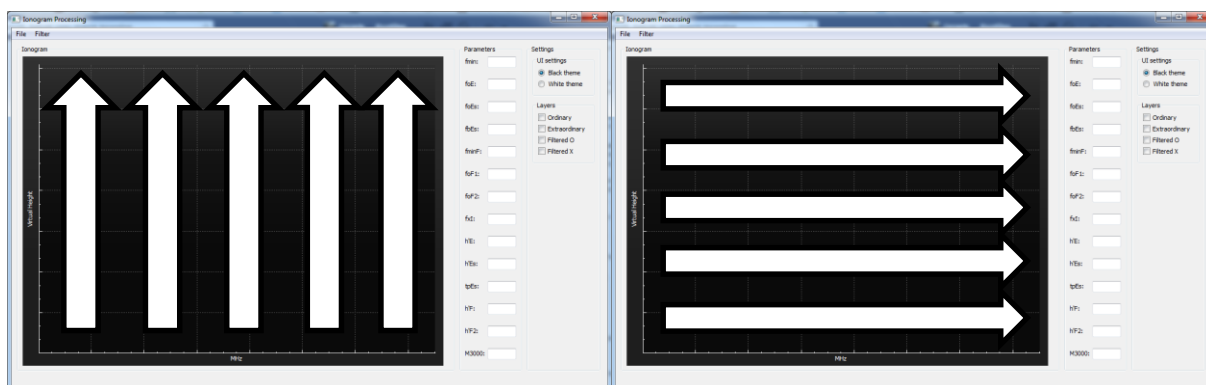


8. ábra - Szűrés különböző küszöbértékekkel

A 8. ábrán látható, hogy egyre kevesebb zaj jellegű folt maradt a képen. A bal oldali 30-as küszöbszám esetében a görbeszerű releváns folt felett sokkal több kisebb folt van, mint a jobb oldali 80-as küszöbértéknél. A továbbra is megmaradt zaj jellegű foltok a jobb oldali képen könnyedén kiszűrhetők lesznek a későbbiekben, mivel a kép jobb oldali lent elhelyezkedő kis foltja, nem értelmezhető tartományban van. Valamint a görbe felett megmaradt nagyobb folt, úgynevezett takarásban van a többi releváns folttal, így ez lehet, hogy egy szporadikus E réteg jelenség, vagy pedig zaj. Itt felmerült, hogy a rendszerbe beépítésre kerül egy frekvencia- és

magasság szűrés, mivel bizonyos magasság alatt és bizonyos frekvencia felett már csak nem értelmezhető adatok érkeznek be, de ez egyelőre még csak teszt fázisban van.

Az előző példa ordinary komponensén, 80-as küszöbértékkel és vertical scan technikát használva, az algoritmusunk 1490 milliszekundumot vett igénybe. Ez a kezdetben egy 30-40 másodperces folyamat volt, melyet az előző fejezetben leírt algoritmikus heurisztikákkal, a memóriaolvasások és újraírások számának redukálásával, valamint pontterekkel tudtunk leszorítani kevesebb, mint 1,5 másodpercre. Bár az előző fejezetben részletesebben kitértünk a CCL algoritmusra, de nem tettünk említést a Technique paraméterére, mellyel megadható, hogy az algoritmus milyen technikát használjon pásztázáskor.



9. ábra - (Bal) Vertical scan pásztázás, (Jobb) Horizontal scan pásztázás

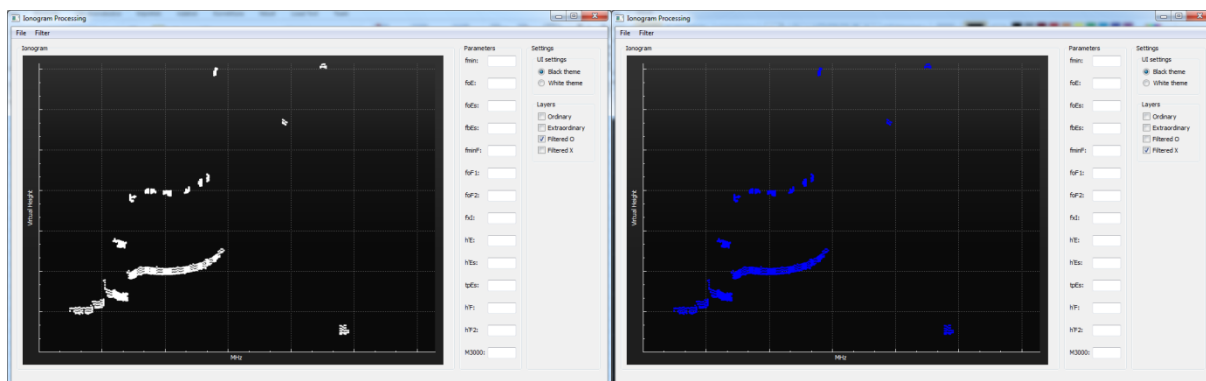
Az implementált CCL algoritmus használata során merült fel, hogy ne csak egy adott irányból lehessen végigpásztázni a képet, hanem külön vertikális (alapeset) és horizontális irányban is. A 9. ábrán látható, hogy a Vertical scan letről felfelé halad oszloponként balról jobbra, valamint a Horizontal scan pedig balról jobbra soronként letről felfelé. Ezzel azt szerettük volna vizsgálni, hogy a különböző irányok esetében változnak-e valamilyen módon a kapcsolódó foltok, esetleg jobb eredményt tudunk-e így elérni. Ehhez ki kellett egészítenünk a CCL algoritmust a horizontális pásztázással, melynek eredményét a 2. táblában foglaltuk össze.

2. táblázat - Szűrés után megmaradó pontok száma technika és küszöbérték függvényében

Küszöbérték	Vertical	Horizontal
20	3303	3303
30	3027	3027
40	2890	2890
50	2672	2672
80	2596	2596

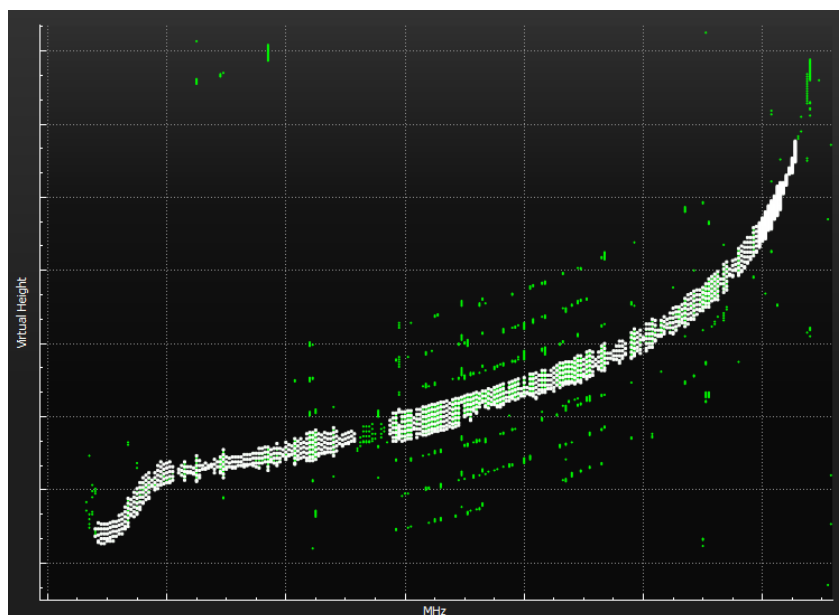
Szűrés után megmaradó pontok száma technika és küszöbérték függvényében című táblázatban az látható, hogy a két technika esetében nincs eltérés a megtartott pontok számában.

Szemléltetésképpen a 2. táblázat második mérése megtekinthető az Ordinary komponens szűrése CCL algoritmussal 30as küszöbérték mellett (bal) Vertical scan és (jobb) Horizontal scan technikával című képen



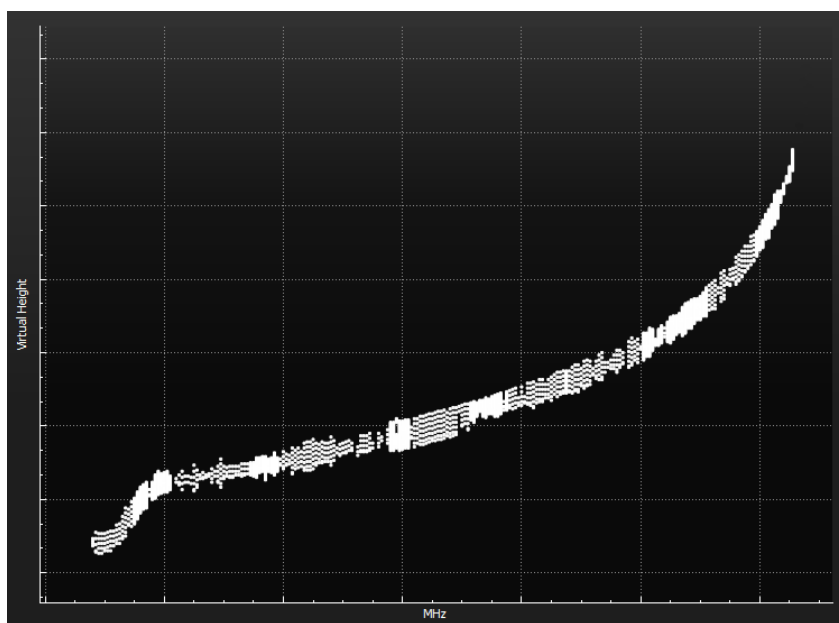
10. ábra - Ordinary komponens szűrése CCL algoritmussal 30as küszöbérték mellett (bal) Vertical scan és (jobb) Horizontal scan technikával

Az ábrán jól látható, hogy a két technika közt kirajzoláskor sincs eltérés, így belátható, hogy esetünkben a pásztázás iránya nem befolyásolja az algoritmus kimenetét. További ionogramok vizsgálata során, sok esetben szembetűnő volt, hogy egyes viszonylag közel elhelyezkedő foltok külön osztályokba kerültek, valamint voltak olyan esetek, ahol szakadások jelentek meg a görbeszerű foltokban. Ez annak volt köszönhető, hogy a CCL algoritmus csak 1 pixel távolságban képes vizsgálni a szomszédokat. Egy ilyen esete látható a F réteghez tartozó görbe szakadása című ábrán.



11. ábra - F réteghez tartozó görbe szakadása

Jobban ráközelítve a problémás területre tisztán látható, hogy a szakadásban elhelyezkedő pontok csupán 2 pixel távolságra vannak a görbe két részétől. Ennek vizsgálata szempontjából kiegészítettük úgy a CCL algoritmust, hogy az képes legyen több mint 1 pixel távolságban is szomszédokat keresni. Ennek egy problémája, hogy a kép szélein elhelyezkedő pixelek esetén null indexelést kapunk. Ezt az alap algoritmusban, úgy oldottunk meg, hogy kiegészítettük körbe a képet egy nullás értékeket tartalmazó kerettel. Belátható, hogy ekkor a két pixel távolságú keresésnél további egy kerettel kéne a képet kiegészíteni, három pixel távolság esetén hárommal és stb. Ezért a CCL algoritmust, úgy kellett módosítanunk, hogy ha az algoritmus a kép szélén halad, akkor csak egy pixel távolságot figyeljen, máskülönben kettőt. Az előző példánkra lefuttatott módosított CCL algoritmus a következő szűrést eredményezte.



12. ábra – F réteghez tartozó görbe javított szűrése

A 12. ábrán látható, hogy az 11. ábra szakadásos görbéje a 2 pixel távolságú szomszédfelderítéssel egyesült. Ezzel sikerült, az F réteghez tartozó görbét a zajtól elkülöníteni. Az algoritmusban használat 2 pixel távolságot a Csebisev távolság alapján határoztuk meg, melyet a következőképpen lehet demonstrálni.[10]

	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1		1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

13. ábra - Csebisev távolságok egy sakktáblán az F6 mezőtől

A 13. ábra sakktáblján, ekkor két pont távolságát a következő képlettel lehet meghatározni. Legyen az első pont $P_1(x_1, y_1)$, a második pedig $P_2(x_2, y_2)$, ekkor:

$$D_{Sakk} = \max(|x_2 - x_1|, |y_2 - y_1|).$$

Az algoritmusba továbbá felhasználtuk ellenőrzéshez a következő képletet:

$$Szomszédok\ száma = (2r + 1)^2 - 1.$$

Ahol r jelöli a Csebisev távolságot. Továbbá a 11. ábrára visszatekintve, ha a görbe felívelő szakaszának végét vizsgáljuk, felmerül a kérdés, hogy az ott elhelyezkedő pontok azok a görbe meghosszabbításának vehetők vagy sem? Erre egy megoldás lehet, hogy a távolságot szabadon tudjuk változtatni és ezzel további releváns pontokra jutunk. Ennek egy hátránya, hogy túlbonyolítja a programlogikát, mivel egy többágú feltételrendszert kell kialakítani, ahhoz hogy a különböző elhelyezkedésű pontoknak megfelelően történjen a szomszédfelderítés.

3. táblázat - Szomszédsági mátrix

0	0	0	0	0
0	3	6	9	0
0	2	5	8	0
0	1	4	7	0
0	0	0	0	0

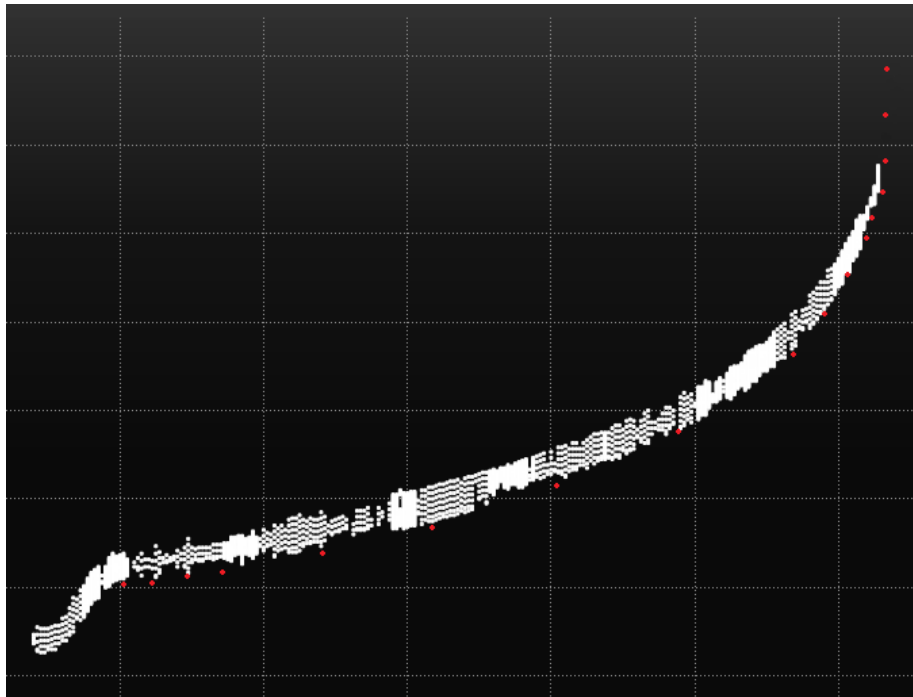
Például, ha a 3. táblázatban látható nem nulla cellák szomszédjait szeretnénk felderíteni 2-es Csebisev távolságig, akkor az egyes cellák szomszédjait különböző módon kéne keresni:

1. pont: 1 távolság lefelé, 1 távolság balra, 2 távolság felfelé
2. pont: 2 távolság lefelé, 1 távolság balra, 2 távolság felfelé
3. pont: 2 távolság lefelé, 1 távolság balra, 1 távolság felfelé
4. pont: 1 távolság lefelé, 2 távolság balra, 2 távolság felfelé
5. pont: 2 távolság lefelé, 2 távolság balra, 2 távolság felfelé
6. pont: 2 távolság lefelé, 2 távolság balra, 1 távolság felfelé
7. pont: 1 távolság lefelé, 2 távolság balra, 2 távolság felfelé
8. pont: 2 távolság lefelé, 2 távolság balra, 2 távolság felfelé
9. pont: 2 távolság lefelé, 2 távolság balra, 1 távolság felfelé

Könnyen belátható, hogy 3 távolság esetén, ez még tovább bonyolódik. Azt ezt megoldó algoritmus fejlesztése már elkezdődött, de nincs még demonstrálható eredményünk ezen a területen. Az elemzések szempontjából, mivel fontos, hogy pontosan meg tudjuk határozni a releváns területeket, így érdemes valamilyen módon felülvizsgálni az F réteghez hozzávehető további pontokat. Erről a módszerről a következő alfejezetben szeretnénk részletesebben írni, amely egy függvényillesztéses foltklaszterező algoritmus.

4.1 FÜGGVÉNYILLESZTÉSEL MEGVALÓSÍTHATÓ TÁVOLSÁGALAPÚ KLASZTEREZŐ MÓDSZER BEMUTATÁSA

A módszer lényege, hogy a szeparált foltra egy függvényt illesztünk, majd azt extrapoláljuk és megvizsgáljuk, hogy az extrapolált szakaszon vagy annak bizonyos közelségébe helyezkednek-e el pontok. Ekkor, ha van olyan pont, ami a közelbe esik, akkor azt az F réteg foltjához hozzávesszük. A módszer szemléltetése a Függvényillesztés bemutatása című képen látható:



14. ábra - Függvényillesztés bemutatása

A folt alsó pontjaira illesztjük a függvényt (piros pontok), majd extrapoláltjuk azt. Ezután pedig, az extrapolációtól megadott távolságra eső pontokat a folthoz hozzávesszük. Ezáltal az F réteghez tartozó kritikus frekvencia értékét sokkal pontosabban tudjuk, majd meghatározni.

A módszer fejlesztései elkezdődtek, de az eljárás egyik sarkalatos pontja, hogy nehéz olyan regressziós módszert találni, amely olyan függvényt tudna a foltra illeszteni, ami annak karakterisztikájával megegyezik. Néhány esetben exponenciális, néhol parabolikus és van, hogy 3-ad fokú polinom függvény jellegű, így nehéz meghatározni a legjobban illeszkedő függvényt. Továbbá nem szeretnénk túl nagy műveletigényű eljárást készíteni, mivel egy megoldás lehetne az, hogy megvizsgáljuk az összes szóba jöhető regressziót és azok illeszkedését, majd a legjobbat választjuk, de ez erőforrás-pazarló. Sajnos az alapesetekben sem tudtunk áttörő eredményt elérni, így ezzel kapcsolatos eredményeinket nem tudjuk publikálni.

5 ÖSSZEFOGLALÁS

TDK dolgozatunk során bemutattuk a geofizika által kutatott ionoszférát és annak vizsgálatát. Ismertettük az ionoszféra egyes rétegeit és a rétegekben végbemenő fizikai jelenségeket. Részletesen áttanulmányoztuk a mérésekből származó ionogramokat, valamint azok kiértékelését. Rámutattunk az ionoszondás mérések miéértjére, valamint ionogramok automatikus kiértékelésének nehézségeire.

A tudományos áttekintés után, kialakítottunk egy alapkoncepciót, mely alapján megtervezésre került a rendszerünk. A fejlesztések az általunk kiválasztott Qt keretrendszerben történtek, mely egy könnyen kezelhető fejlesztői környezetet és egy keresztplatformos osztálykönyvtárat is biztosít, mely nem csak C++ programozási nyelvvel használható. Implementáltuk az általunk megtervezett háromkomponenses koncepciót, mely (1) Adatreprezentációs-, (2) Transzformációs- és (3) Megjelenítési rétegből tevődik össze. Ezek közül az (1) látja el az ionogramok kvázi-bináris képként való betöltését és tárolását adott struktúrába, a (2) foglalja magába a programlogikát, mely az elemzésekhez szükségesek, valamint a (3), melynek a központjában a QCustomPlot áll és a vizualizációt látja el és magába foglalja a kezelői felületet is. Ezek után a megtervezett osztályokat és kezelői felületet implementáltuk, majd bemutattuk a Transzformációs rétegben elhelyezkedő CCL algoritmust, melyet szűrőként használunk az elemzések során. Ennek implementálása során több heurisztikát is alkalmaztunk., melyek gyorsabb működést tesznek lehetővé.

A rendszer használata során észrevettük, hogy szűréskor emelkedő küszöbérték esetében egyre több irreleváns pontot tudunk kiszűrni, viszont ezzel szemben veszíthetünk értékeseket is. Ötleként merült fel, hogy a meglévő CCL algoritmusnál megadható legyen, hogy milyen irányba pásztázza a képet az algoritmus. Ezt implementálva kiderült, hogy az irány nem meghatározó a képosztályozás szempontjából. Rámutattunk a szűrési algoritmusunk egyik gyengeségére, hogy csak a közvetlen kapcsolódó komponenseket tudja figyelni, így egy pixelen túl már nem képes foltokat egyesíteni. Az újbóli módosítás után kiderült, hogy az algoritmus jobb eredményt tud így produkálni. A módszer implementálása során kiderült, hogy ha a távolságot szabadon választhatóvá tesszük, akkor az algoritmus, olyannyira túlbonyolódik, hogy ezt egyelőre nem tudjuk kezelni. Az elemzések szempontjából, mivel fontos, hogy pontosan meg tudjuk határozni a releváns területeket, így érdemes valamilyen módon felülvizsgálni az F réteghez hozzávehető további pontokat. Ehhez bemutattuk az általunk kitalált módszert, amely egy függvény extrapolációját használja fel további klaszterek összevonására.

5.1 TOVÁBBLÉPÉSI LEHETŐSÉGEK

Az egyik legfontosabb továbblépési lehetőségünk az előzőekben leírt két, be nem fejezett módszer tökéletesítése, melyek nagyban segítenék a további munkákat. Ezek alapján el tudnánk kezdeni az egyes rétegekhez tartozó értékek automatikus leolvasását.

Ezzel igény merült fel egy olyan betöltési funkcióra, amely képes az ionoszonda által létrehozott nyers bináris fájlt beolvasni. Továbbá a betöltés során megadható lenne, hogy melyek azok a magasság és frekvencia tartományok, amelyek kiszűrhetők, így azok nem kerülnének beolvasásra.

Ezen felül egy olyan funkció is beépítésre kerül a rendszerbe, amely képes egy PostgreSQL adatbázist figyelni, és ha oda új ionogram érkezik be, akkor azt automatikusan betölti. Ezt a későbbiekben kiértékelné automatikusan és a kapott eredményeket egy riport modulon keresztül lehetőség lenne egy adatbázisba töltésre, vagy szöveges fájlba mentésre.

6 IRODALOMJEGYZÉK

- [1] Bibl, Klaus (1998 November-December): Evolution of ionosonde, *Annali di geofisica*, Vol. 41. <http://www.earth-prints.org/bitstream/2122/1484/1/02%20bibl.pdf> (Megtekintve: 2014. október 22.)
- [2] Vigh, Lajos: Az ionoszféra hatása a rádióhullámok terjedésére, <http://radiomet.mcse.hu/cikkek/vigh.pdf> (Megtekintve: 2014. október 3.)
- [3] Naval Postgraduate School, Meteorology Department: HF and Lower Frequency Radiation - The Ionosphere and the Sun, http://www.met.nps.edu/~psguest/EMEO_online/module3/module_3_2.html (Megtekintve: 2014. október 10.)
- [4] Davis, Chris: The ionogram, UK Solar System Data Centre, http://www.ukssdc.ac.uk/ionosondes/ionogram_interpretation.html (Megtekintve: 2014. október 30.)
- [5] INGV, AutoScala, <http://roma2.rm.ingv.it/en/facilities/software/18/autoscala> (Megtekintve: 2014. október 14.)
- [6] Eötvös Lóránt Tudományos Egyetem: A C++ nyelv, <http://www.cs.elte.hu/linfo/Prog/Forditok/cpp/cpp.htm> (Megtekintve: 2014. október 3.)
- [7] Shapiro, Linda G., and Stockman, George C. (2002): *Computer Vision*, Prentice Hall, pp. 69–73
- [8] Dr. Szabó László (2010), *Optimalizációs algoritmusok előadás*
- [9] Fredman, Michael L., Saks, Michael L. (1989): The cell probe complexity of dynamic data structures, at proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, pp. 345–354
- [10] Cyrus. D. Cantrell (2000). *Modern Mathematical Methods for Physicists and Engineers*. Cambridge University Press. ISBN 0-521-59827-3.