

TKP beszámoló:

GeoImage Workflow Editing Resources

Elek István

2021. március 21.

Tartalomjegyzék

I. A Giwer rövid bemutatása	6
1. Bevezetés	6
2. Célkitűzések	7
3. Megvalósítás	8
3.1. Lehetséges megoldási módok	8
3.2. A desktop alkalmazás	8
3.2.1. Data stock	8
3.2.2. Catalog	9
3.2.3. Workflow builder	10
3.2.4. Config editor	10
3.2.5. Help	11
3.2.6. Getting information	11
II. Részletes leírás	12
1. A Giwer logikai felépítése	12
2. A Giwer keretrendszer	13
3. Data Stock	13
3.1. A program működése	14
3.2. Támogatott adatformátumok	14
3.2.1. Tiff	14
3.2.2. Jpeg	15
3.2.3. Bil	15
3.2.4. DDM	17
3.2.5. Gwr	17
3.3. Osztályok és függvények	18
3.3.1. A GeoImageData osztály	18
3.3.2. A GeoImageTools osztály	18
3.3.3. A StatMath osztály	18
3.3.4. A GeoFilters osztály	18
3.3.5. A GeoMultibandMethods osztály	21
3.3.6. A DTM osztály	21
3.3.7. A Raster calculator osztály	21

3.3.8. Az ImageWindow osztály	21
3.4. A Project osztály	22
3.5. A Mosaic osztály	22
4. Függvények és eljárások bemutatása	23
4.1. A GeoImageData osztály eljárásai	23
4.1.1. parseBilHeader	23
4.1.2. parseTifHeader	23
4.1.3. parseGeoTifHeader	23
4.1.4. parseJpgHeader	24
4.1.5. getExif	24
4.1.6. parseGiwerHeader	24
4.2. A GeoImageTools osztály eljárásai	24
4.2.1. saveGiwerFormat	24
4.2.2. saveHeader2Giwer	24
4.2.3. convertByteArray2GiwerFormat	24
4.2.4. convertOneBandBytesToBitmap	25
4.2.5. getOneBandBytes	25
4.2.6. getOneBandToByteArrayFromBitmap	25
4.2.7. BitmapToByteArray	25
4.2.8. ByteArrayToBitmap	25
4.2.9. convertImageFromTif2Jpg	25
4.2.10. convertImageFromJpg2Tif	25
4.2.11. InvertColor	25
4.2.12. GrayscaleConversion	25
4.2.13. readGwrFile	25
4.2.14. GetAnRGBBand	26
4.2.15. combine2Images	27
4.3. A StatMath osztály eljárásai	27
4.3.1. getMinMax	27
4.3.2. imageAverage	28
4.3.3. imageScatter	28
4.3.4. imagesStandardization	28
4.3.5. computeCorrelationMatrix	28
4.3.6. compEigen	28
4.3.7. PCA	28
4.4. A GeoFilters osztály eljárásai	28
4.4.1. MedianFilter3Bands	28
4.4.2. MedianFilterOneBand	28
4.4.3. Canny	29
4.4.4. Laplace3Bands	29

4.4.5.	LaplaceOneBand	29
4.4.6.	ConvolSingleBand	29
4.4.7.	Convol3bands	29
4.4.8.	highPassKernel	29
4.4.9.	lowPassKernelGauss	29
4.4.10.	lowPassKernelBox	29
4.4.11.	resampling	32
4.4.12.	Prewitt	32
4.4.13.	Sobel	32
4.4.14.	Isotropic	32
4.4.15.	vectorize	32
4.5.	GeoMultibandMethods osztály eljárásai	32
4.5.1.	RGB	32
4.5.2.	NDVI	33
4.5.3.	Cross Plot	33
4.5.4.	compEigen	33
4.5.5.	compPCA	33
4.5.6.	Clustering	37
4.5.7.	Segmentation	37
4.6.	A DTM osztály	38
4.6.1.	readParameters	38
4.6.2.	readDDM	38
4.7.	A Raster Calculator osztály	38
4.8.	A Project osztály	38
5.	A DataStock alrendszer használata	41
5.1.	Menürendszer	41
5.1.1.	File	41
5.1.2.	One band processes	43
5.1.3.	Multiband processes	44
5.1.4.	Data tools	45
5.1.5.	Néhány hasznos funkció	46
6.	A Catalog alrendszer használata	47
6.1.	Első lépések	47
6.2.	A fájlrendszer előkészítése	49
6.3.	Az adatbázis feltöltése	50
6.4.	Funkciók	50
6.5.	Lekérdezés	52

III. Függelék	55
1. A távérzékelés fizikai alapjai	55
2. A főkomponens analízis matematikai alapjai	59
3. A digitális szűrési eljárások áttekintése	61
3.1. Konvolúciós szűrők	62
3.1.1. Szeparábilis szűrők	63
3.1.2. Box szűrő	64
3.1.3. Gauss-szűrő	64
3.1.4. Nem-szeparábilis szűrők	66
3.2. Éldetektorok	66
3.2.1. Gradiens szűrő	66
3.2.2. Laplace-szűrő	67
3.2.3. LoG szűrő	67
3.2.4. Emboss szűrő	68
3.2.5. Canny-féle éldetektor	68
3.2.6. Kép élesítése	70
3.3. Nemlineáris szűrők	70
3.3.1. Rangszűrők	70
3.3.2. Olimpiai szűrő	71
3.3.3. Konzervatív simítás	71
3.4. Szegmentálás, küszöbölés	72
3.4.1. Otsu-féle küszöbölés	72
4. Osztályozás, klaszterezés	74
4.1. Particionáló klaszterezés	76
4.2. Hierarchikus eljárások	76
4.3. Képek klaszterezése	77
4.3.1. ISODATA eljárás	77
4.3.2. Felügyelt osztályozás	77
5. Szegmentálás	79
5.1. A Giwer rendszer éldetektálási módszere	80
5.1.1. Sokszámos képek szegmentálása	80
6. Textúra elemzés	81

I. rész

A Giwer rövid bemutatása

1. Bevezetés

A térinformatika már több évtizede jelen van a tudományban és a gazdasági életben. Mára azonban szinte alig van olyan adatféléség, amelynek térbeli vonatkozása ne kapna szerepet a problémák megoldásában, hiszen már nemcsak a hagyományos területeken van jelen (agrárium, környezet- és természetvédelem, közmű-informatikai rendszerek, önkormányzat, ingatlan-nyilvántartás, stb.), hanem olyan új területeken is megjelent, mint a bankvilág, a biztosító társaságok, gyárak és ipari intézmények belső térinformatikai rendszere, társadalomtudományok, szociológia, politológia, bölcsészet (pl. régészett, nyelvészett) és még számos terület.

A térinformatika utóbbi évtizedben végmenet robbanásszerű változása az adatbázis-technológia fejlődésének és az open source rendszerek (Postgres/Postgis, QGIS, stb.) hihetetlen mértékű előretörésének köszönhetően teljesen átalakult. Megszűnt a nagy cégek hegemoniája (pl. ESRI, Integraph), és a szakterület eddig sosem látott mértékben az adatbázis-technológia világába betagozódott. Az okos telefonok előretörésével már a minden nap élet részévé vált a digitális térkép, a Google térképkezelő funkcionálisája fejlesztéseinek köszönhetően már bármely felhasználó hozzáférhet villámggyorsan digitális térképi tartalmakhoz, legyenek azok vektortérképek, ūrfelvételek, vagy domborzati modellek a világ szinte bármely pontjára vonatkozóan. Online adatgyűjtők adatait lehet térképen megjeleníteni ezen technológiai fejlődés eredményeképpen (vízszint adatok, meteorológia, gépjármű nyomkövetés, stb.). Mindezeknek köszönhetően a hagyományos térképészeti háttérbe szorult, viszont a megnövekedett digitális térkép igények miatt a térképekhez, azok létrehozásához és az informatikához is értők szaktudása felértékelődött.

A távérzékelésben is fejlődési ugrást jelentett az utóbbi években jelentősen megnövekedett adatmennyiség, a TB számra keletkező raszteres adat (fénykép, ūrfelvétel, hiperspektrális raszteres adatok, lidar, stb.). További örvendetes fejlődési elem, hogy az egykor csak katonai célokra használt robotrepülőgépek (UAV) ma már polgári célokra is egyre nagyobb mértékben használatosak. Még a nevük és megváltozott UAV-ról DRON-ra. Jelentőségük ott van, ahol nem nagy magasságból, és nem nagy, hanem kis területekről kell, nagy felbontással képet készíteni. Ilyen szakterület például a mezőgazdaság, pontosabban a precíziós mezőgazdaság, valamint a környezet- és természetvédelem, továbbá ilyenek a rendészeti, rendőri alkalmazások.

2. Célkitűzések

Szinte minden képértelmezés alapvető funkcionális eleme a kép felbontása összetartozó területekre (ez lehet osztályozás vagy szegmentálás). A hagyományos módszerek szegmentálnak, osztályoznak. Ezek futási ideje nagy méretű képekre igen hosszú lehet, ezért gyorsabb eljárás kidolgozását tűztem ki célul. Ennek elméleti alapjait és szintetikus adatokon elérte eredményeit már publikáltam 2019-ben. Ez az eljárás a látásunk éldetektálási képességének szimulációján alapulna, amely először a képek nagyobb egységeinek, markánsabb határainak meghatározását végezné el, majd ezeken belül állapítana meg kisebb egységeket. A látás fizikáját, fisiológiáját kutatók körében ismert tény, hogy a szemünkben nyolc irányban Gabor-szűrő detektál éleket, amelyek működése elképesztően gyors. (Az evolúció során a gyors kontúr megállapítás alapvető jelentőségű volt, mert a látott képen a detektált élek a felismerni kívánt objektumok határait jelentették, ami adott esetben a veszélyforrás vagy a táplálék felismerését volt hivatott szolgálni). Az így kapott élek jelentenék a következő szegmentálási fázis határait.

Számos további eljárást tervezek megvalósítani, amelyet digitális szűrőbank néven foglalnék össze, amely vázlatosan a következő eljárásokból állna: felül és alul vágó szűrők, sávszűrők, konvolúciós és szeparábilis szűrők, élmegőrzők, éldetektorok, küszöbölés, intenzitás transzformációk, szín konverziók, statisztikai elemzések, főkomponens analízis, stb. A felsorolás távolról sem teljes, sőt fejlesztés közben kiderülhet, hogy további eljárások implementálása is szükséges lehet.

További eljárások kidolgozását is tervezem, amelyeknek a távérzékelésben van jelentősége. Ezek az RGB frekvencia sávokon túl infravörös sávokat tartalmazó képek különböző sávjaiból számított képeket állítják elő (pl. NDVI: normalized differential vegetation index). Multispektrális képekre ezek már bevált eljárások, de hiperspektrális képekre még számos fejleszteni való van ezen a területen.

A *Giwer* rendszer felépítését a következőkben fogom összefoglalni. Két lehetséges megoldási módot ismertetek. Az első egy hagyományos megközelítés, amely egyetlen programban egyesíti a fenti funkcionalitást (ennek a neve *DataStock*). A másik lehetőség, hogy az eljárások tetszőleges kombinációját, vagyis workflow-k összeállítását tesszük lehetővé (ennek a neve *Workflow builder*)

3. Megvalósítás

3.1. Lehetséges megoldási módok

1. A pályázatban szeretnék kidolgozni egy programcsomagot, amely bár-mely térbeli referenciával rendelkező, úrből, repülőgépről vagy drónról készített kép feldolgozását képes elvégezni. A feldolgozáshoz számos eljárást fogok implementálni. Jelenleg úgy tervezem, hogy a sokféle eljárást egy nagy monolit programban fogom egyesíteni, amely interaktív üzemmódban, a felhasználó tudása és céljai alapján lesz működtethető. Mivel számos esetben nem tudni, hogy milyen eljárást kombinációk produkálják a legjobb eredményt, ráadásul az eredményeket emberi interpretátoroknak is kell vizsgálniuk, így az interaktív működési mód indokolt.
2. Az interaktív megközelítés mellett szeretnék egy olyan működési módot is megvalósítani, amely a fent említett eljárások függvényeit tetszőlegesen kombinálhatóvá teszi (**Giwer: GeoImage Workflow Editing Resoures**).

Mindkét megközelítés alkalmazható úgy a hagyományos forrásból származó képekre, mint más, például drón által készített képekre, legyenek azok RGB, infra vagy hiperspektrális kamerák képei. A két megoldási mód azonban nem alternatíva, hiszen mindenki szeretné valósítani.

3.2. A desktop alkalmazás

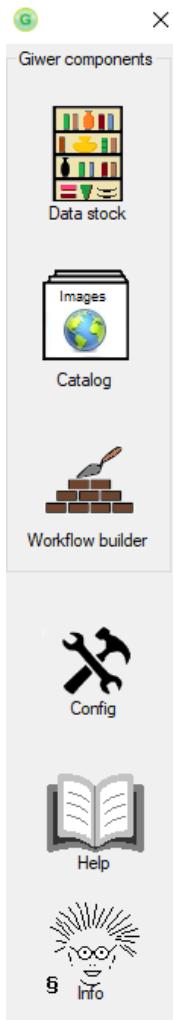
Egy keretprogram vezérli a különböző programrészeket. Ez a Giwer nevű program. Célja a rendszer működésének irányítása. Segítségével indíthatjuk el a monolit alkalmazást (*Data stock* ikon), és a workflow szerkesztőt és a futtatót (*Workflow builder* ikon). A keretrendszer induló képernyőjét mutatja a 1. ábra.

Itt indíthatjuk el a programok konfigurálását végző programrészt (*Config* icon), valamint a program használatát segítő leírást, végül pedig a program metaadatait bemutató információs részt (*About* ikon).

A *DataStock* és a *Workflow builder* önállóan is elindítható a keretrendszer nélkül, ha éppen úgy akarja a felhasználó.

3.2.1. Data stock

Ez az alkalmazás egy nagy, minolit program, amelyet interaktív működésre tervezünk. Számos függvényt fogunk implementálni, amelyek az adatok

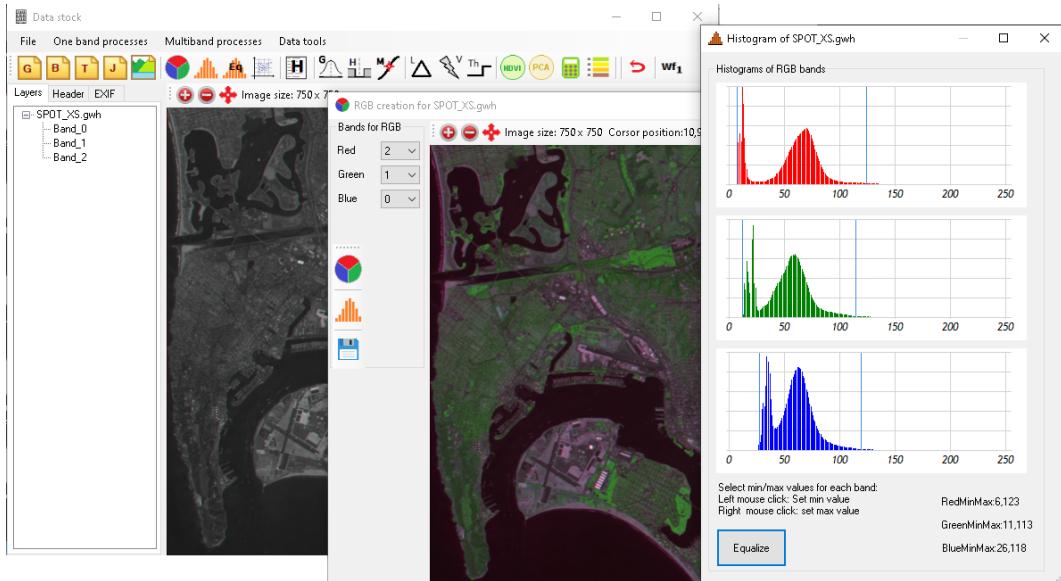


1. ábra. A Giwer keretrendszer

olvasását, írását, manipulálását végzik. Ezek a program menürendszerében fognak megjelenni, amit a felhasználó interaktívan, az egyes eljárások eredményességét vizsgálandó, aktivizálhat.

3.2.2. Catalog

A képek olyan nagy mennyiségben keletkeznek, hogy ezek áttekintése egy idő után reménytelen feladatnak látszik, és nagy a hibázás lehetősége. Ezért létrehozunk egy kép katalógust, egy nyilvántartó, kezelő alrendszeret, amely adatbázisban tárolja, rendszerezi a képeket.



2. ábra. A Data stock program egy képernyő képe

View/edit config file		
	Name	Value
▶	GiverDataFolder	D:\data\gwr
	BilDataFolder	D:\data\bil
	TifDataFolder	D:\data\tif
	JpgDataFolder	D:\data\jpg
	3DDataFolder	D:\data\dtm
*		

3. ábra. A Config viewer/editor képernyő képe

3.2.3. Workflow builder

A *Workflow builderrel* a rendelkezésre álló függvényekből tetszőleges munkafolyamatot (workflowt) állíthatunk elő.

3.2.4. Config editor

A *Config editorral*, amelyet a keretprogramból indíthatunk el (1. ábra) a rendszer adatforrásait állíthatjuk be (3. ábra). Megadhatjuk, hogy hol találhatjuk a fájlrendszerben az idegen formátumú adatokat (*bil*, *tif*, *jpg*), és a rendszer saját adatformátumú fájljait (*gwh*).

3.2.5. Help

A rendszer használatát angol nyelvű users' guide támogatja, amely a *Help* ikonnal aktivizálható. Jelenleg ez még nem áll rendelkezésre. Várhatóan akkor fog elkészülni, amikor már alapvető változtatások nem várhatók a rendszerben.

3.2.6. Getting information

Az *About* ikonra klikkeléssel a rendszer metaadatait nézhetjük meg (szerzők, verziószámok, copyright, stb.)

II. rész

Részletes leírás

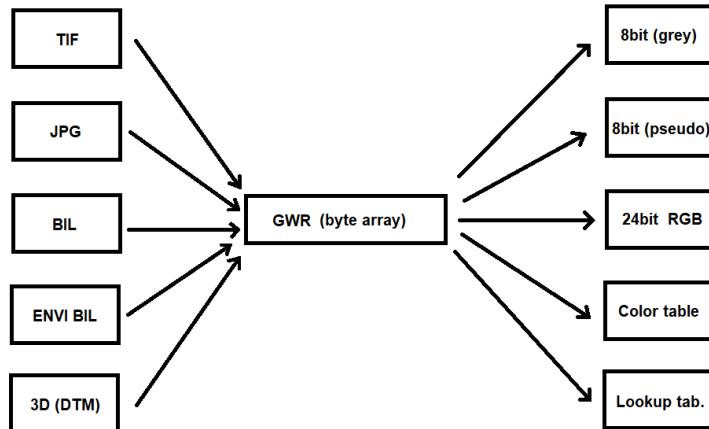
1. A Giwer logikai felépítése

Számos adatforrást kívánunk beolvasni (tif, jpg, bil, ddm) és vele műveleteket végezni. Az adatforrások közvetlen használata a feldolgozási műveletek során ugyan lehetséges, de nem elég gyors. Mivel gyakran óriási méretűek a képek, ezért a futási sebesség kritikus tényező, amit mindenkorban a lehető legkisebbre kell csökkenteni.

Ezért úgy gondoltuk, hogy saját adatformátumot fogunk használni, amely a gyors működést szolgálja. A nyers formátumoknak csak a megjelenítését biztosítjuk, hogy beolvasáskor lássuk, miről is van szó, de további műveletek nem engedélyezünk. Ha számításokat, képi elemzési funkciókat is szeretnénk végezni, akkor először át kell konvertáljuk a kívánt nyers fájlt *gwr* formátumba (4. ábra). A *gwr* egy bináris formátum, amely egy frekvenciasáv intenzitás értékeit tartalmazza sorfolytonosan, ahol is egy pixel egy bájt. Ahány frekvenciasáv van a képben annyi bináris fájlunk lesz.

A számítási műveletek főként egy frekvenciasávon történnek, kivéve a többsávos műveleteket (pl. RGB kép generálás, főkomponens analízis, NDVI számítás, stb.). Gyakran a frekvenciasávonkénti műveletek eredményeit egyesítjük egy képen.

Az eredmények megjelenítése igen lényeges része a funkcionalitásnak. A default megjelenítési stílus az, amikor greyscale-ként jelenik meg a kiválasztott frekvenciasáv. Számos számítási eredmény azonban éppen abban nyilvánul meg, hogy a kép intenzitás értékeit manipuláljuk, és a számított intenzitás értéknek fizikai jelentése van (pl. NDVI). Ezért szükséges, hogy olyan megjelenítést is alkalmazzunk, amely ezt a fizikai tartalmat teszi láthatóvá. Ilyenkor az RGB színmodellre alapuló 24 bites megjelenítés nem megfelelő. Ezért a megjelenítést nem 24 biten végezzük, hanem 8 bites color palette-t alkalmazunk, így a kép színes lesz, de legfeljebb 256 színű (4. ábra). Ekkor lehetőség nyílik arra, hogy a számítási eredményekhez a legmegfelelőbb színeket alkalmazzuk (pl. hipszometrikus színes megjelenítés magassági adatokhoz, vagy osztályozott képeken az egyes osztályok eltérő színnel történő megjelenítése.)



4. ábra. A **Giwer** adatforrásai és konvertálása saját (gwr) formátumba. Valamennyi adatformátumot előbb gwr formátumba konvertáljuk, és csak azután végzünk velük számításokat. A megjelenítéshez, attól függően, hogy milyen típusú adatról van szó, és milyen színmodellt kíván a megjelenítés, vagy color táblás /8 bites/ vagy RGB /24 bites/ megjelenítést hajtunk végre.

2. A Giwer keretrendszer

A keretrendszer összefogja a különböző alkalmazásokat, segít konfigurálni a programot. Innen Indítható az interaktív desktop alkalmazás, a DataStock. Ugyancsak innen indítható a Workflow edit. Innen állíthatjuk be a program konfigurációs fájlját (config.cfg), amely a startup könyvtárban (ahol található az giwer.exe) található. Ez a fájl a forrásadatok (*bil*, *tif*, *jpg*) elérési útvonalát tartalmazza (DataFolder). A gwr formátumú adatok helyét is megadhatjuk itt (GiwerDataFolder). A Help segítségével megnyithatunk egy pdf fájlt, amely a program használatában segít. Az About röviden bemutatja a programot, és a különböző programkomponensek verziószámait.

3. Data Stock

A *DataStock* raszteres képek beolvasását, manipulálását és az eredmények megjelenítését és tárolását végzi. Interaktív működésű. A rendelkezésre álló függvényeket a menürendszer által lehet meghívni. Számos képfeldolgozó eljárás lett beépítve, továbbá olyan adatkonverziós függvények, amelyekkel az optimális működéshez szükséges adatkonverziókat el lehet végezni.

3.1. A program működése

A *DataStock* alrendszer tervezésekor saját adatformátum kidolgozását tartottuk a legcélszerűbbnek. Alapelvek, hogy minden frekvenciasáv intenzitásértékeit egy-egy bináris fájlban tároljuk. A program ezeket az adatokat egy bytetömbben tartja, amivel gyorsan lehet műveleteket végezni. Mivel a kéatformátumok terén igen nagy a sokszínűség, ezért ezt a megoldást tartottuk a legjobbnak. Azáltal, hogy minden képet előbb saját formátumra hozunk (adat előkészítés), és az egységes adatszerkezetben végezünk műveleteket, így a későbbiekben könnyen bővíthetjük a támogatott fájlformátumok számát.

Kétféle műveletcsoportot hoztunk létre. Az egyik a *OneBand functions* a másik a *Multiband functions*. Az első csoportba azok a műveletek tartoznak, amelyek egyetlen frekvenciasáv adataival elvégezhetők (pl. digitális szűrések egy frekvenciasávra, hiszogram kiegyenlítés, adatkonverziók, stb.) A másik csoportba azok a funkciók tartoznak, amelyek egyszerre több frekvenciasáv adatait igénylik (pl. RGB képek létrehozása, PCA, NDVI, stb.)

A főbb funkciókörök a következők:

- különböző adatformátumú képek beolvasása és konverziója (*tif*, *jpg*, *bil*, *gwh*)
- képek megjelenítése
- egy és többsávos képfeldolgozási műveletek végrehajtása, amelyek a későbbiekben részletesen is be lesznek mutatva

3.2. Támogatott adatformátumok

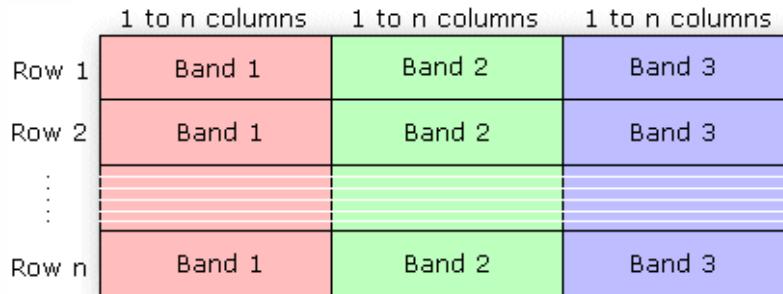
Háromféle rászteres adatformátum beolvasását támogatja a program: *bil*, *tif* és *gwr* (a *Giwer* saját bináris adatformátuma).

3.2.1. Tiff

A *tif* a jól ismert tagged image format, amelynek egy speciális, térinformatika célú változatát, a geo tiffet is használjuk. Ez a formátum nemcsak pixelenkénti intenzitás értékeit tárol, hanem a headerben a kép georeferencia adatait (vetületi rendszer, kalibrációs pontok /többnyire sarok pont koordináták/ egyéb a képkotásra vonatkozó adatok). Részletes leírása megtalálható a következő linken:

<https://earthdata.nasa.gov/esdis/eso/standards-and-references/geotiff>

Hagyományos *tiff* formátumot is támogat a program, amely nem tartalmaz georeferencia adatokat. Ilyenkor máshonnan vesszük a képek térbeli



5. ábra. A *BIL* fájl adatstruktúrája

referencia adatait. Általában minden tif kép tartalmaz *exif* adatokat, amelyek olvashatók. Mivel az *exif* nem szabványos, így az olvasásával számos nehézség előfordul. Például DJI drónra szerelt multispektrális Micasense és az RGB Zenmuse kamerák minden repülési adata, mint pl. drón pozíció, kép orientáció (északi iránynal bezárt szög), képméret, sávok száma, színmélység, stb. az *exif*-ben van tárolva. Ezeket csak erre a célra kidolgozott eljárásokkal tudjuk olvasni.

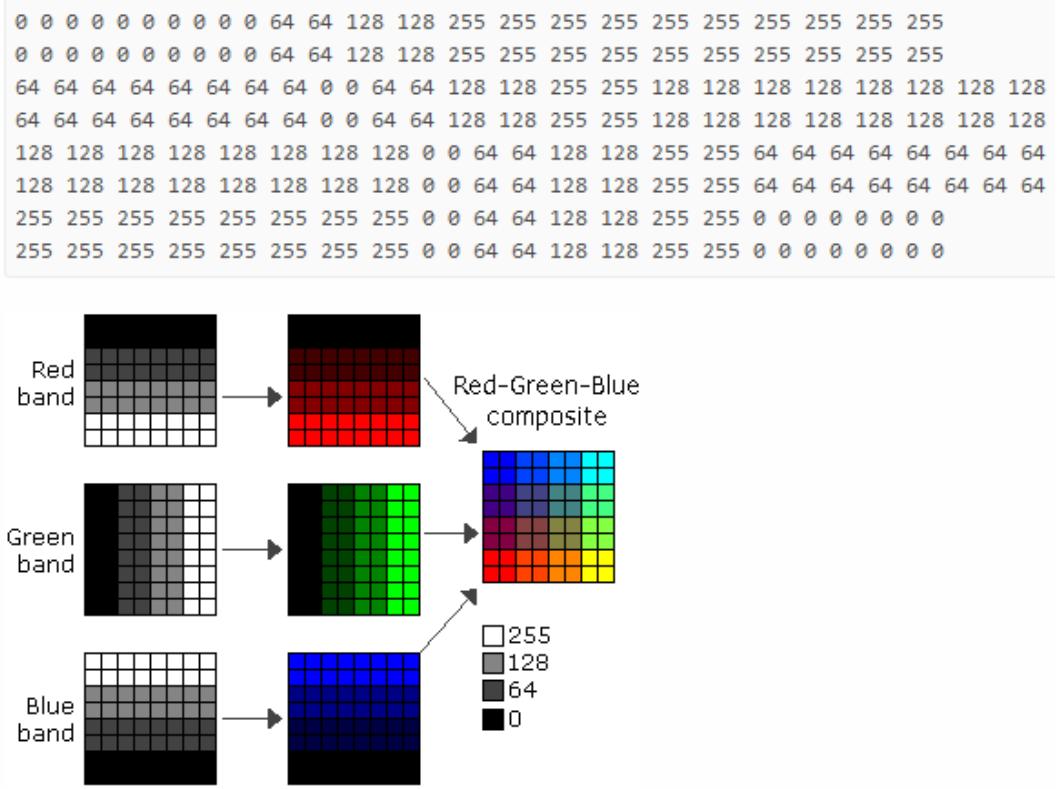
3.2.2. Jpeg

A jól ismert *jpg* formátumú képek beolvasását is támogatja a program. Ez többnyire nem tartalmat georeferencia adatokat. Bizonyos esetekben léteznek hozzá kiegészítő fájlok, amelyek tartalmaznak a vetületi rendszerre, kalibrációs pontokra vonatkozó adatokat. Drón képek esetén ilyen fájlok nincsenek, csak a hagyományos *jpg* fájl. Hasonlóan a *tiff* fájlokhöz, a *jpg* fájlok leíró adatait, a replüléssel összefüggő kép adatokat csak erre a célra kidolgozott *exif* eljárásokkal lehet beolvasni.

3.2.3. Bil

A *bil* egy speciális formátum, amelyet sokszámos ūrfelvételek tárolására dolgoztak ki. Részletes leírása megtalálható a következő linken:
<http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/bil-bip-and-bsq-raster-files.htm>

Röviden összefoglalom a *BIL* leírását, mert kiemelt jelentőségű, és a harmadik formátum, *gwr* ismertetéséhez is szükség lesz rá. A *BIL* a Band Interleaved by Line szavak rövidése, amelyet multispektrális (vagyis több frekvenciasávos) képek tárolására fejlesztettek ki. A *BIL* önmagában nem

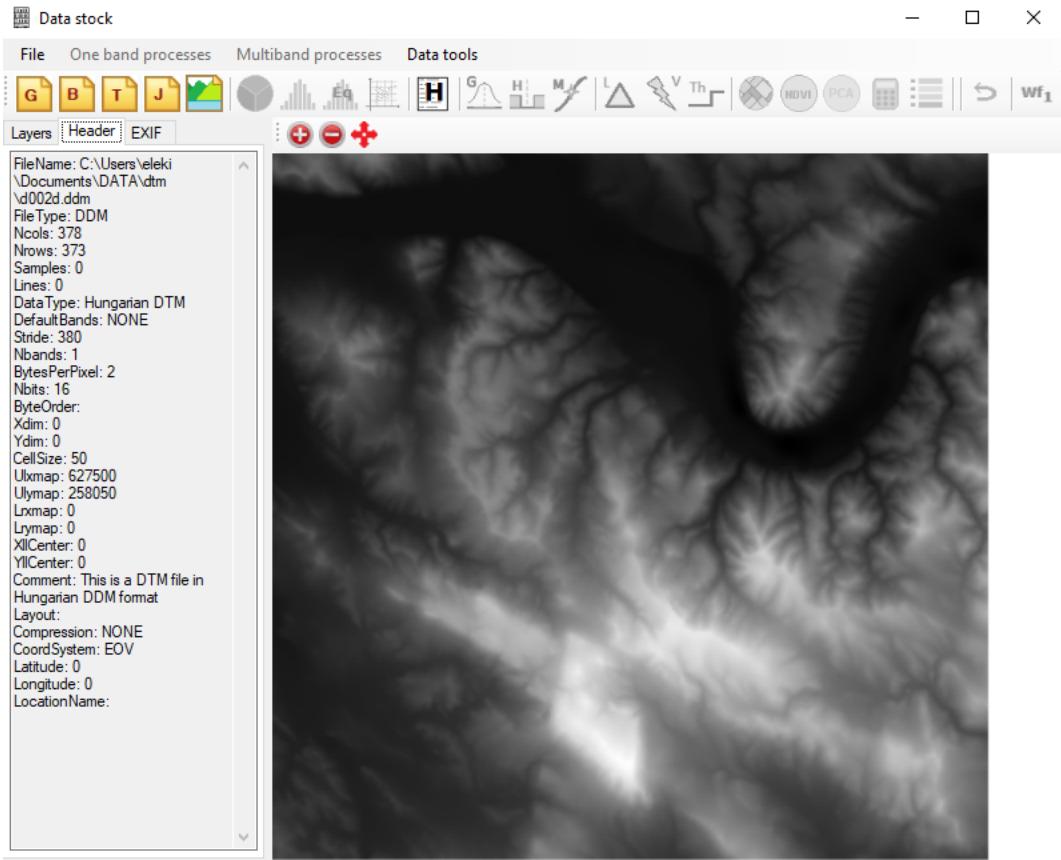


6. ábra. Egy képi példa *BIL* fájl adatstruktúrájára

képformátum, hanem a kép tényleges pixelértékeinek fájlban való tárolására szolgáló adatstruktúra. A *BIL* támogatja az egy- és többsávos képek tárolását legyen az fekete-fehér, a szürkeárnyalatos, a pszeudo color, true color vagy multispektrális esetleg hiperspektrális.

A *BIL* fájlhoz, amely bináris, tartozik egy header fájl is (ASCII fájl), amely alapján a *BIL*-ben található bináris adatok értelmezhetők. Kiegészítő adatokat tartalmaz a képről, mint például a képen látható sorok és oszlopok számát, a frekvenciasávok száma, színmélység (bits per pixel), sarokpont adatok földi koordináta rendszerben, bájt sorrend, egy pixel mekkora valós terület reprezentál, stb.

A *BIL* savonként tárolja a kép minden sorát. Például, ha egy háromsávos képünk van, akkor az 1. sor mindhárom sávját tároljuk az 1. sorban, majd a 2. sor mindhárom sávját, és így tovább, amíg el nem érjük a képen a sorok számát. A 5. ábra egy három sávos *bil* fájl adatait szemlélteti. Ha az ábrát sorfolytonosan nézzük, akkor megkapjuk a disken lévő *bil* hosszú sorvektorát.



7. ábra. A Dunakanyar digitális terepmodellje grayscale képként ábrázolva. A legalacsonyabb értékek feketével, a legmagasabbak fehérrel vannak jelölve

3.2.4. DDM

A *DDM* formátum a magyar digitális domborzatmodell bináris adatformátuma, amely sorfolytonosan tárolja a pontonkénti magasság adatokat. Egy magasságadat 2 byte. Az bináris adatok metaadatait egy header fájl tartalmazza textként, amelyből kiolvashatók az adatsor főbb paraméterei, mint pl. a sorok és oszlopok száma, a pixel fizikai mérete, a sarokpont koordináták, a koordináta rendszer, és még néhány apróság. A 7. ábrán a Dunakanyar digitális terepmodelljét láthatjuk.

3.2.5. Gwr

A *gwr* egy bináris fájlformátum, amely a frekvenciasávok pixelértékeit egy-egy különálló fájlban tárolja. Ehhez is tartozik egy header fájl (*gwh*), amely metaadatokat tartalmaz a képről, hasonlóan a *bil* headeréhez.

Name	Function	DataIn	DataOut
parseGiwerHeader	read Giwer header	File name (string)	properties
parseBilHeader	read Bil header	File name (string)	properties
parseTifHeader	read Tif	File name (string)	properties
parseJpgHeader	read jpg	File name (string)	properties
parseTifParams	read Tif	File name (string)	properties
parseGeoTifParams	read geoTif	File name (string)	properties
get Exif data	read Exif	File name (string)	properties
compute Bil stride	compute stride		properties

1. táblázat. A **GeoImageData** osztály főbb függvényei

3.3. Osztályok és függvények

A következőkben bemutatjuk, hogy mely osztályok, és mely függvények teszik lehetővé a fentebb vázlatosan bemutatott funkciionalitást:

3.3.1. A **GeoImageData** osztály

A **GeoImageData** osztály a képek header fájljainak beolvasására és parszolására hivatott. A fájlnév megadása révén a **GeoImageData** osztály tulajdonságai felveszik azt az értéket, amelyeket a fájlból kiolvasott. Az osztály főbb függvényei láthatók a 1. táblázatban.

3.3.2. A **GeoImageTools** osztály

A **GeoImageTools** számos eljárást tartalmat, amelyek a képek előfeldolgozásához, különféle konverziókhöz, manipulációkhöz szükségesek. Vázlatos leírásuk a 2. táblázatban láthatók.

3.3.3. A **StatMath** osztály

A **StatMath** osztály matematikai eljárásokat tartalmaz, amelyek szerepet kapnak a képfeldolgozás során. Ezekt láthatjuk a 3. táblázatban.

3.3.4. A **GeoFilters** osztály

A **GeoFilters** osztály különböző digitális szűrések függvényeit tartalmazza. A lista még távolról sem teljes. Egyelőre csak azok vannak benne, amelyek már működnek. Valamennyi függvény egyetlen, általunk kiválasztott frekvenciasávra működik, kivéve, amelyek kifejezetten RGB sávokra működnek. Vázlatos leírásuk a 4. táblázatban láthatók.

Name	Function	DataIn	DataOut
readGwrFile	read source data file	File name	byte[]
convertOneBandBytes toBitmap	convert image from a given band into bitmap	byte[]	bitmap
byteArray2Bitmap	convert bytarray to a bitmap	byte[]	bitmap
bitmap2ByteArray	convert bitmap to a bytarray	bitmap	byte[]
getAnRGBBand	get a band data from a bytarray	which band	byte[]
getOneBand2Bytes	get a band data from a bytarray	which band	byte[]
combine2Images	combine current image with another	byte1[], byte2[]	byte[]
vectorize	vectorize an image	byte[], params	byte[]
thresholding	thresholding	byte[], params	byte[]
computeHistogram	compute histogram of a bytarray	byte[]	float[]
HistoEqualization	histogram equalization	byte[]	byte[]
grayscale	grayscale conversion	byte[]	byte[]
invertColor	color inversion	byte[]	byte[]
convert2GiwerFormat	convert images (tif,jpg,bil) to Giwer format	File name	byte
saveGiwerFormat	save a bytarray to gwr file format	File name	File name
convertImagefromTif2Jpg	convert an image from tif to jpg	bitmap	bitmap
convertImagefromJpg2Tif	convert an image from jpg to tif	bitmap	bitmap

2. táblázat. A **GeoImageTools** osztály függvényei

Name	Function	DataIn	DataOut
getMinMax	compute image min, max values	int[] or double[]	string
imageAverage	compute image's mean	byte[]	float
imageScatter	compute image's scatter	byte[]	float
imageStandardization	compute standardized image	byte[], double, double	byte[]
compCorrelationMatrix	compute corr.matrix	list of bands	byte[]
compEigen	compute eigenvectors and eigenvalues	byte[]	Eigenvalues and vectors
PCA	compute the desired principal component	N * byte[], eigens	byte[]

3. táblázat. A **StatMath** osztály főbb függvényei

Name	Function	DataIn	DataOut
medianFilter1Band	median filtering on one band	byte[], kernel	byte[]
medianFilte3Band	median filtering on 3 bands	byte1[],byte2[],byte3[], kernel	byte[]
LaplaceOneBand	Laplace filtering on one band	byte[]	byte[]
Laplace3Bands	Laplace filtering on 3 bands	byte1[],byte2[],byte3[]	byte[]
convolSingleBand	convolution filtering on one band	byte[]	byte[]
convol3Bands	convolution filtering on 3 bands	byte1[],byte2[],byte3[]	byte[]
highPassKernel	Laplace filtering on one band	float	double[,]
lowPassKernel	Laplace filtering on 3 bands	float	double[,]
lowPassKernelGauss	convolution filtering on one band	float	double[,]
resampling	convolution filtering on 3 bands	int rate, bitmap	byte[]
gradientFilter	convolution filtering on 3 bands	byte[]	byte[]

4. táblázat. A **GeoFilters** osztály főbb függvényei

Name	Function	DataIn	DataOut
NDVI	computes NDVI	byte1[], byte2[]	byte[]
clustering	clustering	N * byte[] or PC1	byte[]
Segment	segmentation	N * byte[] or PC1	byte[]
createRGB_gwr	creates an RGB image from 3 bytarrays	byte1[],byte2[],byte3[]	bitmap
crossPlot	create cross-plot from 2 bands	byte1[], byte2[]	byte[]

5. táblázat. A **GeoMultibandMethods** osztály főbb függvényei

3.3.5. A GeoMultibandMethods osztály

A **GeoMultibandMethods** osztály olyan függvényeket tartalmaz, amelyek egynél több frekvenciasávra működnek. Vázlatos leírásuk a 5. táblázatban láthatók.

3.3.6. A DTM osztály

A DTM osztály digitális domborzati modellek (digital terrain modell) kezelését szolgálja. Segítségével raszteres adatszerkezetű magassági adatok olvashatók be, és jeleníthetők meg. Egyelőre több funkciót nem valósítottunk meg, de az adatok a Giwer saját formátumában rendelkezésre állnak további feldolgozó függvények számára.

3.3.7. A Raster calculator osztály

A raszter kalkulátor arra szolgál, hogy segítségével megadott feltételek szerint lekérdezést hajthassunk végre az aktuális képre. Ez egyrészt azt jelenti, hogy kiemelhetünk a képből olyan területeket, amelyet hangsúlyosabban kívánunk láttatni, másrészt a legyűjtés eredményeként létrejött képen tetszőleges további műveleteket hajthatunk végre.

3.3.8. Az ImageWindow osztály

Az **ImageWindow** osztály képek megjelenítésére szolgál. Nemcsak a bemenő képek szemlélhetünk általa, hanem bármely művelet eredményét is megnézhetjük. Vázlatos leírása a 6. táblázatban látható.

A fenti függvények biztosítják a képek zoomolását, de annak sokkal többet is tud az **ImageWindows** osztály. A betöltött kép fölött mozgatott kurzor pozícióját is megjeleníti, sőt a kép adott pozíciójának értékét is megmutatja. Ez többnyire egy 24 bites RGB érték, egy 8 bites grayscale kép, de lehet

Name	Function	DataIn	DataOut
loadImageFromFile	load an image from a file	file	Load and display image
clear	clear screen		empty screen
zoomIn	image zoom in	byte[]	zoomed image
zoomOut	image zoom out	byte[]	zoomed image
DrawImage	draw an image from a byte array	byte[]	
DrawImageRGB	draw an image from RGB byte arrays	byteR[],byteG[],byteB[]	
DrawImageDDM	draw elevation data from a byte array	byte[], float[,]	

6. táblázat. A **ImageWindow** osztály főbb függvényei

egy csoportba tartozási érték is. Ebben az esetben a kép megjelenését a lookup table határozza meg, amely meghatározza, hogy a 8 bites kép milyen színtábla (colortable) szerint jelenik meg.

Ha domborzati adatokat tartalmaz a kép, akkor a kurzos pozícióján kívül, az ehhez a pozícióhoz tartozó magasságértékeket is megmutatja. Ha a lookup table nem *defaultra* van állítva (greyscale), hanem *hypsometric*-re, akkor a 8 bites képet színesben fogjuk látni, ahol a színek a különböző magasságokat szimbolizálják.

3.4. A Project osztály

A drónok általában nagy tömegben hoznak létre képeket, emiatt célszerű őket projektbe szervezve feldolgozni. Olyankor is hasznos lehet a projekt használata, ha egyszerre több képpel foglalkozunk. Ilyenkor a projekt nevével hivatkozhatunk arra a gyűjteményre, amelybe összegyűjtöttük a használni kívánt képeket.

3.5. A Mosaic osztály

Amikor sok drón kép készül egy adott területről, akkor ezek a képek valamikorra átfedéssel (általában 50-75 %-os átfedés) képezik le a kívánt terület. Ezek megjelenítése különösen problematikus. Nem járható az az út, hogy egyetlen hatalmas képpé ragasszuk össze a képeket, ezért meg kell maradjanak a képek önállónak, de egyszerre kell tudni megjeleníteni őket, sőt a feldolgozási folyamatokat is egyszerre kell futtassuk.

4. Függvények és eljárások bemutatása

Ebben a részben röviden bemutatjuk a **Giwer** rendszerbe beépített eljárások és függvények működését, elvi hátterét.

4.1. A **GeoImageData** osztály eljárásai

Ez az osztály arra hivatott, hogy támogatott kéatformátumok metaadatait beolvassa és tárolja. Egységesen, minden képhez ugyanazokat a property-keket, ugyanolyan néven hozza létre, függetlenül attól, hogy eredetileg milyen név volt a forrás header fájlban. A következő eljárásai vannak: parseGiwerHeader, parseTifHeader, parseBilHeader, parseJpgHeader, parseExif. Ennek jelentősége abban van, hogy ha később bővülne a támogatott fájl formátumok száma (pl. ENVI headert is kezelní szeretnénk), akkor emiatt nem kell módosítanunk a többi osztály működésén, mert minden művelet a **GeoImageData** osztályt használja.

4.1.1. parseBilHeader

Ez a függvény beolvassa, és betölti a *bil* fájl headerjének tartalmát a **GeoImageData** osztály propertyjeibe. Mivel sokféle bil fájl létezik, ezért nem minden property kap értéket, de valamennyi kiszámítható valamely másikból. Ezeket a számításokat a Giwer elvégzi.

4.1.2. parseTifHeader

Ez a függvény beolvassa a *tif* fájl metaadatait, és betölti a **GeoImageData** osztály propertyjeibe. Külön függvény végzi a *geotif* formátumú képek paramétereinek beolvasását (parseGeoTifParams), amelyhez az open source gdal library-t is felhasználtuk. A drónok által szolgáltatott képek nem georeferált fotrmátumkat használnak, ezért ezeket külön erre a célra kifejlesztett eljárással olvassuk be (parseTifParams és parseJpgParams).

4.1.3. parseGeoTifHeader

Mivel a hagyományos *Tif* fájl nem tartalmaz georeferencia adatokat, ezért szükséges a GeoTif formátumra külön parsolót készíteni. Noha a pixelek olvasása hasonlóan történhet, mint a hagyományos Tif fájlok esetében, de a fájl headerben további beolvasandó adatok vannak (pl. Datum, szferoid adatok, vetületi rendszer, a vetületi egyenletek paraméterei, stb.), amelyeknek az ismerete szükséges további feldolgozások számára. Ez a függvény ezeket az adatokat beölti a **GeoImageData** osztály propertyjeibe.

4.1.4. parseJpgHeader

Ez a függvény beolvassa a *jpg* fájl metaadatait, és betölti a **GeoImageData** osztály propertyjeibe. Nem minden property kap értéket, hiszen például a drónok által alkotott képek nem georeferáltak, így koordináta információt nem tartalmaznak. Ezeket más módon kell megállapítani, és betölteni a **GeoImageData** osztály propertyjeibe.

4.1.5. getExif

Ez a függvény a *tif* és *jpg* formátumú fájlok exif adatait olvassa be. Ezek között szerepelhetnek georeferencia adatok is, ha *geotif* vagy *geojpg* a fájlok formátuma. Ezenkívül az exponálással kapcsolatos adatok is (expozíciós idő, blende nyílás, kamera típus, stb.) szerepel az adatok között.

4.1.6. parseGiwerHeader

Ez a függvény a *gwh* formátumú header fájl adatait olvassa be, és tölti be a **GeoImageData** osztály propertyjeibe. Ez a fájl csak akkor létezik, ha előzőleg valamely más formátumú adatot (*bil*, *tif*, *jpg*) már átkonvertáltunk *gwr* formátumba.

4.2. A GeoImageTools osztály eljárásai

Ebben a fejezetben áttekintjük a **GeoImageTools** osztály eljárásait.

4.2.1. saveGiwerFormat

Ez a függvény elmenti valamely forrásfájl képét *gwr* formátumban. Megadható a mentendő fájl neve, és helye. A megadott néven létrehoz egy header fajlt (*gwh*), amely a kép metaadatait tartalmazza, valamint egy a kiterjesztés nélküli fájl néven egy könyvtárat, ahová a bináris adatok bekerülnek 0,1,2 ... fájlnéven, annyiadik számon, ahányadik csatorna képét tartalmazza.

4.2.2. saveHeader2Giwer

Ez a függvény a metaadatokat giwer header (*gwh*) formátumban menti el.

4.2.3. convertByteArray2GiwerFormat

Ez a függvény az adott byte tömb tartalmát menti el giwer formátumban, a megadott helyen és néven.

4.2.4. convertOneBandBytesToBitmap

Ez a függvény egy byte tömb adataiból egy bitmapet készít. Mivel egyetlen sávról van szó, a bitmap egy szürke kép lesz, mivel a bitmap minden színcsatornájában ugyanannak a byte tömbnek az adatai vannak.

4.2.5. getOneBandBytes

Ez a függvény kiolvassa egy csatorna byte adatait és egy byte tömbbe tölti.

4.2.6. getOneBandToArrayFromBitmap

Ez a függvény egy bitmapból kiolvassa a megadott színsáv adatait és egy byte tömbbe tölti.

4.2.7. BitmapToByteArray

Ez a függvény egy adott bitmapet három színsávra bont, és ezeket egy-egy byte tömbbe tölti.

4.2.8. ByteArrayToBitmap

Ez a függvény egy vagy három byte tömbben tárolt kép adataiból egy bitmapet számol.

4.2.9. convertImageFromTif2Jpg

Ez a függvény tif-ből jpg-be konvertálja a megadott fájlt.

4.2.10. convertImageFromJpg2Tif

Ez a függvény jpg-ből tif-be konvertálja a megadott fájlt.

4.2.11. InvertColor

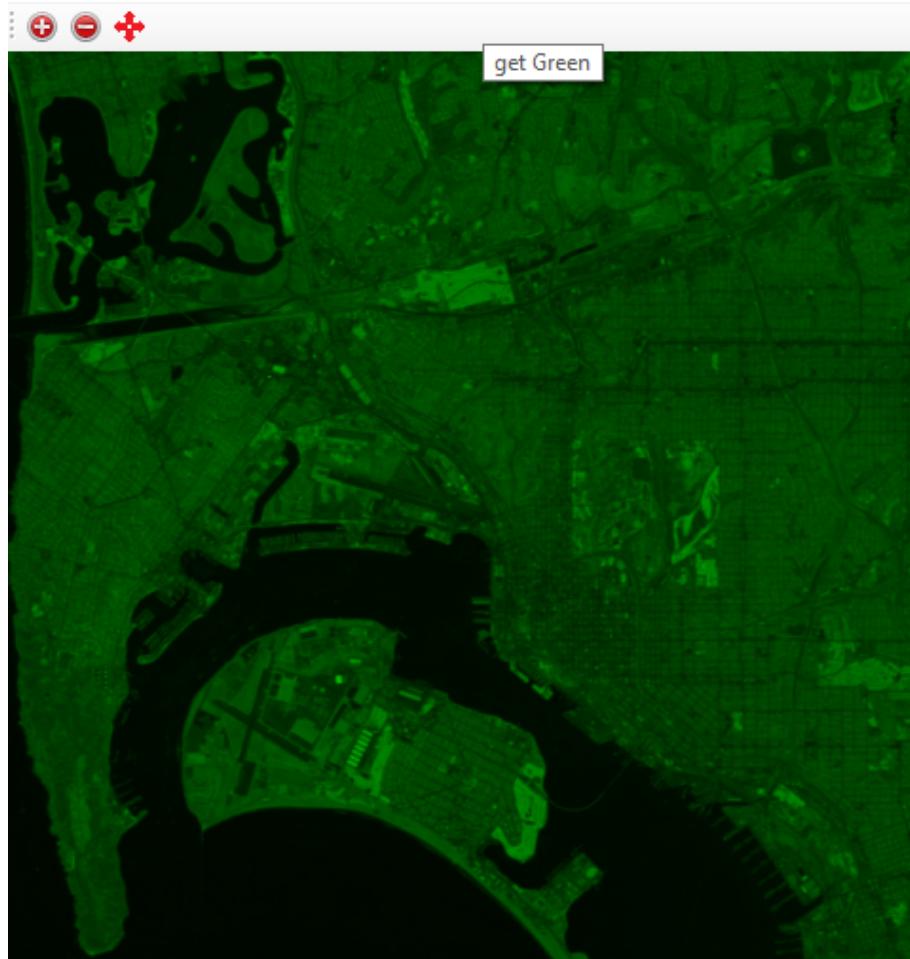
Ez a függvény invertálja az adott bitmap képét

4.2.12. GrayscaleConversion

Ez a függvény egy bitmapet szürkévé alakít.

4.2.13. readGwrFile

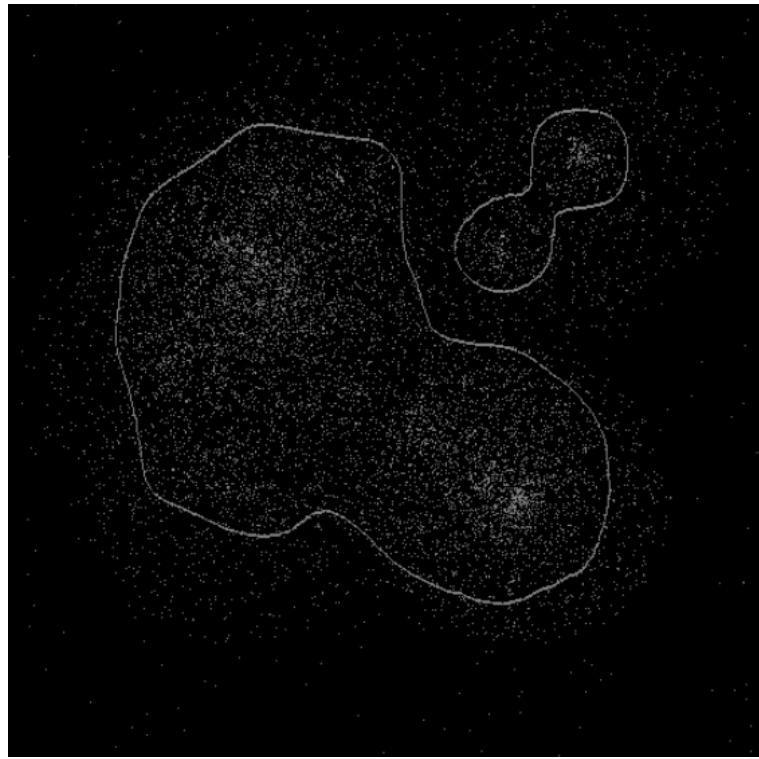
Ez a függvény beolvas egy giwer formátumú fájlt.



8. ábra. A zöld sáv megmutatása

4.2.14. GetAnRGBBand

A képek feldolgozása szempontjából nem alapvető jelentőségű, de vizualizációs szempontból hasznos lehet, ha meg tudjuk mutatni egy-egy kép RGB színsávjait az alapszínekben ábrázolva. Erre szolgál a getAnRGBBand(WhichBand) függvény, ahol meg kell adnunk, hogy mely RGB sávot szeretnénk látni (whichBand). Eredménye egy bitmap, amelyet megmutat az ImageWindow (8. ábra). További számítások az eredménnyel nem végezhetők.



9. ábra. Két kép kombinációja *EXOR* művelettel. Az egyik kép egy RGB image a pontfelhőről, a másik pedig a pontfelhő határait mutatja

4.2.15. `combine2Images`

Ez függvény az aktuális képet kombinálja egy tetszőleges másikkal. Egyelőre három művelet készült el: az összeadás, a kivonás és a kizáró vagy (EXOR) művelet. A '+' és a '-' művelet funkciója nyilvánvaló. Az 'exor'-t olyankor használjuk, amikor az egyik képet úgy kombináljuk egymással, hogy bizonyos feltételek teljesülése esetén vagy csak az egyik vagy csak másik legyen a kombinált kép tartalma (9. ábra).

4.3. A StatMath osztály eljárásai

4.3.1. `getMinMax`

Ez a függvény kiszámítja egy byte tömbben tárolt kép minimális és maximális intenzitásértékét.

4.3.2. imageAverage

Ez a függvény kiszámítja egy byte tömbben tárolt frekvenciasáv intenzitásértékeinek átlagát.

4.3.3. imageScatter

Ez a függvény kiszámítja egy byte tömbben tárolt frekvenciasáv intenzitásértékeinek szórását.

4.3.4. imagesStandardization

Ez a függvény standardizál egy byte tömbben tárolt képet

4.3.5. computeCorrelationMatrix

Ez a függvény kiszámítja tetszőleges számú, byte tömbben tárolt képek korrelációs mátrixát

4.3.6. compEigen

Ez a függvény kiszámítja egy mátrix sajátértékeit és sajátvektorait

4.3.7. PCA

Ez a függvény kiszámítja tetszőleges számú, byte tömbben tárolt kép bármely főkomponensét

4.4. A GeoFilters osztály eljárásai

A **GeoFilters** osztály olyan eljárásokat tartalmaz, amelyek egy kiválasztott frekvenciasávra vonatkozó szűréseket végeznek.

4.4.1. MedianFilter3Bands

Ez a függvény mediánszűrést végez három megadott frekvenciasáv intenzitásértékeit tartalmaz byte tömbökre. A kernel hosszát bemenő paraméterként kell megadni.

4.4.2. MedianFilterOneBand

Ez a függvény mediánszűrést végez egy megadott frekvenciasáv intenzitásértékeit tartalmazó byte tömbre. A kernel hosszát bemenő paraméterként kell megadni.

4.4.3. Canny

Ez a függvény éldetektálást végez egy megadott frekvenciasáv intenzitásértékeit tartalmaz byte tömbre a Canny-féle eljárás alapján.

4.4.4. Laplace3Bands

Ez a függvény éldetektálást végez három megadott frekvenciasáv intenzitásértékeit tartalmazó byte tömbökre a Laplace-féle eljárás alapján.

4.4.5. LaplaceOneBand

Ez a függvény éldetektálást végez egy megadott frekvenciasáv intenzitásértékeit tartalmazó byte tömbre a Laplace-féle eljárás alapján (10. ábra).

Itt kell megjegyezni, hogy mivel a Laplace-szűrés egy második deriválton alapuló szűrő, ezért érzékeny a zajokra, így érdemes előtte valamilyen simító szűrést alkalmazni (Gauss-simítás vagy medián szűrő), és csak azután végezni el a Laplace szűrést. Ennek eredményét mutatja a 11. ábra.

4.4.6. ConvolveSingleBand

Ez a függvény konvolúciós szűrést végez egy megadott frekvenciasáv intenzitásértékeit tartalmaz byte tömbre.

4.4.7. Convolve3bands

Ez a függvény konvolúciós szűrést végez három megadott frekvenciasáv intenzitásértékeit tartalmaz byte tömbökre.

4.4.8. highPassKernel

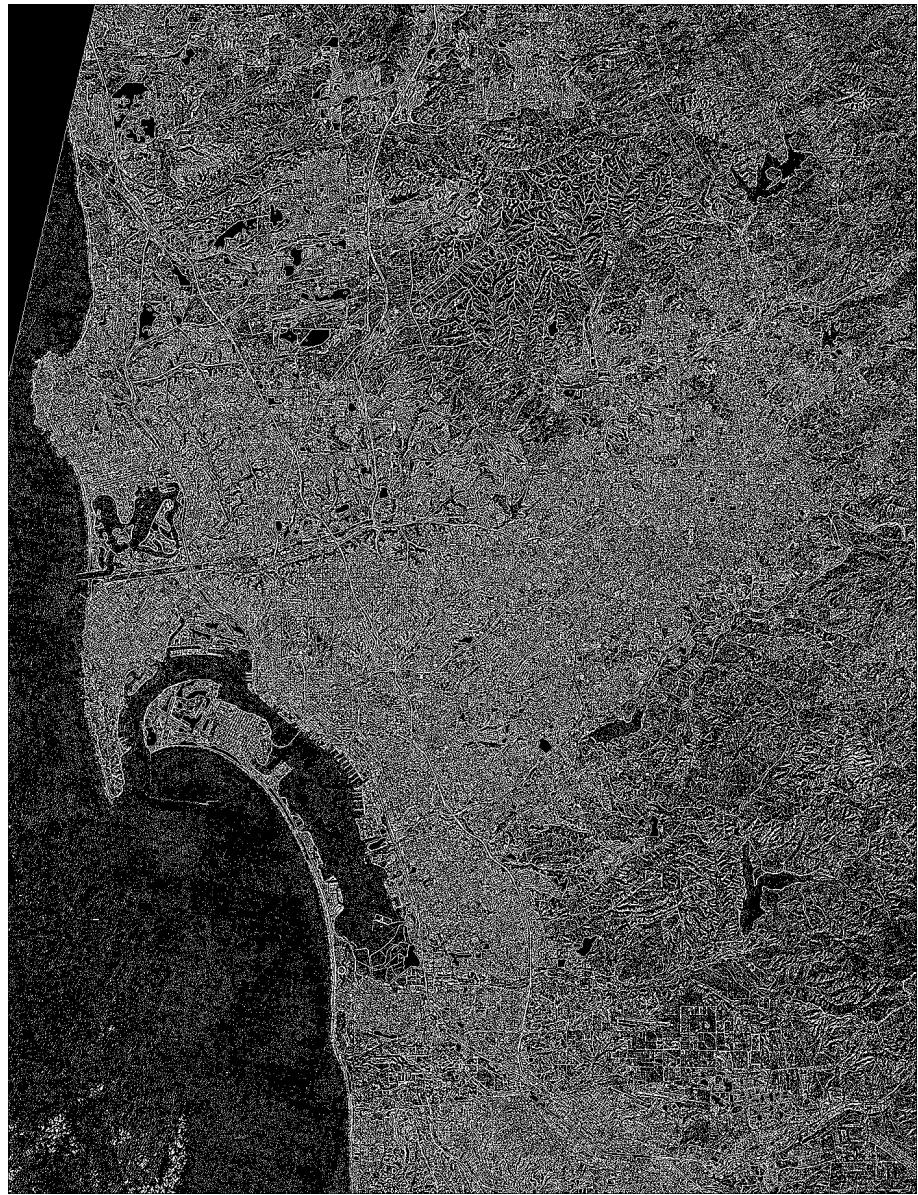
Ez a függvény felül áteresztő kernelt számol.

4.4.9. lowPassKernelGauss

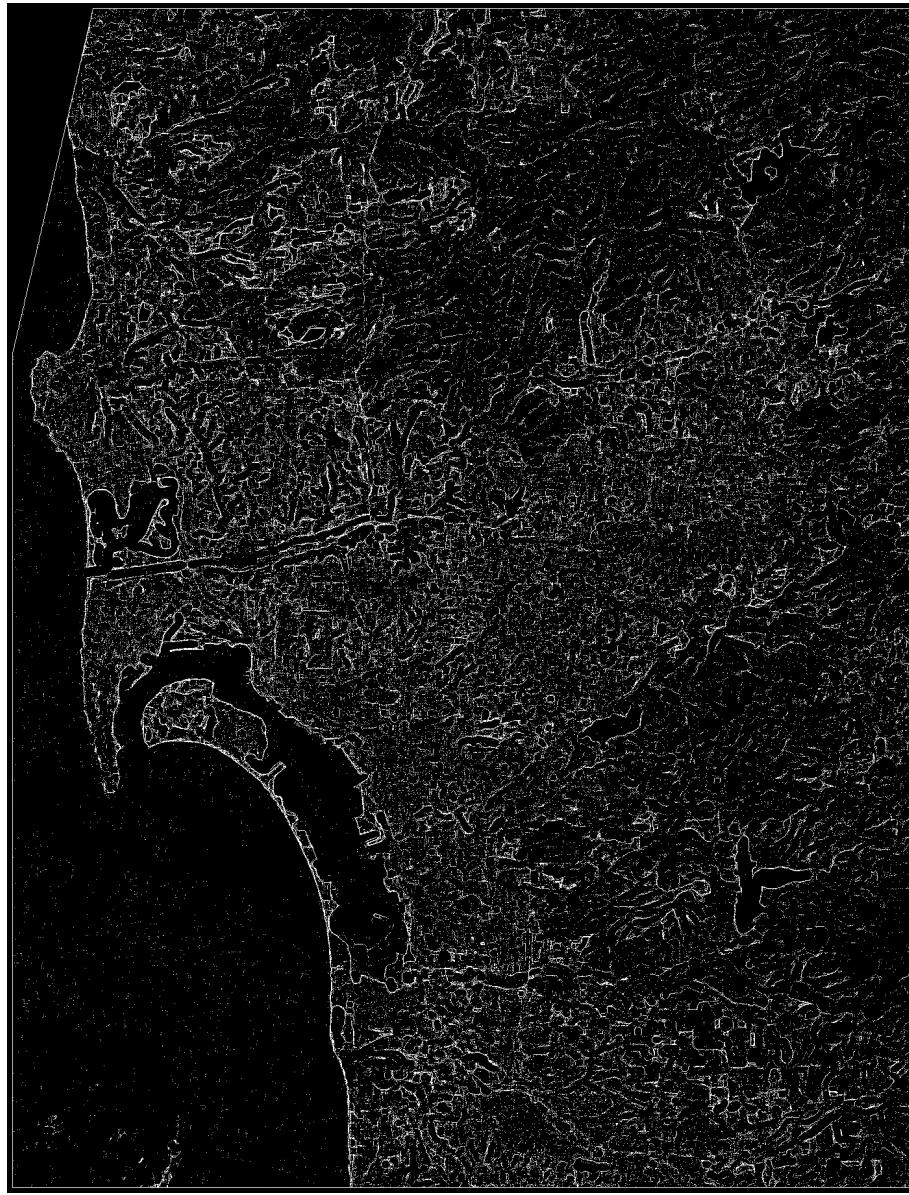
Ez a függvény alul áteresztő kernelt számol.

4.4.10. lowPassKernelBox

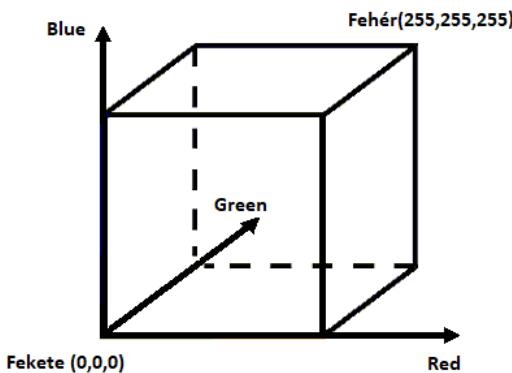
Ez a függvény alul áteresztő kernelt számol a box szűrőhöz.



10. ábra. A Laplace-szűrés eredménye az egyik frekvenciasávra



11. ábra. A Laplace-szűrés eredménye egy medián szűrt úrfelvételre. A medián szűrés jelentősen mértékben eltüntette a zajokat, így az élek már sokkal világosabban látszanak



12. ábra. Az RGB színek kocka

4.4.11. resampling

Ez a függvény átmintavételezi az adott képet egy megadott mintavételi távolság alapján.

4.4.12. Prewitt

Ez a függvény Prewitt-féle éldetektálást végez.

4.4.13. Sobel

Ez a függvény Sobel-féle éldetektálást végez.

4.4.14. Isotropic

Ez a függvény Izotropikus éldetektálást végez.

4.4.15. vectorize

Ez a függvény raszteres kép vektorizálását végzi.

4.5. GeoMultibandMethods osztály eljárásai

4.5.1. RGB

Képzeljünk el egy háromdimenziós koordináta rendszert (12. ábra), ahol a három tengely a három alapszínnek felel meg.

Az RGB színmodell a színeket a vörös (Red), a zöld (Green) és a kék (Blue) alapszínek lineáris kombinációjaként állítja elő a következő módon:

$$Color = a \cdot Red + b \cdot Green + c \cdot Blue \quad (1)$$

ahol a, b, c együtthatók az egyes alapszínek részaránya az adott színben. Alkalmazzuk erre az esetre a 24 bites színmodellt. 2^{24} féle színárnyalatot tudunk megjeleníteni, ami azt jelenti, hogy 2^8 vörös, 2^8 zöld és 2^8 kék árnyalat jeleníthető meg, vagyis alapszínenként 256 fényerősség érték adható meg.

Egy adott frekvenciasáv intenzitás értékeit egy byte tömbben tároljuk, és egy RGB kép egy bitmappel reprezentálható, így három byte tömb alapján a bitmap kiszámolható, ha szükséges, meg is jeleníthető.

Definíciója a következő:

```
Bitmap createRGB(byte[] bytelnR, byte[] bytelnG, byte[] bytelnB, int width,
int height)
```

4.5.2. NDVI

Az NDVI (Normalized Difference Vegetation Index) szavakból alkotott mozaikszó, amely egy adott terület vegetációs aktivitását fejezi ki. A növényzet által visszavert fény intenzitásából számolható ki a következő módon: a közeli infravörös (NIR) és a látható vörös (RED) intenzitásainak különbsége és ezek összegének hányadosa szolgáltatja.

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

A NDVI szoros összefüggést mutat a területet fedő növényzet fajlagos klorofill tartalmával. Így az egészséges növény más NDVI értéket mutat, mint a beteg vagy már halott (lásd 13. és 14. ábrák). Definíciója a következő:

```
byte[] NDVI(byte[] bNIR, byte[] bIR)
```

4.5.3. Cross Plot

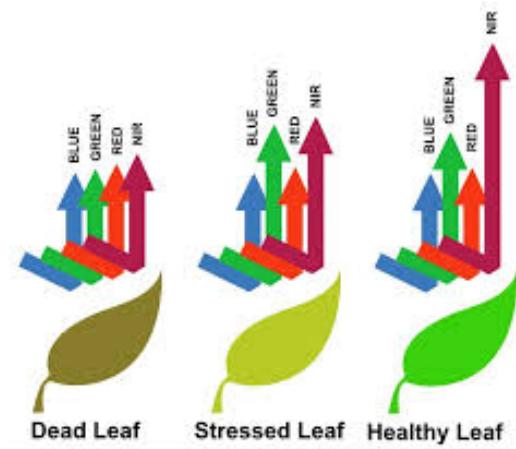
Az a függvény arra szolgál, hogy összehasonlítsuk két tetszőleges frekvenciasáv intenzitás értékeit. Az X-tengelyen az egyik, az Y-tengelyen a másik frekvenciasáv értékeit jelenítjük meg, ahogy azt a 16. ábrán láthatjuk.

4.5.4. compEigen

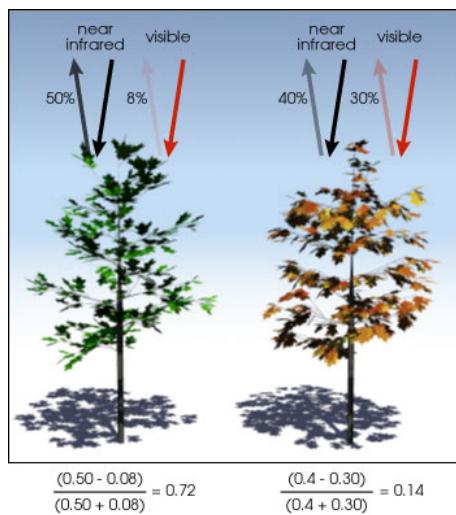
Kiszámítja egy megadott mátrix sajátértékeit és sajátvektorait.

4.5.5. compPCA

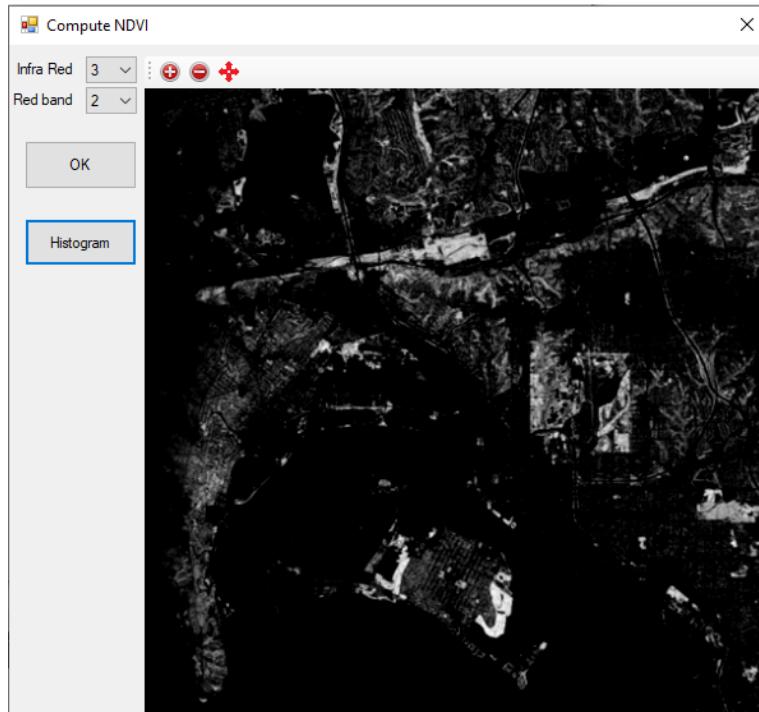
Ez a függvény kiszámítja a megadott byte tömbök által tárolt frekvenciasávok főkomponenseit. A 17. ábrán egy Landsat kép hét frekvenciasávjából számított első főkomponens látható.



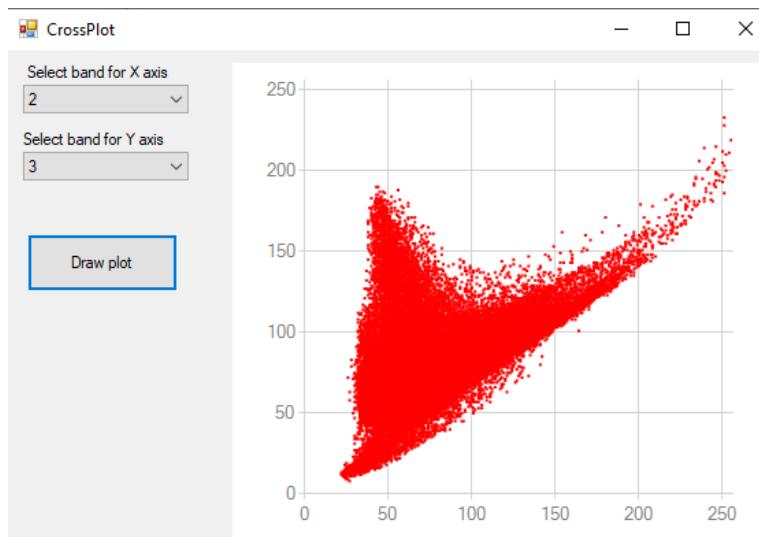
13. ábra. A különböző hullámhosszú fénysugarak visszaverődése különböző állapotú növényekről



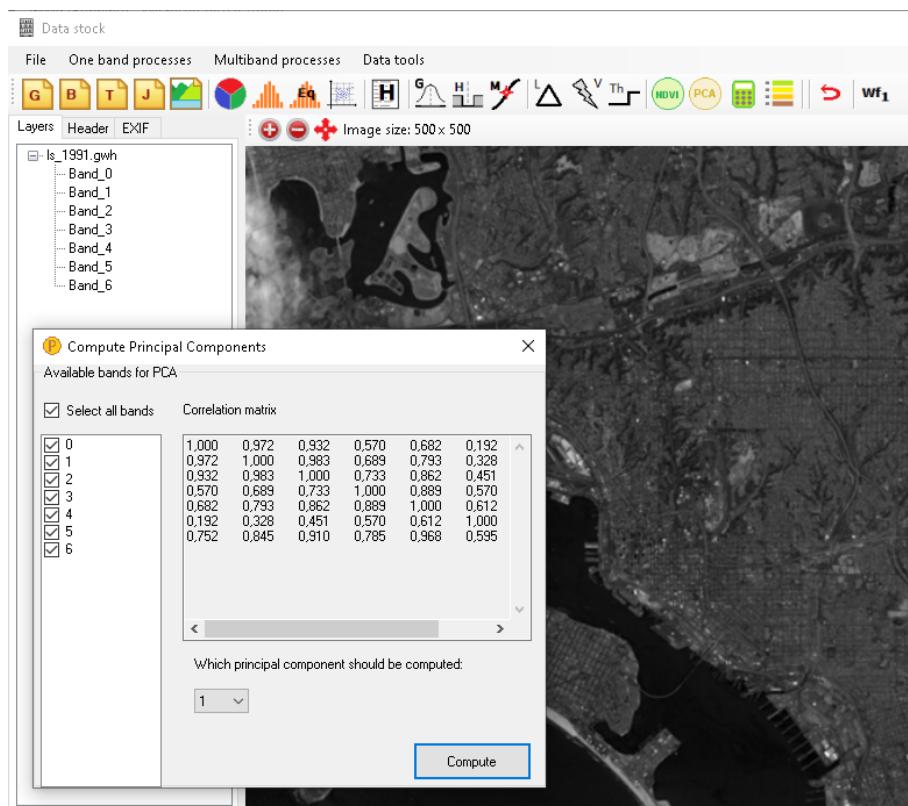
14. ábra. Az egészséges és a beteg növény viselkedése



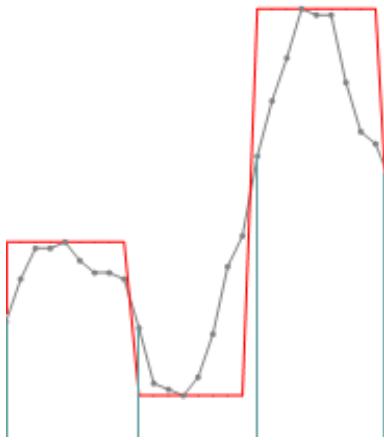
15. ábra. Egy példa az NDVI-ra. A sötét területek a nem növényekkel fedett területeket mutatják (épületek, kopár talajfelszín, víz, stb.), míg a világos területek a növényekkel fedett részeket. Minél világosabb egy terület annál dúsabb rajta aa vegetáció



16. ábra. Egy példa két különböző frekvenciasáv intenzitásainak összehasonlítására



17. ábra. Példa egy hét frekvenciasávból álló Landsat kép első főkomponensére



18. ábra. Az ábrán egy szintetikus adatsor (szürke görbe) intenzitásváltozásait láthatjuk egy dimenziós esetben. A szegmens határok ott vannak, ahol két csúcs között a legnagyobb az intenzitásváltozás. A kék vonal a szegmensek határait mutatja, a piros görbe pedig a szegmensekre kiszámított intenzitásértékeket. A szürke görbén lévő pontok az adatpontokat jelölik.

4.5.6. Clustering

Az klaszterezési/osztályozási eljárások számos változata létezik. A *Giwer* rendszer saját osztályozó eljárást fog alkalmazni, amely előzetesen végrehajtott szegmentálás eredményén fog dolgozni. Az osztályozás a szegmensekre kiszámított értékekre fog történni.

4.5.7. Segmentation

Számos szegmentálás eljárás létezik. A *Giwer* rendszer egy éldetektáláson alapuló szegmentálási eljárást fog használni, amely elsődleges élek detektálása után, azok további szelekciójával fog előállítani szegmenshatárokat. A szegmensek értékei ezeken a határokon belüli pixelek értékeit alapján lesznek kiszámítva.

A következőkben ismertetjük az eljárás elvi alapjait. Szegmenshatár ott van, ahol az intenzitás értékében markáns változás van. Tekintsük a legnagyobb intenzitásváltozás helyét a szegmens határának. Erre látunk példát a 53. ábrán. Az a kék vonal azokat helyeket mutatja, ahol az eljárás szegmenshatárt detektált.

Nemcsak a szegmenshatárok érdekesek a számunkra, hanem az intenzitásértékek is, amelyeket a szegmensekhez hozzárendelünk. Egy szegmensen belül homogénné tesszük az intenzitásokat, vagyis egy a szegmensre jellemző intenzitásértéket rendelünk hozzá. Jelen esetben azt a logikát követtük, hogy legyen a szegmens értéke a két szegmenshatár közötti értékek extrema, ha ezek a szegmenshatárokon belülre esnek. Ha az extremumok a széleken vannak, akkor legyen a jellemző érték a határok közötti értékek átlaga.

A 19. ábrán egy SPOT kép szegmenshatárait láthatjuk. Ezek a határok a fent ismertetett inflexiós pont keresés alapján lettek megállapítva.

4.6. A DTM osztály

A digitális domborzatmodellre (DTM: digital terrain model) egy külön osztályt hoztunk létre, mivel az adatok természete jelentősen eltér a képekétől. minden DTM két byteon tárol egy magasság adatot, ugyanis egy byte 256 értéket képes csak tárolni, ami a magassághoz nem elegendő, mivel a Földön a magasságértékek -200 – 8848 m között változnak. Két belső eljárása van, és egy propertyje, amely egy kétdimenziós tömb a magasságadatokkal.

A domborzat megjelenítésben is fontos, hogy ne csak greyscaleben, hanem valamely konvencionális megjelenési stílusban (pl. hipszometrikusan) is meg tudjuk jeleníteni az adatokat.

4.6.1. `readParameters`

Ez az eljárás kiolvassa a DDM header fájljának tartalmát, és betölti a megfelelő propertykbe. A fájlnév megadásakor hívódik meg.

4.6.2. `readDDM`

Ez az eljárás beolvassa az adatsor paraméterivel leírt magasság adatokat egy kétdimenziós tömbbe, amelyet tárol a megfelelő propertybe. A `readParameters` eljárás után hívódik meg.

4.7. A Raster Calculator osztály

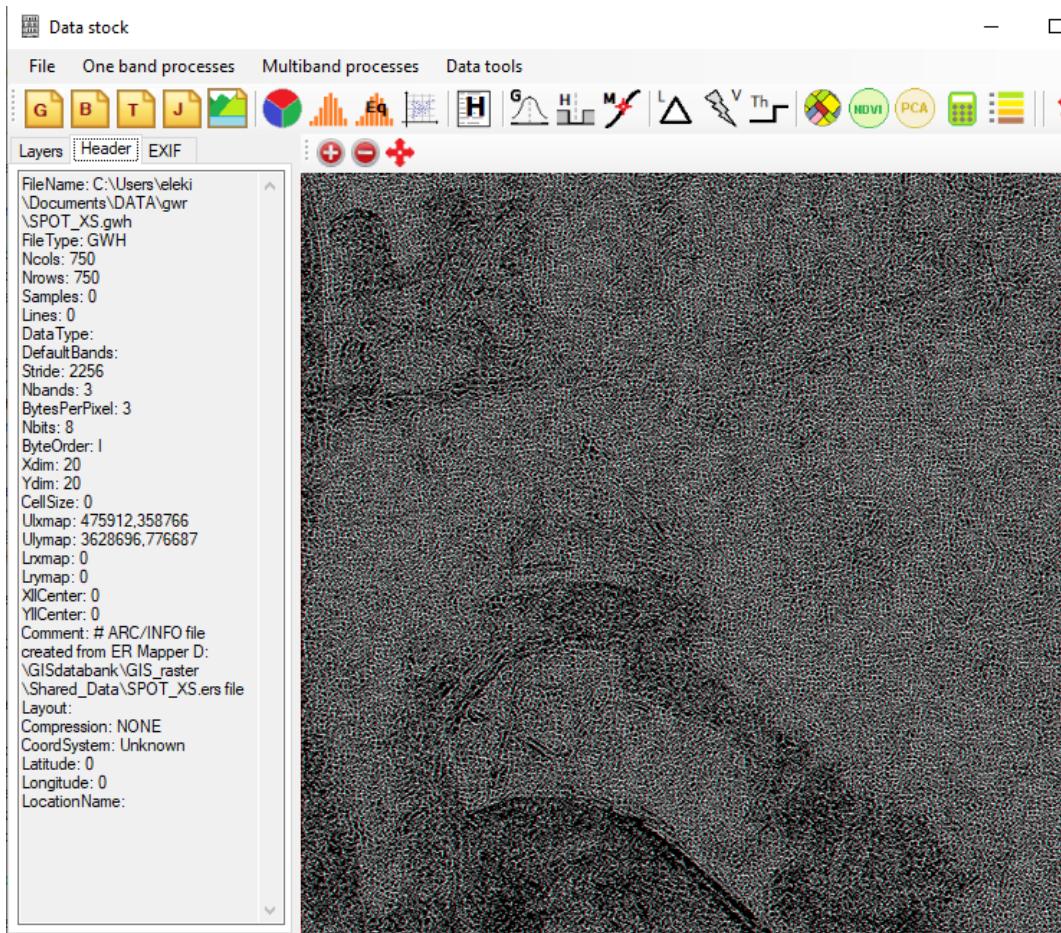
A **Raszter calculator** osztálynak egyelőre két fő funkciója van, amelynek paramétereit a felhasználó állíthatja be, amint a 21. ábrán látható.

- Where feltételeként egy értéket adhatunk meg összehasonlításra
- Where feltételeként két értéket (between) adhatunk meg összehasonlításra

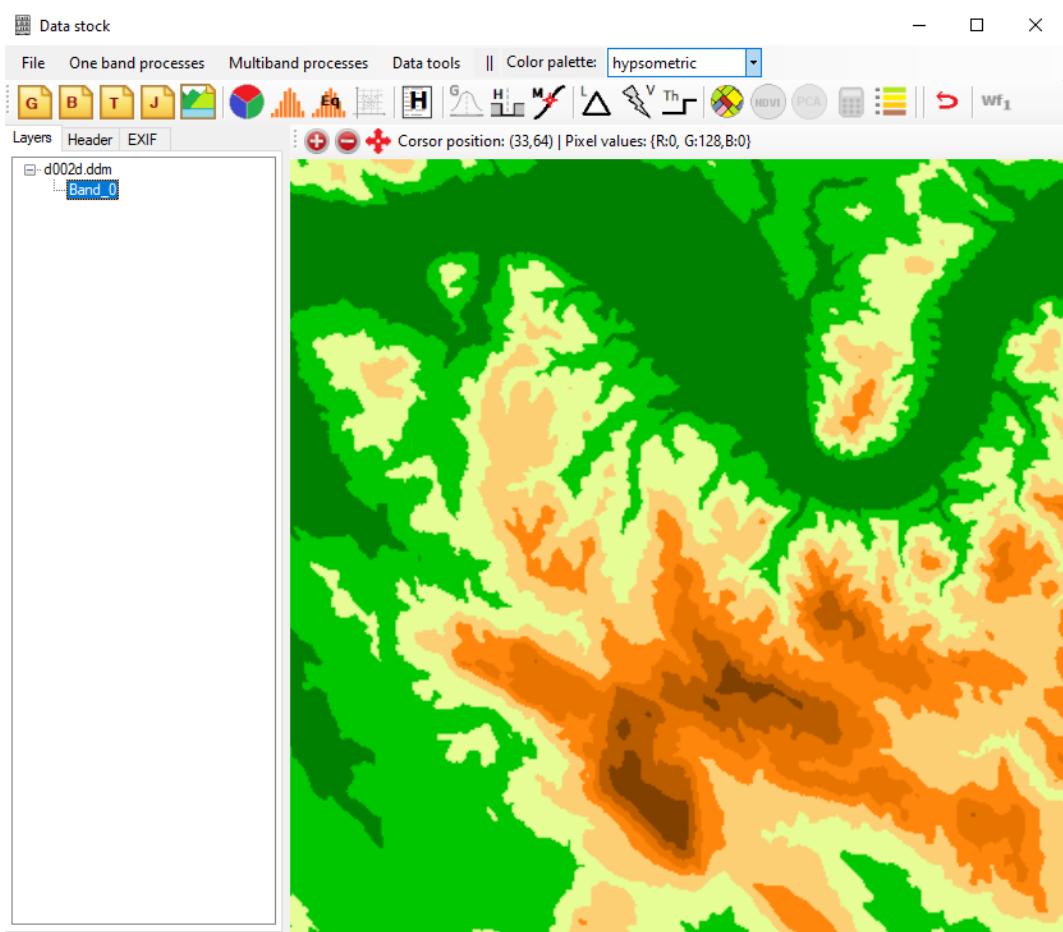
4.8. A Project osztály

A **project** osztálynak egy eljárása van, a *loadProjectFile*, amely betölti projectben lévő képek fájljait a *FileNames* nevű listába. A későbbiekben ez a gyűjtemény akkor is jól használható lesz, ha ugyanazt az eljárást, vagy egy workflowt kívánunk lefuttatni a projekt összes fájljára.

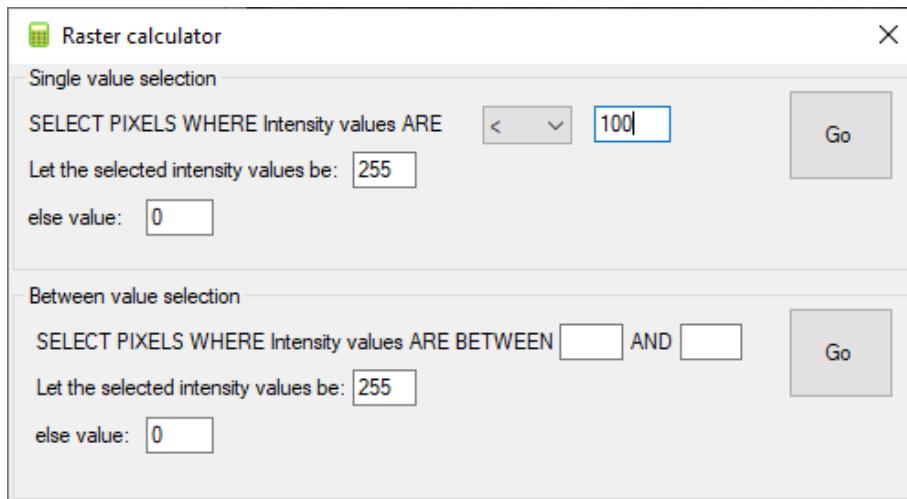
Különösen akkor válik ez igen hasznossá, ha sok drón felvételből kívánunk egy mozaik képet létrehozni.



19. ábra. Az ábrán egy SPOT kép elemi szegmenshatárait láthatjuk. A kép azért furcsa, mert az eredeti képet EXOR műveletteset kombináltuk össze a szegmenshatárokat tartalmazó képpel, vagyis, ahol nincs szegmenshatár, ott az eredeti képet láthatjuk, ahol viszont van határ, ott a határt láthatjuk



20. ábra. A Dunakanyar hipszometrikusan megjelenített képe. A zöld szín a legalacsonyabb, a sötétbarna a legmagasabb területeket mutatja



21. ábra. Az ábrán a **Raster calculator** kommunikációs felületét láthatjuk

5. A DataStock alrendszer használata

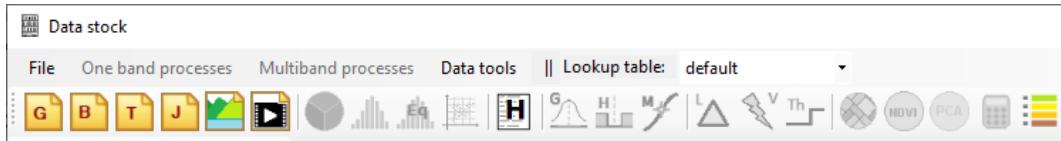
A **Giwer** rendszer interaktív modulja a **DataStock**, ami a menürendszeren keresztül teszi elérhetővé a függvénykönyvtárat és az adatokat. Különböző típusú grafikus fájlok (*bil*, *tif*, *jpg*) beolvasását és manipulálását végzi. Speciális, saját fájlformátuma a *GWR/GWH* formátum. A *GWH* egy header fájl, amely az adott kép metaadatait tartalmazza, míg a *GWR* egy bináris formátum, amely egységesen kezelhetővé teszi a legkülönbözőbb forrásokból származó adatokat, és lényegesen gyorsabbá teszi a feldolgozási műveleteket.

5.1. Menürendszer

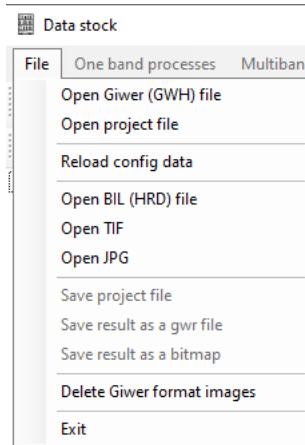
A menürendszer (22. ábra) *File*, *One band processes*, *Multiband processes* és *Data tools* menüelemekből áll, valamint jelzi az aktuális *Lookup table*-t mutatja (default, hypsometric, ndvi, user stb), amelyet szükség szerint átállíthatunk egy másikra. A *default* beállítás greyscale megjelenítést végez. A *hypsometric* egy 8 bites, maximum 256 színű színezést tesz lehetővé attól függően, hogy a *lookup table*-ben hány színt állítottunk be. Ez a hagyományos hipszometrikus megjelenítés stílusa, amely az alacsony területeket a zöld árnyalataival, a kissé magasabb területek a sárga árnyalataival, és a magas területeket a barna árnyalataival jeleníti meg. Az *ndvi* az ndvi számításkor használatos színes megjelenítést teszi lehetővé. A *user* a felhasználó által beállított *lookup table*-t állítja be.

5.1.1. File

A *File* menüvel (23. ábra) meg tudunk nyitni *gwh*, *bil*, *tif*, *jpg* típusú fájlokat, valamint projekt fájlokat, amelyek egyszerre több kép kezelésére szolgálnak. Tö-



22. ábra. A **datastock** fő menüje



23. ábra. A **File** menü

rölhetjük valamely (akár több) *gwh* fájlt. A *gwh*, *gwr* fájlok a **Giwer** rendszer saját formátumú fájljai. Elmenthetjük egy feldolgozás eredményét giwer formátumban, vagy egyszerű bitmapként. Elmenthetjük továbbá projektként is a **Giwer**-ben lévő adatoknak azt az állapotát, ami egy adott pillanatban éppen fennáll. Végül pedig ismételten betölthetjük a rendszer konfigurációs adatait, ha időközben megváltoztattuk azt a keretprogrammal (nem frissül automatikusan).

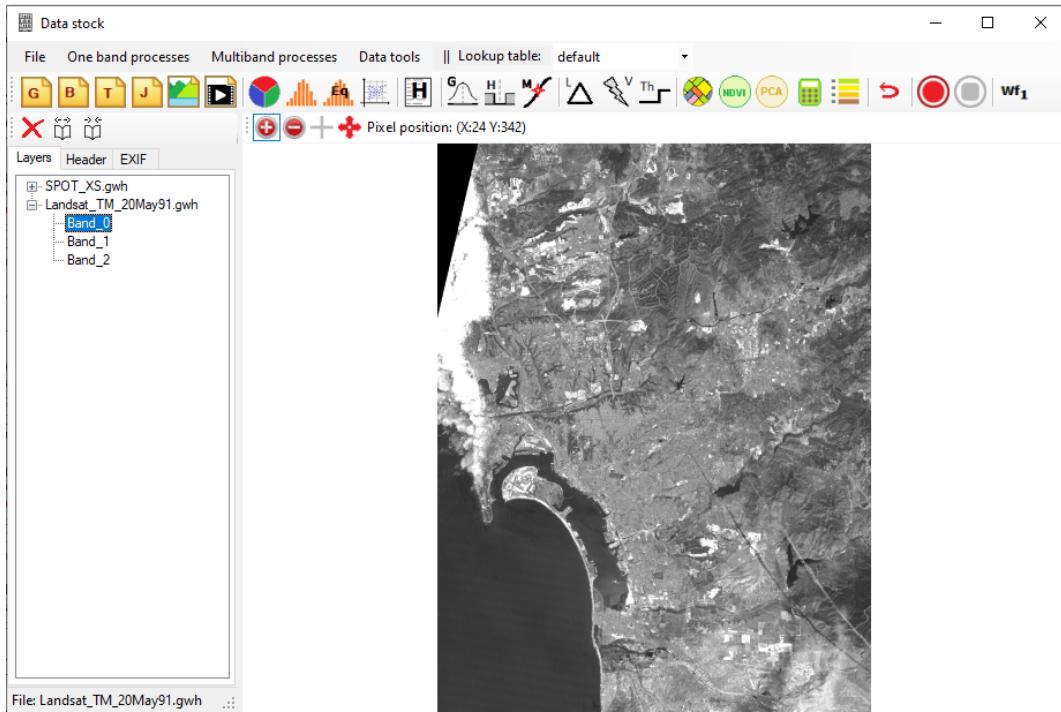
A képek beolvasása még nem jelent megjelenítést. Ehhez ki kell választanunk a megjeleníteni kívánt frekvenciasávot a *Data stock* ablak *Layers* fülén lévő valamelyik listaelemre kattintással (24. ábra). A kiválasztás után a legtöbb menüelem és gyors gombok (főmenü alatti ikonok) aktív állapotba kerülnek.

A *gwh* egy text fájl, amely header típusú adatokat tartalmaz a képről (szélesség, magasság, frekvenciasávok száma, bitmélység, stb.), míg a *gwr* egy bináris fájl, amely pixeladatokat tartalmaz sorfolytonosan.

A *bil* fájl egy régi ūrfelvétel formátum, amely szintén header fájlból és egy bináris fájlból áll. A *.hdr fájl a kép metaadatait, *.bil pedig a pixel adatokat tartalmaz. Részletes leírása megtalálható a <http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/bil-bip-and-bsq-raster-files.htm> weboldalon.

A *tiff*, *jpg* fájlok jól ismert képformátumok, amelyek különböző színmélységű és sávszámú képek tárolására alkalmasak.

A menüsor alatti ikonosztázion (gyors gombok) láthatók a leggyakoribb fájltí-



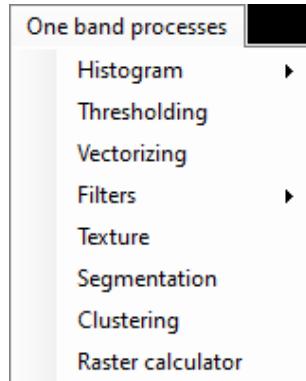
24. ábra. Képek megjelenítése a *Layer list* valamely elemének kiválasztásával

pusok megnyitására szolgáló ikonok, mint (open giwer format), (open bil format), (open tif format), (open jpg format), (open 3D) és (open video).

5.1.2. One band processes

Ez a menü akkor használható, amikor egy kiválasztott frekvenciasávval kívánunk műveleteket végre hajtani. Kiválasztás után meg is jelenik a képablakban. A *One band processes* menü elemei a következők: *Histogram*, *Thresholding*, *Vecorizing*, *Filters*, *Texture*, *Segmentation*, *Clustering*, *Raster calculator* (25. ábra).

- A *Draw histogram* almenü kirajzolja a kép hisztogramját, amelyet a kép kiegyenlítésére (kontrasztosításra) használhatunk. A bal egérgomb klikkel a minimális, a jobb egérgomb klikkel a maximális értéket választhatjuk ki. Az *Equalize* gombbal elvégezzük a kiegyenlítést.
- Ha a *Histogram equalization* menüre kattintunk, akkor automatikus választjuk ki a minimális és maximális értéket, és hajthatjuk végre a kiegyenlítést. Ilyenkor a hisztogram nem rajzolódik ki, csak az eredmény a képen.
- A *Thresholding* küszöbölést hajt végre egy megadható küszöbértéktől függően. Általában más eljárásokkal kombinálva használható.



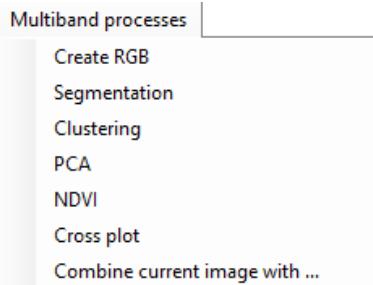
25. ábra. A **One band processes** menü

- A *Vectorizing* vektoros jellegű adatot állít elő a képből. A nyers képre nem érdemes alkalmazni, mert az eredmény rossz lesz. Mielőtt alkalmaznánk, előtte simítószűrést, küszöbölést és éldetektálás (Laplace filter) kell végezni.
- A *Filters* egy több almenüből álló gyűjtemény, amely különböző szűrési eljárásokat (*Gauss smoothing*, *High pass filter*, *Gradient filters*, *Laplace filter*, *Median filter*) végez a képen.
- A *Texture* menü az aktuális kép textúráját elemzi (fejlesztés alatt)
- A *Segmentation* menü az aktuális frekvenciasáv szegmentálását végzi.
- A *Clustering* menü az aktuális frekvenciasáv osztályozását végzi.
- A *Raster calculator* menü leválogatja az aktuális frekvenciasáv azon pixeljeit, amelyek a megadott feltételeket eleget tesznek.

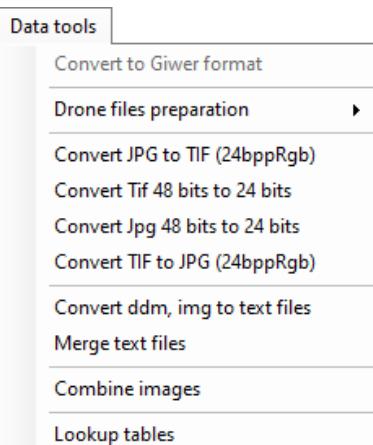
5.1.3. Multiband processes

A *Multiband processes* csoportba azok a funkciók tartoznak, amelyek egyszerre több frekvenciasáv adatait igénylik mint pl. RGB képek létrehozása, PCA (főkomponens analízis a megadott frekvenciasávokra), NDVI (vegetációs index), több frekvenciasáv alapján történő osztályozás, képek kombinálása (26. ábra).

- A *Create RGB* menü RGB képet kreál a megadott három frekvenciasávból
- A *Segmentation* menü a kiválasztott frekvenciasávokra szegmentálást végez.
- A *Clustering* menü a kiválasztott frekvenciasávokra osztályozást végez.
- A *PCA* menü a kiválasztott frekvenciasávokra főkomponens analízist végez
- Az *NDVI* menü a kiválasztott frekvenciasávokból vegetációs indexet számol.



26. ábra. A *Multiband processes* menü



27. ábra. A *Data tools* menü

- A *Cross plot* menü a kiválasztott két frekvenciasávból cross plottot rajzol.
- A *Combine current image with...* az aktuális képet kombinálja (+, -, EXOR) egy tetszőleges másik képpel. Ez olyankor hasznos, ha például egy vektorizált képet össze akarunk rajzolni az eredeti képpel (akkor az EXOR operátort kell használni).

5.1.4. Data tools

A *Data tools* menü az adatok előkészítésére való funkciókat tartalmazza (27. ábra). Különböző formátumok konverzióját, egyesítését, kombinálást végzi. Néhány a drón képekkel kapcsolatos problémát old meg. A *Lookup table* szerkesztését is lehetővé teszi. Legfontosabb funkciója azonban a *Convert to Giwer format* almenü. Ennek segítségével bármely nyers képformátumot (bil, tif, jpg) giwer formátumba konvertálja.

A giwer formátum kétféle fájlt jelent: a *.gwr* egy bináris fájl, ami a képet tartalmazza frekvenciasávonként, a *.gwh* pedig a kép header információit tartalmazza.

A *Convert to Giwer format* menü csak akkor aktív, ha a *Layers* fül listáján valamely tif, jpg vagy bil formátumban lévő kép ki van választva. Erre a menüre kattintva végre hajtódik a konverzió, és a *Config* fájlban megadott helyre (Giwer-DataFolder) képződik a konvertált adat. Ezután ezt a fájlt megnyitva a *Data Stock* teljes funkcionalitása rendelkezésre áll. A többi formátumú kép számára a feldolgozó műveletek nem aktiválhatók, csak a *giwer* formátumúakra.

A *Drone files preparation* menü bizonyos képek megfelelő formátumba hozására szolgál. A Micasense multispektrális kamera, amelyeknek 3 RGB, és 2 infravörös sávja, valamint egy termális sávja van, annyi tif fájl állít elő, ahány sávja van, jelen esetben 6 db tif fájlt. Ha ezeket szeretnénk gwr formátumba konvertálni, akkor két lehetőségünk van, amit két almenü tesz lehetővé:

- A *Merge multiple images to giwer format* menüre kattintva megjelenik egy dialógus ablak, ahol kiválaszthatjuk a kérdéses fájlokat. Ezután elindul a konverziós folyamat, amelynek eredményeként egy db *GWH* fájl és 6 db *GWR* fájl keletkezik. A header egy 6 sávos képet fog leírni. mindenéppen ez a módszer ajánlható, mert a *Multiband processes* feldolgozási folyamatok csak ilyen fájlokra működnek.
- A *Convert each multiple image to giwer format* menüre kattintva megjelenik egy dialógus ablak, ahol kiválaszthatjuk a kérdéses fájlokat. Ezután elindul a konverziós folyamat, amelynek eredményeként 6 db *GWH* fájl és 6 db *GWR* fájl keletkezik. Így minden egyes sáv egysávos képnek fog látszani. Ezekre csak a *One band processes* menü funkciók fognak működni.

5.1.5. Néhány hasznos funkció

A főbb funkciócsoporthoz után néhány apróbb, inkább kényelmi funkciót is ismertetünk. A menüsor alatt található egy ikonosztáz, amely a menürendszerből bizonyos funkciókat gyorsabban elérhetővé tesz:



Balról jobbra a következő funkciók érhetők el: gwr, bil, tif, jpg és video fájlok megnyitása, RGB kép készítése, kétféle hisztogram művelet, ahol az első interaktív, és meg is jelenízi a hisztogramot, a másik automatikus. Cross plot rajzoló két frekvenciasáv adatait jeleníti meg egy grafikonon. A **H** feliratú ikon a header adatok elrejtését/megjelenítését végzi.

A **G** ikon Gauss-simítást, a **H** a high pass filter indítja, az **M** a medián szűrést, az **L** a Laplace szűrést, a **Th** a thresholdingot. Ezután jön az osztályozás ikonja, majd az NDVI számítóé, a főkomponens analízisé (PCA), majd a rászter kalkulátor, és végül a Lookup table szerkesztő. A piros vissza-nyíl *undo* funkció, vagyis visszaállítja az utolsó művelet előtti állapotot (csak egy visszalépés lehetséges). A nagy piros gombok a workflow editáláshoz használatosak.

A *Layers* fülön három ikon van: . A piros kereszt törli az egész layer listát. A kinyíló könyv az összes listán lévő kép frekvenciasávjainak listáját

kinyitja, míg a bezáródó könyv bezárja őket. Ha csak egyetlen elemet akarunk törölni a listáról, akkor a jobb egérgombbal klikkeljünk a kívánt listaelemre, és a felbukkanó menüből válasszuk ki a törlést.

6. A Catalog alrendszer használata

A **Catalog** alrendszer nagy tömegben keletkező képek (csak *jpg* és *tif* formátumú képek) rendszerezésére, adatbázisba szervezésére való (ehhez sqlite-ot használ a program). Az adatok és a képek gyors szemrevételezését is lehetővé teszi. A **DataStock** alrendszer használata enélkül is lehetséges, hiszen bármely képet használatba vehetjük vele. A **Catalogot** olyankor célszerű használni, amikor több száz vagy több ezer képet kívánunk egységesen kezelni, a leíró adataik alapján keresést végezni.

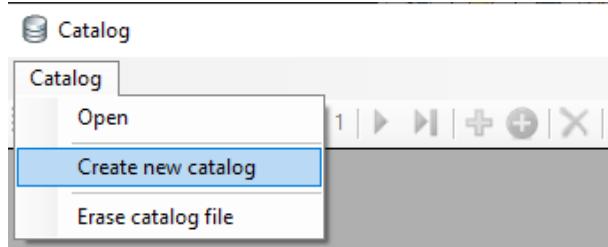
6.1. Első lépések

- Másoljuk be egy könyvtárba a catalog.zip fájlt.
- Bontsuk ki.
- Ahová kibontottuk, onnan indítható a catalog.exe, nem kell külön telepíteni.
- Az első induláskor még nincs képi adatbázis, ezért panaszkodni fog a hiányára (28. ábra):



28. ábra. A 'Missing database' üzenet

- Ezután megjelenik a program fő formja, ahol kreálhatunk egy új, üres adatfájl (29. ábra) (a default név 'dronimagecatalog.s3db', de lehet bármilyen más is)
- Ezután a 'Open' menüre kiklikkelve megnyílik az üres adatbázis-fájl.
- Ha már van létező adatbázis (pl. dronimagecatalog néven), akkor megjelenik a tartalma egy táblázatban a program fő formján (30. ábra). Ritka eset, de ha nincs, akkor panaszkodni fog, hogy nincs ilyen adatbázis — mert például



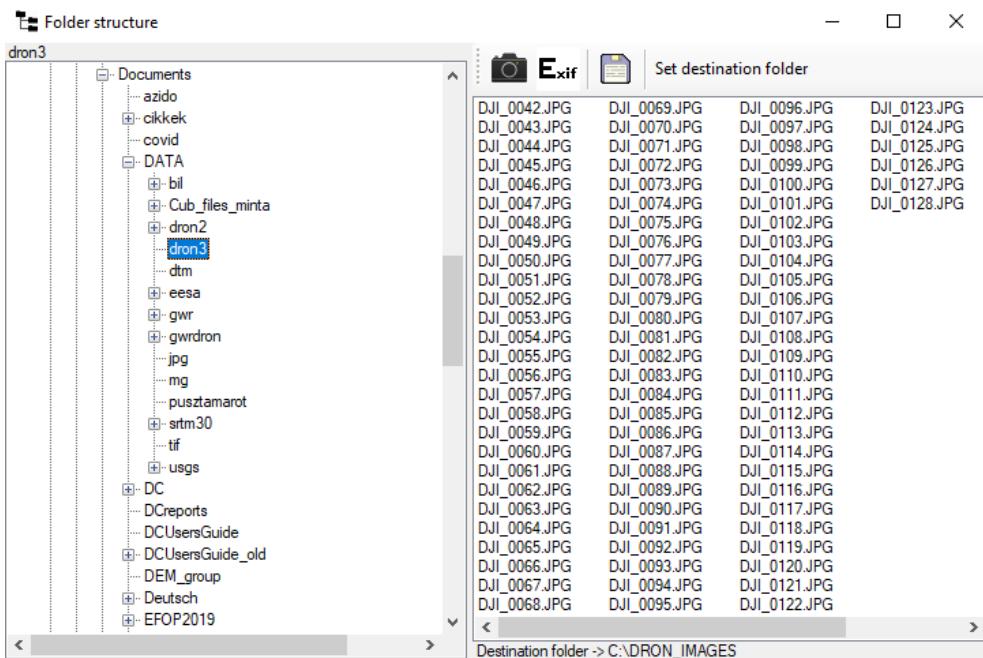
29. ábra. A *Create new catalog* menü

	id	filename	timestamp	type	bitspersample	samplesperpixel	image_size	file_size	location
▶	1	DJI_0042.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6935024	Tisztat
	2	DJI_0043.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6785908	Tisztat
	3	DJI_0044.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6783780	Tisztat
	4	DJI_0045.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6749706	Tisztat
	5	DJI_0046.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6768355	Tisztat
	6	DJI_0047.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6916669	Tisztat
	7	DJI_0048.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7021292	Tisztat
	8	DJI_0049.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6368130	Tisztat
	9	DJI_0050.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7082813	Tisztat
	10	DJI_0051.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6992482	Tisztat
	11	DJI_0052.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7360162	Tisztat
	12	DJI_0053.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6771052	Tisztat
	13	DJI_0054.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7270292	Tisztat
	14	DJI_0055.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7010302	Tisztat
	15	DJI_0056.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6969170	Tisztat
	16	DJI_0057.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7111348	Tisztat
	17	DJI_0058.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6983419	Tisztat

30. ábra. A *Catalog* fő formja

kitörültük a fájlrendszerből, de a catalog még úgy emlékszik, hogy van ilyen fájl. Kattintsunk az OK-ra, majd nyomjuk meg az F2 gombot -- bal felső sarok környéke a billentyűzeten. Ekkor megjelenik egy 'Catalog' nevű menü. Válasszuk ki a 'Create new catalog' almenüt, amely létrehoz egy 'dronimanagercatalog' nevű adatbázist, és benne egy üres adattáblát, amelynek 'images' lesz a neve. Ide fognak képződni a felvett képek adatai.

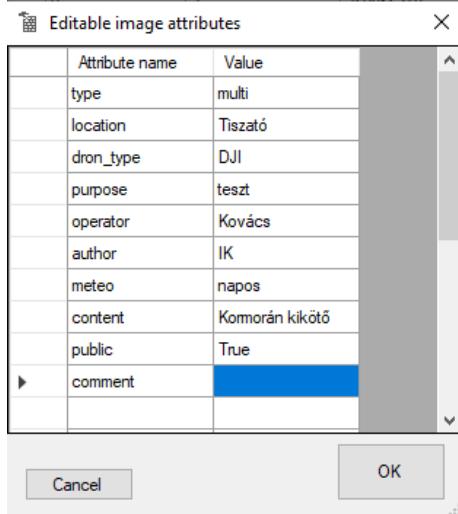
- Normál indulásnál a menürendszer nem látszik (F2-t megnyomva jelenik meg és tűnik el)
- Az ikonok elmondják, hogy mit tudnak, ha az egeret föléjük mozgatjuk.



31. ábra. A *Folder structure* ablak

6.2. A fájlrendszer előkészítése

- Kéreljunk egy könyvtárat 'DRON_IMAGES' néven valahol a fájlrendszerben.
- Klikkeljünk az 'open folder tree' ikonra (TREE). Ekkor megnyílik a 'Folder structure' nevű ablak (31. ábra).
- Klikkeljünk a 'Set destination folder' nevű menü gombra, majd keressük meg és válasszuk ki a 'DRON_IMAGES' nevű könyvtárat. Ezzel megadtuk a kép katalógus helyét a fájlrendszerben, amire ezentúl emlékezni fog a program, ha újra megnyitjuk a 'Folder structure' ablakot.
- Keressük meg a flash driven-on (ami a dronon a képeket tárolja) azt a könyvtárat, ahol az éppen most készített képek vannak. Ha a jobb oldali ablakban megjelennek a fájlok, klikkeljünk a 'Save files to 'c:\DRON_IMAGES folder' ikonra (FILE). Ennek hatására az egész könyvtár tartalma átmásolódik a flash driverről a 'DRON_IMAGES' nevű könyvtárba.
- Ennek hatására a 'DRON_IMAGES' nevű könyvtárban megjelenik egy új directory, aminek a neve az első fájl mentésének időpontja. Ez a könyvtár fogja tartalmazni az adott időben történt repülés képeit.



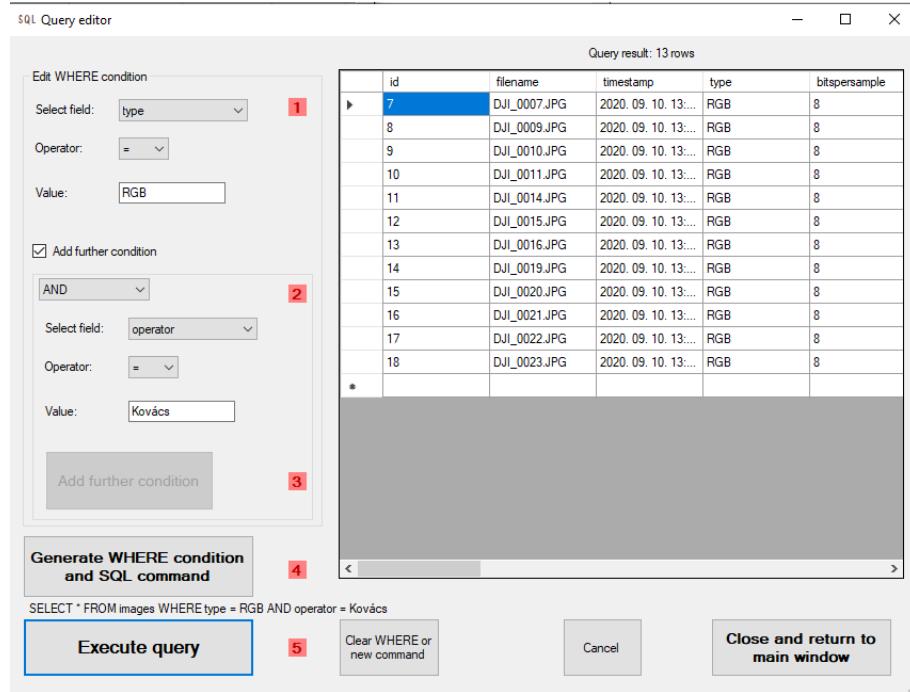
32. ábra. Az *Editable image attributes* ablak

6.3. Az adatbázis feltöltése

- Kétféleképpen tölthetjük fel az adatbázist: vagy egyenként (vagy multiselecttel több fájlt is) vagy egy directory-t kijelölve tömegesen, annak teljes tartalmát (csak jpg és tif fájl, más nem). A fájlonkénti kijelöléshez klikkeljünk a sárga plusz jelre (+), a teljes directory kijelöléséhez a zöld karikában fehér keresztkontra (⊕).
- Bármelyikre kikkeltünk, felbukkan a 'Editable image attributes' nevű ablak (32. ábra), ahol megadhatjuk azokat az adatokat, amelyek minden most beemelendő képre vonatkoznak. A többi adatot a program automatikus feltölti (fájlnév, long, lat, timestamp, folder, stb.).
- A táblázat nem automatikus adatai szerkeszthetők, amik el is mentődnek, amint a következő rekordra lépünk.
- A fényképezőgép ikonra (📷) kattintva megjelenik az aktuális rekordhoz tartozó kép. Az 'Exif' (Exif) feliratú ikon az aktuális rekordhoz tartozó kép exif adatait mutatja meg egy külön ablakban.

6.4. Funkciók

Az adatokat mutató táblázat felett egy ikonosztáz látható, amelyen a főbb funkciók lettek elhelyezve. A (E) ikon megnyit egy a fájlrendszer nézegető ablakot, hol megnézhetjük az adatok forrását, mint pl. egy pendrive-ot, ami közvetlenül a drón



33. ábra. Az *Sql editor* ablak

adattároló eszköze, és amelyen a legfrissebb mérési adatok vannak(31. ábra). A kiválasztott fájlokat (az egész könyvtárat) a *DRON_IMAGES* nevű könyvtárba másolja be. Amúgy ezt az első használat során meg kell adni (*Set destination folder*). A másolást a ikonra való klikkelés végzi.

Az adatbázisban már bent lévő képeket a ikonnal, míg a hozzá tartozó EXIF adatokat az ikonnal nézhetjük meg.

Új képeket, egyenként a ikonnal, míg tömegesen, vagyis egy egész directory tartalmát, a ikonnal adhatjuk hozzá az adatbázishoz. A hozzáadás egyben az adatbázis feltöltését is elvégzi, persze csak azokat az adatokat, amelyek a képekből kinyerhetők. Interaktívan is hozzáadhatók adatok, ha azokat a megfelelő mezőbe beírjuk. A ikonnal egy kijelölt rekordot törölhetünk. Nemcsak a leíró adatok törlődnek (az 'images' nevű tábla kijelölt rekordja), hanem a *DRON_IMAGES* könyvtárból is a kijelölt kép fájl (UNDO nincs!).

Az ikonnal SQL parancsokat állíthatunk össze, amelyekkel tetszőleges feltéTEL szerint kereshetünk (legyűjthetünk) a rendelkezésre álló képek paraméterei alapján. Az 33. ábrán olyan képek legyűjtésének eredménye látható, amelyek a Tiszán készültek, és a kép típusa 'multispektrális'. Az 'Sql editor' az Sql-t nem, vagy csak alapszinten ismerők számára is használható. (Az Sql-ben járatos felhasz-

nálók számára előhívható egy rejtett Sql parancssor, amely azért rejttet, mert hozzá nem értők kezében veszélyes fegyver lehet, amellyel súlyos károkat is lehet okozni az adatbázisban. Akik biztosak az Sql tudásukban, azok az F12 gomb megnyomásával előhívhatják az Sql parancsot, amely eltüntethető, ha újra megnyomjuk az F12 gombot. Nemcsak lekérdező, hanem non query típusú parancsok is kiadhatók. A parancs **Enterrel** hajtható végre.)

A  ikonnal egy adott mérésre vonatkozó riport fájlt nézetünk meg, vagy hozhatunk létre, amelybe olyan adatokat tehetünk bele, amelyeket a mérési körülmenyek miatt, vagy bármilyen szempontból érdekesnek találunk, de nem az egyes képekhez kötöttek.

6.5. Lekérdezés

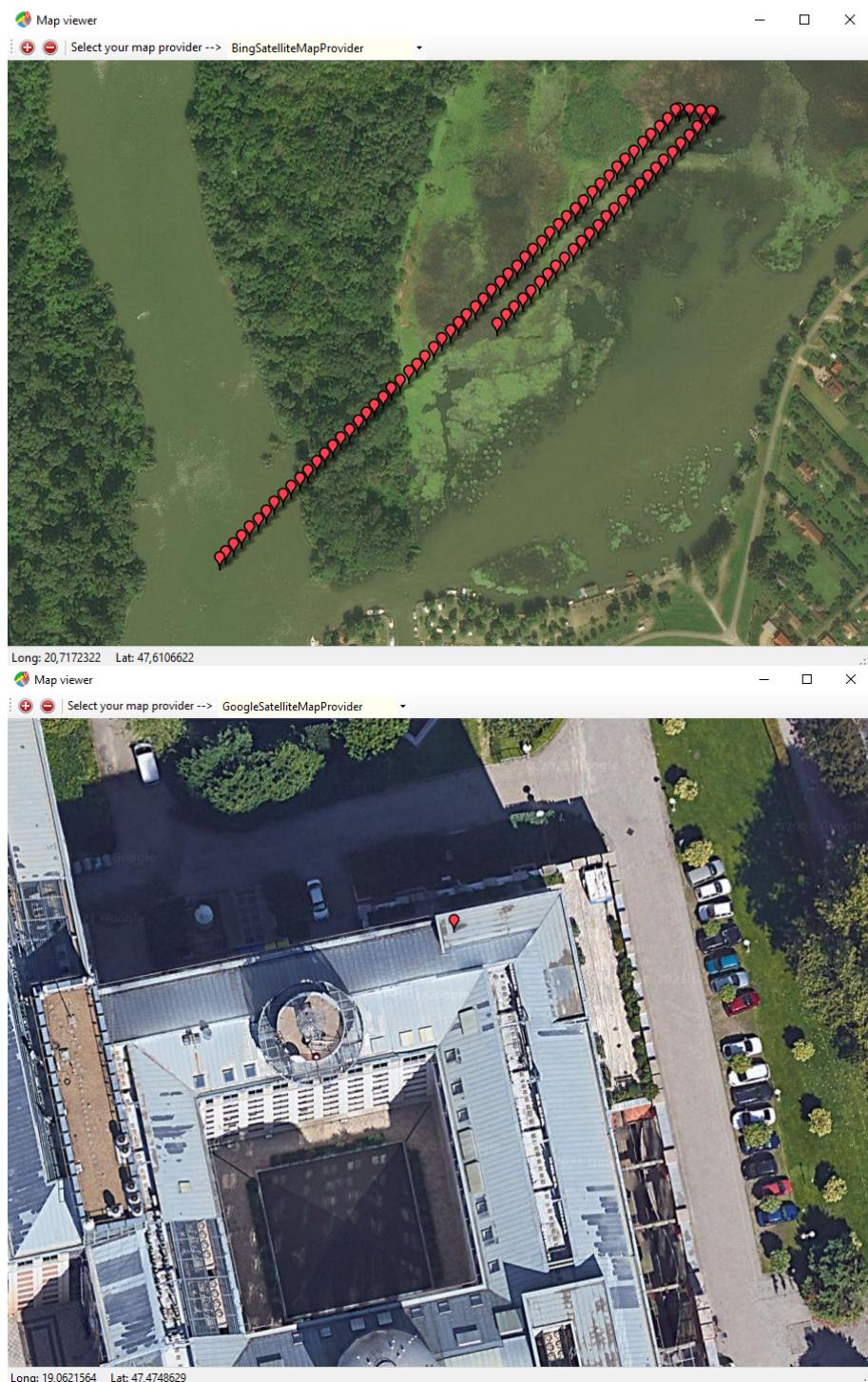
- Az 'SQL' feliratú ikonra kikkelve () megjelenik egy 'Query editor' nevű ablak. Itt ki lehet választani, hogy melyik mezőre kérdezünk, milyen feltételt szabunk.
- pl. select field: type; Operator: =; Value: RGB =====> WHERE type=RGB. Ha itt vége, akkor click to 'Generate WHERE condition and Sql command' majd 'Execute query'.
- Ha új lekérdezés lesz, akkor előtte click to 'Clear WHERE or new command'. Vigyázat, az Sql editor case sensitive (rgb != RGB)
- Összetettebb lekérdezésekhez az előbbihez hasonló lekérdezés után kikkéljünk az 'Add further condition' nevű check boxra.
- Ha kész vagyunk egy further feltétellel, kikkéljünk az 'Add further condition'-gombra. Ha az utolsót is hozzáadtuk, akkor kikkéljünk a 'Generate WHERE condition and Sql command' majd az 'Execute query'-re. Ha jó volt az sql parancs, akkor megjelenik az eredmény az adatrácsban.
- Ha meg vagyunk elégedve az eredménnyel, kikkéljünk a 'Close and return to main window' gombra. Ekkor becsukódik a 'Query editor' ablak, és a lekérdezés eredménye megjelenik a fő ablakban. Itt nézegethetjük a képek listáját.
- A 'Select all' feliratú gombra a bal egérgombbal kikkelve az összes képet legyűjthetjük az adatbázisból, amelyek adatai meg is jelennek az adatrácsban.
- A 'Select all' feliratú gombra a jobb egérgombbal kikkelve az összes képet kijelölhetjük az adatrácsban (34. ábra). Ezt olyankor hasznos, amikor térképen akarjuk megjeleníteni a legyűjtött képek centroidjait. Ehhez még rá kell kattintani a  ikonra. Ekkor megjelennek a 'Map viewer' ablakban (35. ábra felső része) a képek centroidjai. Ha úgy kikkeltünk a  ikonra, hogy

C:\Users\eleki\Documents\proba.s3db

	id	filename	timestamp	type	bitspersample	samplesperpixel	image_size	file_size	location
▶	1	DJI_0042.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6935024	Tiszat
	2	DJI_0043.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6785908	Tiszat
	3	DJI_0044.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6783780	Tiszat
	4	DJI_0045.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6749706	Tiszat
	5	DJI_0046.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6768355	Tiszat
	6	DJI_0047.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6916669	Tiszat
	7	DJI_0048.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7021292	Tiszat
	8	DJI_0049.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6368130	Tiszat
	9	DJI_0050.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7082813	Tiszat
	10	DJI_0051.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6992482	Tiszat
	11	DJI_0052.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7360162	Tiszat
	12	DJI_0053.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6771052	Tiszat
	13	DJI_0054.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7270292	Tiszat
	14	DJI_0055.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7010302	Tiszat
	15	DJI_0056.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6969170	Tiszat
	16	DJI_0057.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	7111348	Tiszat
	17	DJI_0058.JPG	2020. 09. 10. 14:16	RGB	8	3	5280x2970	6983419	Tiszat

34. ábra. A *Map viewer* ablak

nem jelöltünk ki egyetlen képet sem, akkor az ELTE Térképtudományi és Geoinformatikai Intézet helye jelenik meg a térképen (a 35. ábra alsó része).



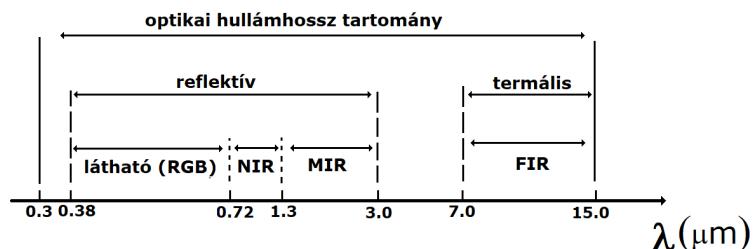
35. ábra. A *Map viewer* ablak. Felső részen a Kormorán kikötő (Tiszafüred), míg az alsón az ELTE látható

III. rész

Függelék

1. A távérzékelés fizikai alapjai

A távérzékelés megjelenését a globális erőforrások kimerülése okozta, mint például nyersanyagok (olaj), környezeti problémák (környezetszenyezés, fajok kipusztulása) nagyjából a 60 évektől kezdődően. Az ūrtechnika és a számítástechnika felgyorsult fejlődése tette lehetővé. Az elektromágneses spektrumnak főként az optikai részét (36. ábra) használjuk (RGB + infra sávok), de előfordulnak hosszabb frekvencia tartományok is, mint pl. a rádióhullámok (RADAR).



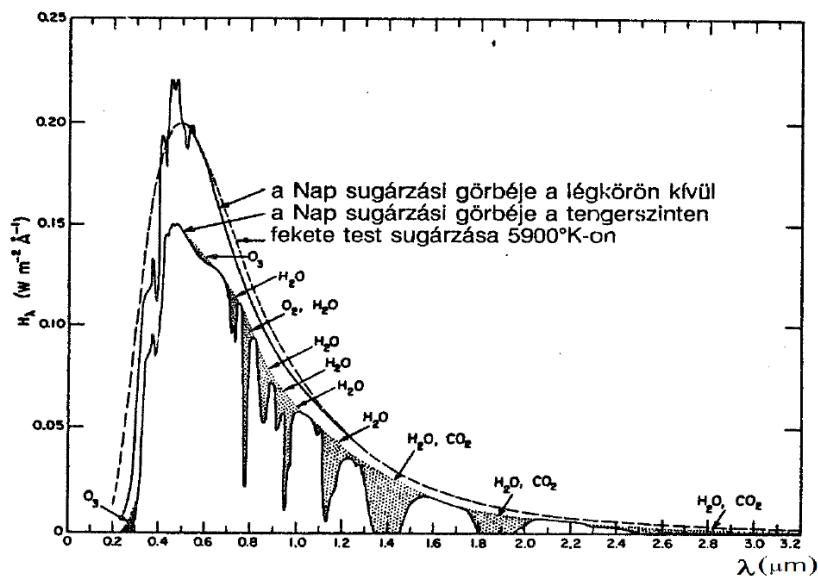
36. ábra. Az elektromágneses spektrum optikai része, ahol az RGB a látható tartományt, NIR a közeli infravörös, a MIR a közepes infravörös, és a FIR a a távoli infravörös tartományt jelöli

A Napból bejövő elektromágneses sugárzást, amelynek energiaeloszlását a 37. ábrán látható, a légkör jelentősen meg változtatja (37. ábra), amit figyelembe kell vennünk az eszközök tervezésekor és a képek értelmezésekor. A visszavert elektromágneses sugárzást befolyásolják a felszínt fedő képződmények (víz, növényzet, talajok), ezért azok jelenlétére következtethetünk a visszavert képből (38. ábra).

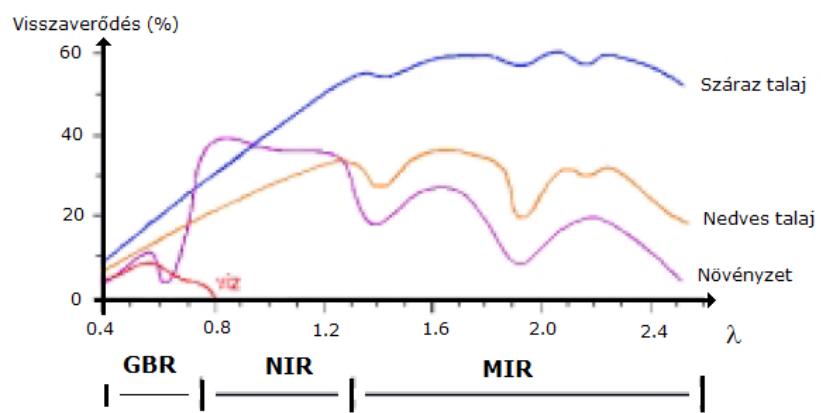
A Napból jövő elektromágneses sugárzást érzékelik a kameráink, amely alapján következtetünk a felszín bizonyos tulajdonságaira. A kamerák, főként a multispektrálisok vörös és infravörös tartományai számunkra fontos felszíni képződmények anyagminőségére érzékenyek, így a kép alapján következtethetünk az ott lévő anyagokra (39. ábra).

A távérzékelés egyik fő fókuszpontja az agrárium. A cél valamely előre meghatározott anyagminőség detektálása és azok minősítése. Nézzünk néhány lehetséges célkitűzést, vagyis hogy mit akarunk kimutatni. Megállapítunk szakmailag indokolt tematikus kategóriákat, és megpróbáljuk a felszíni képződményeket valamelyik tematikus csoportba besorolni. Nézzünk néhány példát a lehetséges tematikus csoportokra:

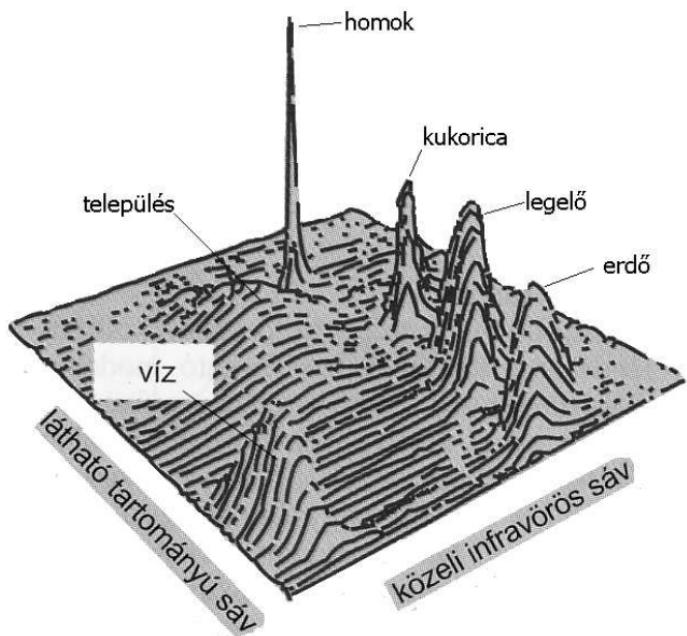
1. Aszályval kapcsolatos kategóriák



37. ábra. A Nap direkt sugárzásának energiaeloszlása



38. ábra. A különböző felszíni képződmények másként verik vissza a rájuk eső elektromágneses sugárzást



39. ábra. A növényzet reakciója a vörös és az infravörös tartományra

- aszályal erősen sújtott terület
- aszályal közepesen sújtott terület
- aszály által érintett terület
- aszály által nem érintett terület
- aszály által nem veszélyeztetett terület

2. Haszonnövényekkel fedett területek kategóriái

- Őszi búza
- Tavaszi árpa
- Őszi árpa
- Kukorica
- Silókukorica
- Napraforgó
- Cukorrépa
- Lucerna
- Vízfelszínek
- Nem mezőgazdasági területek

- Egyéb szántóföldi növények

3. Felszín fedettségi kategóriák

- Lakott területek
- Ipari, kereskedelmi területek
- Bányaák, lerakóhelyek, építési munkahelyek/
- Mesterséges, nem mezőgazdasági területek
- Szántóföldek
- Állandó növényi kultúrák
- Legelők
- Vegyes mezőgazdasági területek
- Erdők
- Cserjés, vagy lágyszárú növényzet
- Növényzet nélküli területek
- Szárazföldi vizenyős területek
- Kontinentális vizek

4. Erdőkárok kategorizálása

- Nincs károsodás
- kis mértékű károsodás
- Közepes mértékű károsodás
- Erős károsodás
- Egyéb terület

5. Szőlő és gyümölcskultúrák kategóriái

- Szőlőültetvény
- Gyümölcsültetvény
- Aprótáblás művelési rendszerű szőlő- vagy gyümölcsültetvény
- Bizonytalan, de lehetséges ültetvény

A fent bemutatott néhány példa csak szemléltetés volt a számtalan lehetséges csoportosítás közül. Ezeket minden az a szakmai cél dönti el, hogy mit kívánunk kimutatni (pl. gyomos területek felderítése).

2. A főkomponens analízis matematikai alapjai

Sokdimenziós adatrendszer esetében alkalmazunk dimenzió csökkentő eljáráskat. A dimenzió csökkentés egyik lehetséges módja a főkomponens analízis (Principal Component Analysis), amely a többváltozós matematikai statisztika egy széles körben elterjedt eljárása. A következőkben a főkomponens analízis valószínűségi megfogalmazását adjuk meg, de lehetséges algebrai megoldás is, amelyet a fizikusok használnak előszeretettel a mechanikában (főtengely transzformáció néven).

A valószínűségi megfogalmazás a következő: legyen p számú megfigyelési egységiünk, amelyek egyenként n számú adatot tartalmaznak (p számú megfigyelési vektorunk van).

\mathbf{x}^1	\mathbf{x}^2	\dots	\mathbf{x}^p
x_1^1	x_1^2	\dots	x_1^p
x_2^1	x_2^2	\dots	x_2^p
\vdots			\vdots
x_n^1	x_n^2	\dots	x_n^p

Tekintsük az \mathbf{x}^j vektorokat valószínűségi változóknak, a vektorok elemeit a valószínűségi változók realizációinak. Standardizáljuk a változókat:

$$\tilde{x}_i^j = \frac{x_i^j - \bar{x}^j}{s^j}$$

ahol \bar{x}^j a j -edik vektor elemeinek átlaga (a várható érték becslése), és s^j az empirikus szórása. Így tehát 0 várható értékűvé, és 1 szórásúvá tettük a valószínűségi változóinkat. Ezek után számítsuk ki az adatrendszerünk korrelációs mátrixát:

$$\underline{\mathbf{R}} = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1p} \\ r_{21} & r_{22} & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \\ r_{p1} & r_{p2} & \dots & r_{pp} \end{pmatrix}$$

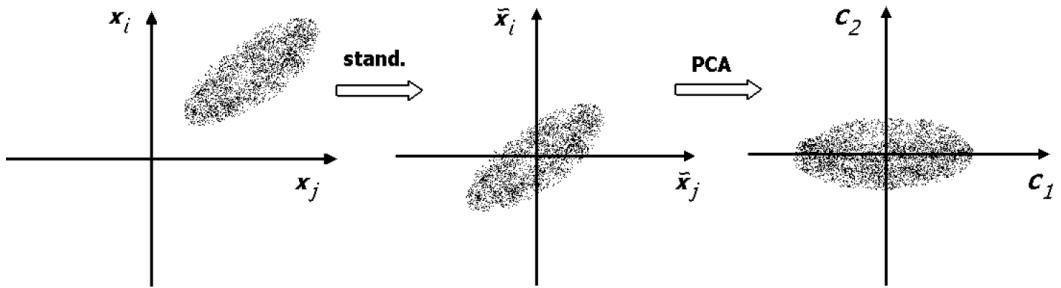
ahol r_{ij} az i és j -edik megfigyelési egységek korrelációs együtthatója. Határozuk meg a korrelációs mátrix sajátértékeit és sajátvektorait, vagyis oldjuk meg a következő sajátérték egyenletet:

$$\underline{\mathbf{R}}\mathbf{v} = \lambda\mathbf{v}$$

A sajátértékek $\lambda_1 < \lambda_2 < \dots < \lambda_p$, a sajátvektorok $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p$. Számítsuk ki a főkomponenseket a következő módon, legyen a j -edik főkomponens a következő:

$$C_i^j = \sum_p x_i^p v_p^j$$

ahol $i = 1, n$ és $j = 1, p$.



40. ábra. A folyamat geometria jelentése: a standardizálás 0 várható értékűvé és 1 empirikus szórásúvá teszi a változókat, vagyis a pontfelhőt betolja az origóba, majd elforgatja a legnagyobb variancia irányába, ami az első főkomponens

A főkomponensek ortogonális rendszert alkotnak, vagyis korrelálatlanok, azaz korrelációs mátrixuk

$$\underline{\mathbf{R}}_C = \begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_p \end{pmatrix}$$

A $\underline{\mathbf{R}}_C$ fontos tulajdonsága, hogy a főkomponensek és a standardizált változók össz-varianciája azonos:

$$\sum_{j=1}^p \lambda_j = \sum_{i=1}^p \tilde{s}_i^2 = \sum_{j=1}^p s_j^2 = p$$

Amint látható, a főkomponensek kiszámításával nagymértékben átrendeztük a varianciákat, mivel (ha ez lehetséges volt), összevontuk őket az első (néhány) főkomponensbe. Az eljárás főbb mozzanatainak geometria jelentését a 40. ábra mutatja.

Abban az esetben, ha például az első főkomponens képes magába sűríteni a megfigyelési egységek varianciáinak nagy részét, akkor megtehetjük, hogy az egész adatrendsztet csak az első főkomponensével helyettesítjük. Így jelentős mértékben csökkentettük az adatrendszer dimenziósát, ezzel adatszámát, azaz meggörültünk, megkönnyítettük egy soron következő eljárás, például a klaszter analízis működését.

Felmerülhet a kérdés, hogy mikor nem használható az első főkomponens a teljes adatrendszer helyettesítésére. Ha a korrelációs mátrix diagonális, vagyis a változók korrelálatlanok, akkor biztosan nem. Ebben az esetben minden további számítás értelmetlen.

Egy másik kézenfekvő kérdés, hogy ha például a megfigyelési egységeink mérföldi értékek (pl. digitális képek, fizikai mennyiségek), akkor miért tekintjük őket valószínűségi változóknak. Ennek pusztán az az oka, hogy először a többváltozós matematikai statisztika használta ezt az eljárást dimenziószám csökkentésre. A probléma leírható algebrai módszerekkel is, mint fentebb említettük.

3. A digitális szűrési eljárások áttekintése

A digitális szűrések általában azok a képfeldolgozó eljárások, amelyek vagy az időtartományban, vagy a frekvenciatartományban manipulálják a képet. Az időtartományt történeti okokból nevezzük annak, helyesebb volna inkább tértartománynak hívni. A frekvenciatartománybeli képet spektrumnak is nevezzük.

Jelöljük a képet $s(t)$ -vel, a kép spektrumát $S(f)$ -el és a Fourier-transzformációt \mathcal{F} -el. Az idő- és a frekvenciatartományt a Fourier-transzformáció köti össze:

$$S(f) = \mathcal{F}[s(t)] \quad (2)$$

illetve

$$s(t) = \mathcal{F}^{-1}[S(f)] \quad (3)$$

A 2 formula a direkt Fourier-transzformáció, míg a 3 az inverz Fourier-transzformáció.

Bizonyos szűrési eljárások hatását függvények formájában (átviteli függvény) adjuk meg a frekvenciatartományban (pl. konvolúciós szűrők), míg mások csak az időtartományban értelmezhetők (pl. medián szűrő). Ha például egy kép spektrumát megsorozzuk egy olyan négyzetfüggvényel, amely $[-f_f, f_f]$ intervallumon kívül nulla, akkor a spektrumból eltávolítjuk az f_f felső határfrekvenciánál nagyobb frekvenciájú összetevőket (vagyis simítjuk). A megváltozott spektrum inverz Fourier-transzformációjával megkapjuk a simított képet. A simítás mértéke természetesen függ a felső határfrekvenciától, mely minél alacsonyabb, annál erősebb a simítás. Általánosságban tehát a digitális szűrők működése a következő:

1. legyen egy $s(t)$ függvényünk (ez jelenti a digitális képet)
2. Fourier-transzformáljuk, vagyis számítsuk ki a spektrumát: $S(f) = \mathcal{F}s(t)$
3. szorozzuk meg a spektrumot az átviteli függvényel: $S'(f) = S(f)A(f)$, ahol $A(f)$ az átviteli függvény
4. inverz Fourier-transzformáljuk a kapott spektrumot, és ezzel megkaptuk a szűrt képet: $s'(t) = \mathcal{F}^{-1}S'(f)$

A legtöbb „jól viselkedő” függvény Fourier-transzformálható. Mivel ismert a szűrési műveletünk átviteli függvénye (magunk adjuk meg, attól függően, hogy mit akarunk csinálni a képpel), akkor annak inverz Fourier-transzformáltja előre

kiszámítható. Az ismert konvolúciós azonosság szerint két függvény konvolúciója az időtartományban, megegyezik ezen függvények spektrumainak szorzatával:

$$s(t) * a(t) = \mathcal{F}^{-1}[S(f) \cdot A(f)] \quad (4)$$

Ezt az összefüggést kihasználva, és az átviteli függvény inverz Fourier-transzformációjának ismeretében, az időtartományban is elvégezhetjük a szűrést, mégpedig úgy, hogy az eredeti képet (az időtartományban) konvolváljuk az átviteli függvény inverz Fourier-transzformáltjával:

$$s'(t) = s(t) * \mathcal{F}^{-1}[A(f)], \quad (5)$$

ahol $A(f)$ az átviteli függvény.

Az eddig okfejtések folytonos esetekre vonatkoztak. Diszkrét esetben a Fourier-transzformáció neve diszkrét Fourier-transzformáció (DFT), amelynek gyors és hatékony algoritmusa a gyors Fourier-transzformáció (FFT). $s(t)$ a digitális kép, és az átviteli függvény ($A(f)$) is diszkrét, beleértve annak inverz Fourier-transzformáltját is, vagyis az $a(t)$ függvény mintavételezett értékeivel fog történni az időtartománybeli konvolúció (5. formula).

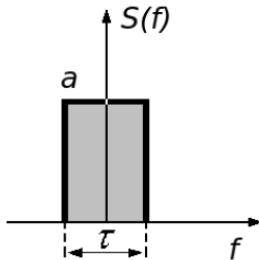
Az átviteli függvény inverz Fourier-transzformáltjának diszkrét változatára bevezették a kernel elnevezést, amely mára önálló fogalomtáv vált. Nemcsak olyan esetekben használják, amikor a művelet hatása megadható a spektrum valamely függvényteljesítésével, hanem olyan esetekben is, amikor az időtartománybeli művelet hatása nem adható meg a frekvenciatartományban (pl. rangszűrők).

A legtöbb digitális szűrő kernelt használ. Ezek a szűrők alapvetően úgy működnek, hogy egy kernelt, vagyis egy $[(2k+1) \times (2k+1)]$ méretű ablakot, futtatunk végig a szűrni kívánt kép minden pontján, úgy, hogy az aktuális képpont a kernel közepére esik, majd a kernel alá eső értékekkel valamilyen eljárással kiszámolják az aktuális pixel szűrt értékét.

Általában ez a következő módon történik: legyen $g(x, y) = F\{f(x, y)\}$, ahol $g(x, y)$ a szűrt kép x, y koordinátájú pontját, f az eredeti képet, $F\{\}$ azt az operátort jelenti, amely az eredeti kép x, y koordinátájú pontjának szomszédaiból kiszámolja a szűrt értéket. Azt a megfigyelést használjuk ki, hogy az egymáshoz közeli képpontok értékei jobban összefügggenek, mint az egymástól távoli pixelek. Ezeknek a szűrőknek egy másik fontos jellemzője, hogy nem rekurzívak. Ez azt jelenti, hogy az eljárás csak az eredeti intenzitásértéktől függ, azaz minden képpontnak ugyanaz a szomszédság.

3.1. Konvolúciós szűrők

Jelölje $f_1(t)$ a konvolválandó függvényt, és $f_2(t)$ a kernelt. A konvolúciós, vagy lineáris szűrők úgy működnek, hogy a kernelben szereplő értékekkel konvolválják az időfüggvényt, és ez lesz a konvolúció értéke a t -edik helyen:



41. ábra. A négyszögimpulzus a felülvágó szűrő átviteli függvénye a frekvencia tartományban (az egyszerűség kedvéért egy egydimenziós időfüggvényt látunk)

$$h(t) = \sum_{\tau=-\infty}^{\infty} f_1(\tau) f_2(t - \tau) \quad (6)$$

Kétdimenziós esetre, mint amelyen a digitális kép, a 6 formulával megadott konvolúció a következőképpen alakul:

$$h(x, y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f_1(u, v) f_2(x - u, y - v) \quad (7)$$

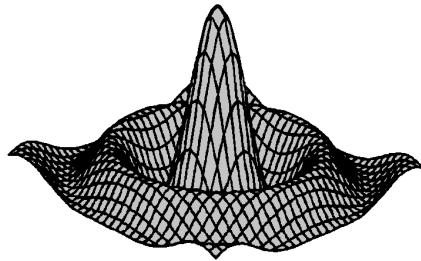
A szűrő átviteli karakterisztikáját a kernelben lévő értékek határozzák meg, amelyek egyébként az átviteli függvény inverz Fourier-transzformáltjának diszkrét értékeiből állnak.

A szemléletesség kedvéért nézzünk meg egy f_f felső határfrekvenciájú felülvágó szűrőt. A szűrő átviteli függvényét mutatja a 41. ábra, illetve ennek inverz Fourier-transzformáltját a 42. ábra, amely megfelelően mintavételezve adja a kernelbe töltendő együtthatókat.

Ezzel a kernellel végrehajtva a konvolúciót kapjuk az ú.n. konvolúciós szűrőket. Átviteli karakterisztikájuk a kernel tartalmától (vagyis a frekvenciatartományban definiált átviteli függvénytől) függ. Két dimenzióban, mint amilyen a digitális kép, a kernel a 2D-s *sinc* függvény, amely a 42. ábrán látható.

3.1.1. Szeparábilis szűrők

Ha a kernelmátrix speciális alakú, akkor a konvolúció végeredményét lényegesen gyorsabban számolhatjuk ki. Ha fennáll, hogy a w kernelmátrix felbontható egy oszlop- és egy sorvektor szorzatára, azaz $w(i, j) = u(i) * v(j)$, akkor megtehetjük azt, hogy először kiszámoljuk a kép u -val vett konvolúcióját, majd az eredmény v -vel vett konvolúcióját, azaz



42. ábra. A kétdimenziós *sinc* függvény, amelynek mintavételezett értékeiből ál a simító szűrő kernelje

$$f'(x, y) = \sum_{i=x-k}^{i=x+k} f(i, y) * u(i + k),$$

és ebből

$$g(x, y) = \sum_{j=y-k}^{j=y+k} f'(x, j) * v(j + k)$$

3.1.2. Box szűrő

A box szűrő olyan speciális kernelű szűrő, ahol a kernelmátrixban szereplő összes érték ugyanannyi, vagyis a kernel alá eső pixeleket átlagoljuk. A box szűrő hatása a kép simítása. Önmagában nem ad túl jó eredményt, de más szűrőkkel kombinálva (pl. élmegőrzők) hasznos eszköz lehet. Egyszerűségének köszönhetően igen népszerű szűrő, annak ellenére, hogy a spektrumra gyakorolt hatása meglehetősen kedvezőtlen. (Ha meggondoljuk, ilyenkor az időtartományban egy négyzetfüggvénnyel végezzük a konvolúciót, aminek az átviteli függvénye, a frekvencia tartományban egy *sinc* függvény, amiről minden elmondható, csak az nem, hogy értelmezhető lenne rá felső határfrekvencia).

3.1.3. Gauss-szűrő

A Gauss-szűrő kernelében az értékek a kétdimenziós Gauss-eloszlás értékei szerepelnek. A Gauss-eloszlás a következőképp számolható:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Így a 0 várható értékű, σ szórású Gauss-görbét kapunk, amelyből a kernelt úgy számoljuk, hogy a rácspontokon mintavételezzük a G függvényt. Mivel



43. ábra. Balról jobbra: Eredeti kép. A kép 3×3 -as, Gauss-szűrt változata, $\sigma = 1.0$. A kép 7×7 -es, box-szűrt változata. A kép 7×7 -es Gauss-szűrt változata, $\sigma = 2.0$

$$\frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} * \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}$$

ezért a Gauss-szűrő is szeparábilis, az u és v vektor a következőképp számolható:

$$u(x) = v(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-(k+1))^2}{2\sigma^2}}$$

A Gauss szűrőnek létezik egy speciális, még gyorsabb implementációja. Ha a haranggörbe értékeit 2 hatványaival közelítjük, akkor nem kell szoroznunk, amikor a konvolúciót végezzük, hanem megfelelő számú bittel kell csak eltolni az értékeket. Ekkor az u és v vektorok

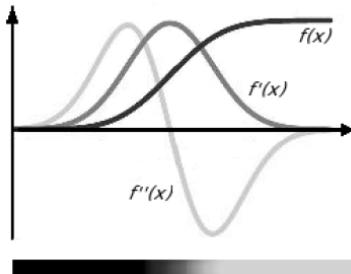
$$u(x) = v(x) = 2^{(k+1-|x-k|)}$$

alakúak lesznek.

A Gauss-szűrő, mint látható, simító jellegű. A simítás mértéke a szórás nagyságától függ, természetesen nagy szórás esetén a kernel méretét is növelni kell (43. ábra). A Gauss-szűrőnek szintén nem önmagában, hanem más algoritmusokban van szerepe, pl. a Canny-féle éldetektorban használjuk. Átviteli függvénye, ha nem is ideális, de lényegesen jobb a box szűrőnél. Gyakran használják „csonkító” függvényként, amivel például az ideális felülvágó kerneljének a hosszát csökkentik le, ezáltal javítva a futásidőt. (Ne feledjük, hogy ebben egy *sinc* függvény van mintavételezve.)

Hátránya, hogy a simítás miatt a képen található élek elmosódottá válnak a felső határfrekencia függvényében. Ha meggondoljuk, nem meglepő ez a eredmény, hiszen a spektrumból eltávolítjuk a nagyfrekvenciás részeket, márpedig az éleken éppen a nagyfrekvenciás összetevők játsszák a legfőbb szerepet.

A Gauss-szűrő segítségével lehetséges a képek méretének egyszerű csökkentése (felezése). Az algoritmus úgy működik, hogy a kiinduló képre alkalmazzuk a Gauss-szűrőt, majd elhagyjuk minden második sort és oszlopot. Az így létrejövő folyamatosan feleződő képeket Gauss-piramisnak is nevezik. A Gauss-piramisban az egymás felettes képeket egymásból kivonva a heterogén részek felerősödnek, ez a tulajdonság jól használható a textúrák detektálásánál.



44. ábra. Az éleken az első deriváltnak maximuma, a második deriváltnak zérushelye van

3.1.4. Nem-szeparábilis szűrők

A nem-szeparábilis szűrők azok, melyek kernelmátrixa nem írható fel két vektor szorzataként. Ilyenkor az eredeti konvolúciós képletben szereplő összegzést kell megvalósítani.

3.2. Éldetektorok

A éldetektorok olyan lokális szűrők, melyek a kép egy pontjában a szomszédos elemek segítségével leírt intenzitásfüggvény deriváltjával dolgoznak (44. ábra). Feladatuk, hogy az éleket kiemeljék, a hasonló pixelekből álló csoportokat pedig eltüntessék.

3.2.1. Gradiens szűrő

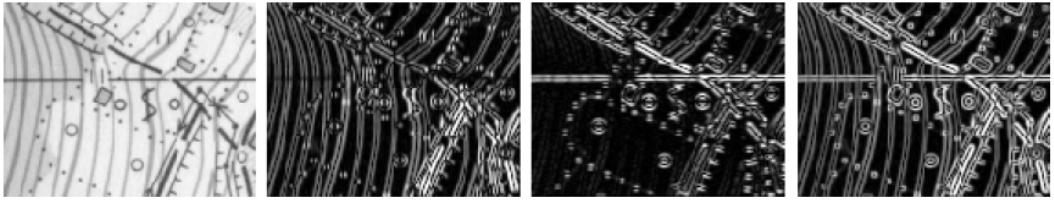
A gradiens szűrő használatával a kép mint felület pontjaiban vett deriváltak x és y irányú gradiensét közelítjük a differencia hánnyadossal.

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

A szűrő nagy intenzitásváltozásokra reagál, a homogén területekre 0-t ad eredményül. Kernelje a következő:

$$\mathbf{G}_x = \begin{pmatrix} -1 & 0 & 1 \\ 2-p & 0 & p-2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{G}_y = \begin{pmatrix} -1 & 2-p & -1 \\ 0 & 0 & 0 \\ 1 & p-2 & 1 \end{pmatrix}$$



45. ábra. Balról jobbra: Eredeti kép. A kép x és y irányú gradienseinek izotropikus gradiens szűrővel. Az előző két képből számolt élkiterjedés

A p érték szabadon választható, gyakran használt értékek a $p = 2$, $p = 3$, $p = 2 + \sqrt{2}$. Az első esetben Prewitt, a másodikban Sobel, a harmadik pedig izotropikus operátorról beszélünk (45. ábra). Az eredményt p -vel normalizálni kell.

3.2.2. Laplace-szűrő

A Laplace operátor definíciója:

$$\Delta f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Ha a Laplace operátort diszkretizáljuk, akkor a következő egyenletet kapjuk az x, y pontbeli második deriváltra:

$$f''(x, y) = f(x - 1, y) + f(x + 1, y) + f(x, y + 1) + f(x, y - 1) - 4 * f(x, y)$$

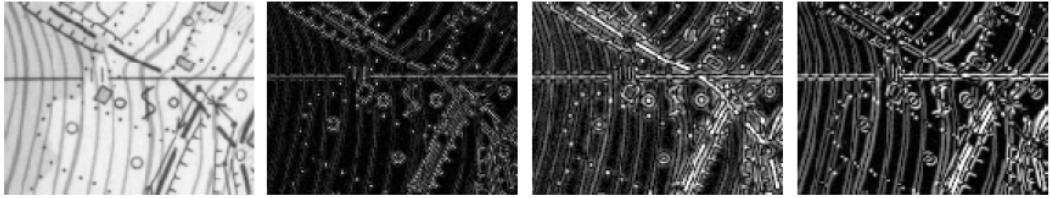
Így a Laplace szűrt értéket az (x, y) pontban a következő konvolúciós kernellel számolhatjuk:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

A Laplace szűrő a gyakorlatban a kép simított és eredeti változatának különbsége, ezért az intenzitásváltozásokra, így a hibákra is nagyon erősen reagál. Simítás nélkül, önmagában nem túl eredményes (46. ábra).

3.2.3. LoG szűrő

A Laplace szűrő előtt egy Gauss-simítást alkalmazva jó éldetektort kaphatunk (46. ábra). Mivel a konvolúció asszociatív, ezért megtehetjük, hogy a Laplace- és Gauss-szűrő kerneljét konvolváljuk, és az eredményként kapott mátrixot használjuk. Ezután a szűrőt szokták „Laplacian of Gaussian” (LoG) nevezni. A LoG-szűrő kevésbé érzékeny a zajra, jó éldetektor. A kernelmátrix a következőképp számolható:



46. ábra. Balról jobbra: Az eredeti kép. A kép Laplace-szűrt változata. A kép LoG-szűrt változata. A kép emboss szűrt változata: ÉK-DNy irányú élek megtalálására beállítva

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

3.2.4. Emboss szűrő

Az emboss szűrők célja speciális irányú élek detektálása (46. ábra). Ehhez olyan kernelt alkalmazunk, amelynek két átellenes szélén +1 illetve -1 található. Attól függően, hogy milyen irányú átlóban vannak az értékek, az arra merőleges élekre reagál érzékenyen a szűrő. Példa egy lehetséges emboss szűrő kernelre:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Ez a kernel az ÉK-DNy irányú éleket fogja detektálni, míg az erre merőlegeseket észre sem fogja venni.

3.2.5. Canny-féle éldetektor

Az éldetektálás különösen fontos szerepet játszik az alakfelismerésben, a raszteres térképek vektorossá alakításában. Az élek a képnek azon helyei, ahol az intenzitás megváltozása a legnagyobb. Először is döntsük el, hogy mennyire kifinomult élek kimutatását szeretnénk. A legtöbbször érdemes előzetesen simító vagy medián szűrésnek alávetni a képet, hogy ne mutassunk ki minden apró, jelentéktelen élt. Ismert és egyszerű módja a simításnak a kép és egy Gauss-függvény konvolúciójának alkalmazása. Legyen h az f és g függvények konvolúciója. Kimutatható, hogy

$$h' = (f * g)' = f * g',$$

vagyis egy kép (jelöljük f -el) Gauss-függvényvel (g) való konvolúciójának a deríváltja egyenlő a kép és a Gauss-függvény deriváltjának a konvolúciójával. Ezek alapján az éldetektálás a koncepciója következő:

1. Konvolváljuk f -t g' -vel.
2. Számítsuk ki h' abszolút értékét.
3. Definiáljuk éleknek minden pontnak a helyeket, ahol a h gradiensének értéke meghaladja előre meghatározott küszöbértéket.

A Canny-féle éldetektor nem érzékeny az élek állására, minden élt helyesen detektál, egy élt egy vonallal rajzol meg. Lássuk kissé részletesebben a működését:

$$\text{grad}f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) = (f_x, f_y)$$

ahol (f_x, f_y) a kép gradiense az (x, y) pontban,

$$M(x, y) = \sqrt{f_x^2 + f_y^2}$$

ahol $M(x, y)$ az él erőssége az (x, y) pontban, valamint

$$\Theta(x, y) = \arctan \left(\frac{f_x}{f_y} \right)$$

az (x, y) pontban vett derivált irányvektora, az él normál-vektora.

Az f_x, f_y értékeket közelíthetjük úgy, hogy az x és y irányú izotropikus gradiens szűrővel vett konvolúcióját számoljuk a képnek az adott pontban, majd ebből kiszámoljuk minden pontban az $M(x, y)$ értékeit. Ezen a képen az élek látszanak, de minden él több pixel vastag, hiszen a képeken az élek általában nem tökéletesek, valamint ezzel a módszerrel még egy tökéletes él szomszédságában is 0-nál nagyobb értékeket kapunk.

Ezt a problémát oldja meg a nem-maximális élek kiküszöbölése. Ezt úgy lehetjük meg, hogy az $M(x, y)$ -t lemásoljuk egy kimeneti képbe, a $\Theta(x, y)$ -ban adott normálvektor által meghatározott szögben lépünk minden pontnak irányba az $M(x, y)$ képen, és ha bármelyik intenzitásérték nagyobb az aktuálisnál, akkor töröljük a pixelt a kimeneti képen. A nem-maximális élek kiküszöbölésének eredményeképp egy olyan képet kapunk, amelyen minden él rajta van, és mindegyik egyetlen vonalként jelenik meg.

Mivel így minden él, még a leggyengébbek is rajta lesznek a kimeneti képen, szükségünk lehet arra, hogy a gyengébb éleket eltüntessük és az erősebb, globális éleket tartsuk meg. Ha valamilyen küszöbölési eljárást használnánk, akkor az nem lenne tekintettel az élek folytonosságára, mi pedig nem szeretnénk, ha az élek megszakadnának. Erre a megoldást az élküszöbölés jelenti. Ennek alapötlete, hogy egy határ alatt minden pontot hagyunk el, egy határ fölött minden pontot tartunk meg, a köztes pixeleket pedig aszerint tartsuk meg, hogy eljutunk-e belőle biztosan jó pixelbe köztes pontokon. Ehhez a legjobb, ha mélységi bejárás egy változatát implementáljuk, kiegészítve azzal, hogy egy biztosan jó pontba érve az egész útvonalon megtartjuk a pixeleket.

A Canny-szűrő ideális olyan esetekre, mikor additív, Gauss-típusú zaj található a képen. Egy egyszerű közelítő módszer, hogy Gauss-szűrővel simítják el a zajokat a képen, és ezután alkalmazzuk a gradiens maszkját. Mivel a két szűrő lineáris, ezért a szűrés megvalósítható egyetlen lépésben, amint azt a szűrő koncepciót bemutató bekezdésben vázoltuk.

3.2.6. Kép élesítése

Az eddig tárgyalt szűrők segítségével lehetséges a képek élesítése, minőségének javítása. Működésének alapelve, hogy a kép eredeti és simított változatának különbségét hozzáadja az eredeti képhez:

$$g(x, y) = f(x, y) + [f(x, y) - f_s(x, y)]$$

ahol f_s a simított képet jelenti.

3.3. Nemlineáris szűrők

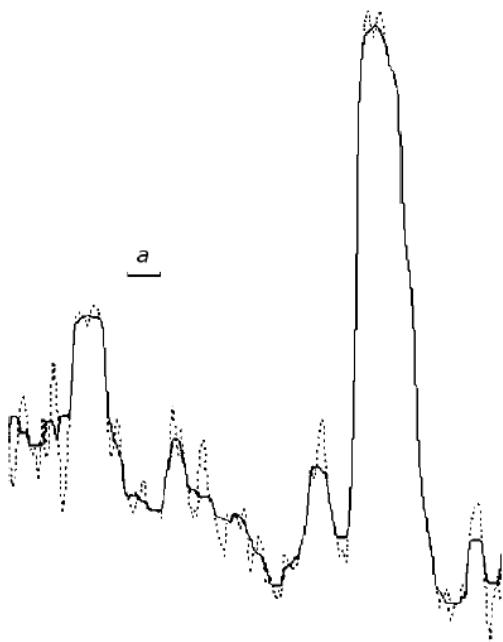
A nemlineáris szűrők azok az eljárások, amelyek ugyancsak egy adott pixel szomszédos pixelei alapján számolják ki a szűrt értéket, de nem a szomszédos pixelek értékeinek valamilyen lineáris kombinációjaként, hanem más módon. Itt nem beszélünk kernelmátrixról, hanem csak kernelről, vagy kernelablakról.

3.3.1. Rangszűrők

A rangszűrők alapvető ötlete az, hogy a kernelablak alá eső pixelek intenzitásértékeit állítsuk nagyság szerint sorba, majd ez alapján a sorrend alapján válasszunk új intenzitásértéket a szűrendő pixelnek. A leggyakrabban használt rang szűrő a medián szűrő, mely a nagyság szerinti középső értéket választja szűrt értékül (48. ábra).

A szűrés eredménye valamiféle simítás, amelyhez azonban átviteli függvény nem rendelhető. A szűrő a lokális zajokat hatékonyan eliminálja. A „salt and pepper” típusú hibákat (kisméretű, pontszerű érték kiugrások) eredményesen eltünteti, mert amikor egy ilyen pixelhez érünk, a környező pontok színétől kiugróan eltérő (sötét vagy világos) színű pontokat a rendezett kernel szélére sorolja. Az 47. ábra a medián szűrőnek egy zajos görbüre gyakorolt hatását mutatja.

Digitális képekre a medián szűrő fontos tulajdonsága, hogy $2k+1$ méretű kernel esetén a k -nál vékonyabb vonalakat eltünteti a képről. Ez kívánatos eredmény, amikor a nagy területeket próbáljuk kiemelni. Sajnos az éleket eltolhatja és a sarkokat lekerekíti, de az algoritmus kiegészíthető úgy, hogy ez a hiba ne forduljon elő.



47. ábra. Egy zajos görbe (szaggatott vonal) és medián szűrt változata (folytonos görbe). A kernel hossza a

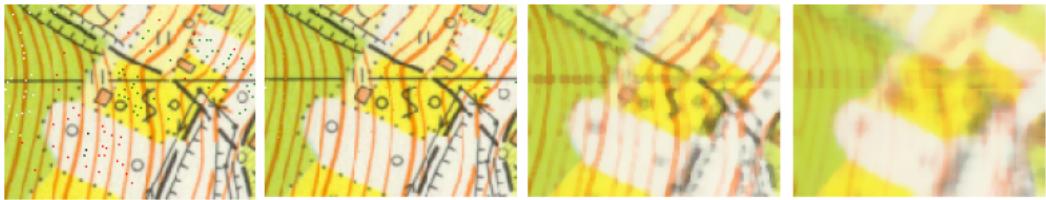
3.3.2. Olimpiai szűrő

Az olimpiai szűrő, a medián szűrőhöz hasonlóan, a kiugró intenzitás értékeket zajforrásból származónak tekinti. Egyes sportok olimpiai pontozási módszerét követi. Sorba rendezi a kernel alatti elemeket, majd a középsőtől legjobban elütőket eldobja. Paraméterként megadható, hogy a legnagyobb és legkisebb elemkből mennyit hagy figyelmen kívül.

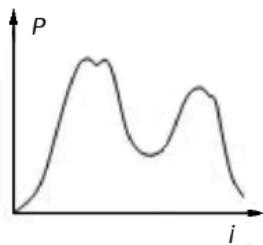
3.3.3. Konzervatív simítás

A konzervatív simítás zajszűrő eljárás, mely leginkább a „salt and pepper” típusú zajt képes eliminálni. Stratégiája, hogy a kernelablakba eső pixeleket nagyság szerint sorba rendezi az aktuális pixel kivételével. Így kapunk egy $[min..max]$ intervallumot, és megnézzük, hogy az aktuális pixel ebbe az intervallumba esik-e (48. ábra).

- ha a $[min..max]$ intervallumba esik, akkor nem változtatunk a pixel intenzitásán
- ha a maximumnál nagyobb, akkor az új érték a maximum lesz
- ha a minimum alá, akkor az új érték a minimum lesz.



48. ábra. Balról jobbra: Az eredeti kép, „salt and pepper” típusú hibákkal terhelve. Konzervatív simítással eltüntetve a hibák. 5×5 méretű mediánszűrővel tisztított kép. 11×11 méretű mediánszűrővel szűrt kép



49. ábra. A kép hisztogramja két csúcsú. A jó szegmentálás a csúcsoknak megfelelő osztályokat állítja elő

3.4. Szegmentálás, küszöbölés

A küszöbölés a szürkeárnyalatos képek szegmentálásának egyik módja. Ilyenkor megadunk néhány küszöbértéket, amelyek intervallumhatárokat fognak jelölni. A küszöbölés speciális esete a kétszintes küszöbölés, a binarizáció, amikor egyetlen küszöbértéket adunk meg, így a pixeleket két osztályba soroljuk.

A küszöbérték meghatározására több stratégia létezik, attól függően, hogy milyen célt tűztünk ki, mi a mértéke annak, hogy mennyire jó egy küszöbérték. Általában azt szeretnénk, ha a képen az objektumok jól eltérjenek a háttérüktől. Mivel két osztályba sorolhatunk be minden pixelt, ezért ez nem sikerülhet minden tökéletesen, de a jó küszöbölés ezt a lehető legjobban közelíti.

3.4.1. Otsu-féle küszöbölés

Tekintsük meg az 49. ábrát, amely egy kép hisztogramját mutatja. Látható, hogy az eloszlás két intenzitás érték körül csoportosul, vagyis a hisztogram két csúcsú. A cél a kép szegmentálása, mégpedig úgy, hogy az intenzitáseloszlás csúcsainak megfelelő osztályok jöjjenek létre.

Otsu szerint az a jó osztályozási eredmény, ha a két osztály közötti szórás a lehető legnagyobb. Ehhez kiszámolja a kép pixeljeinek empirikus várható értékét és szórásnégyzetét.

$$\begin{aligned}\mu &= \sum_{i=0}^{255} iP(i) \\ \sigma^2 &= \sum_{i=0}^{255} (i - \mu)^2 P(i)\end{aligned}$$

Egy t küsszöbérték mellett az egyes osztályokon belüli szórás és várható érték:

$$\begin{aligned}\mu_1(t) &= \frac{1}{q_1(t)} \sum_{i=0}^t iP(i) \\ \sigma_1^2(t) &= \sum_{i=0}^t (i - \mu_1)^2 P(i)\end{aligned}$$

valamint

$$\begin{aligned}\mu_2(t) &= \frac{1}{q_2(t)} \sum_{i=t+1}^{255} iP(i) \\ \sigma_2^2(t) &= \sum_{i=t+1}^{255} (i - \mu_2)^2 P(i)\end{aligned}$$

ahol

$$q_1(t) = \sum_{i=0}^t P(i)$$

és

$$q_2(t) = \sum_{i=t+1}^{255} P(i).$$

Az osztályokon belüli szórás a két osztály szórásának súlyozott összege, azaz

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Az osztályok közti szórást a következőképp definiáljuk:

$$\sigma^2 = \sigma_w^2(t) + \sigma_b^2(t)$$

vagyis

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t).$$

Fejezzük ki $\sigma_b^2(t)$ -t:

$$\sigma_b^2(t) = q_1(t)q_2(t)(\mu_1(t) - \mu_2(t))^2 = q_1(t)(1 - q_1(t))(\mu_1(t) - \mu_2(t))^2$$

Az optimális szórás számunkra az, ami a lehető legjobban elkülöníti az osztályokat. Mivel a kétféle szórás összege egyenlő a teljes szórással, ezért két, egymással

ekvivalens célunk lehet az optimum megtalálásához: minimalizáljuk az osztályokon belüli szórást, vagy maximalizáljuk az osztályok köztit. Ha az utóbbit választjuk, akkor az egyes t értékekre a $q_1(t+1), \mu_1(t+1), \mu_2(t+1)$ értékeket számolhatjuk a $q_1(t), \mu_1(t), \mu_2(t)$ értékek felhasználásával:

$$\begin{aligned} q_1(0) &= 0 \\ \mu_1(t+1) &= \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)} \\ \mu_1(0) &= 0 \\ \mu_2(t+1) &= \frac{\mu - q_1(t+1)(\mu_1 + 1)}{1 - q_1(t+1)} \end{aligned}$$

Ily módon a következő algoritmust kapjuk:

1. számoljuk ki P -t, μ -t és σ -t
2. $t = 0$ -tól 255-ig számoljuk ki minden értékre $q_1(t), \mu_1(t), \mu_2(t)$ értékeket, majd ebből a σ_b^2 értéket
3. válasszuk $t_{optimal}$ -nak az $argmax(\sigma_b^2)$ -t

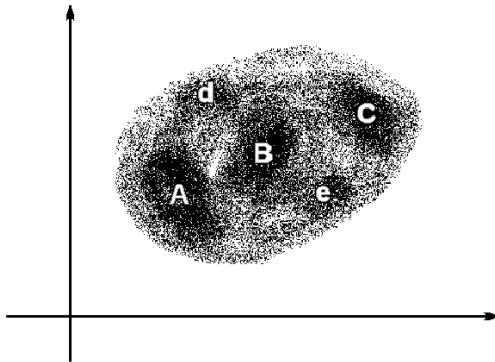
Az Otsu-féle küszöbölés eredményeit mutatja a 49. ábra. Az Otsu-féle küszöbölés általánosítható, azaz több szintű küszöbölést is lehet ezzel a stratégiával előállítani.

4. Osztályozás, klaszterezés

A nagy tömegű adathalmazokban való eligazodás meglehetősen bonyolult feladat, hiszen soha nem látott méretű adatbázisok jöttek létre és folyamatosan jönnek létre. Az adatok keresése, legyűjtése mögött gyakran valamilyen interpretációs szándék húzódik meg, amelyre az adatok szegmentálása, (tematikus) csoportokba foglalása teremti meg a lehetőséget. Mint tudjuk, az adat a gondolkodó ember fejében válik információvá, és ezt a folyamatot nagymértékben elősegítheti az adatok csoportosítása, tekintve, hogy javítja az áttekinthetőséget.

Az informatika egyik modern ága az adatbányászat, amely nem kisebb feladatot tűzött ki, mint a nagy méretű adatbázisokban való eligazodást. A geoinformatikát a szakmai zsargon nem sorolja az adatbányászat témakörébe, pedig az adatbázisok mérete, az adatok sokfélesége, és a grafika teremtette nehézségek ezt akár indokolhatnák is. Nem véletlen, hogy a térinformatikában, különösen a riaszteres adatmodellt követő esetekben, mint amilyen az ūrfotók feldolgozása, az adatbányászatban kiemelkedően fontos eljárás, a klaszter analízis, fontos szerepet játszik.

Kétdimenziós esetben ábrázolva az adatpontokat, már szemrevételezéssel is el tudunk különíteni csoportokat az adatok sűrűsödése alapján (50. ábra).



50. ábra. Csoportok egy kétdimenziós képen

Egy adathalmaz pontjainak az adatrekordok hasonlósága alapján történő diszjunkt csoportokba sorolását klaszterezésnek nevezük. A csoportosítás jósága alapvetően két dolgon műlik: a jó hasonlóság definíció és egy jó algoritmuson, amely a hasonlóságon alapulva valamilyen kritériumok alapján megállapítja a klaszterekeket. Sokszor használjuk az osztályozás kifejezést is, ami majdnem ugyanazt jelenti, mint a klaszterezés. Míg a klaszterezés nem felügyelt csoportosítás, addig az osztályozás felügyelt. Ebben az összefüggésben a felügyelt jelző azt jelenti, hogy a csoportok minőségi paraméterei előre definiáltak, míg a nem felügyelt esetben nem tudjuk, hogy milyen minőségi osztályba fognak tartozni az előálló csoportok, sőt ezek határai sem tudhatók előre.

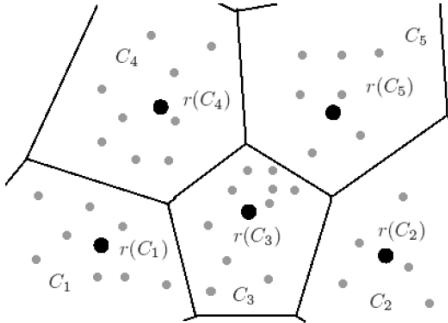
A hasonlóság definiálásának egy kézenfekvő módja az euklideszi távolság fogalom. Jelölje u_i, v_i az adatpontokat, és $d(u, v)$ az adatpontok közötti távolságot.

$$d(u, v) = \sqrt{\sum_{i=1}^d (u_i - v_i)^2}$$

Írjuk be az összes lehetséges adatpont közötti távolságot egy mátrixba, amelyet távolság mátrixnak nevezünk:

$$\mathbf{D} = \begin{pmatrix} d_{11} & d_{12} & \dots \\ d_{21} & d_{22} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Első közelítésben azt mondjuk, hogy egymáshoz hasonló pontokat azonos csoportba, klaszterbe sorolunk. A távolság mátrix alapján viszont ismerjük az adatpontok páronkénti távolságát, így a távolságok alapján a hasonlóságra is következtetni tudunk. Azt mondjuk tehát, hogy azonos klaszterbe tartozó pontok egymás-hoz közel vannak.



51. ábra. Adatpontok (szürke körök), klaszter középpontok (fekete körök) és klaszterhatárok

Ez a megfogalmazás elég tág határokat enged meg a klaszterek meghatározására, de valamennyi klaszterező eljárás hátterében megtalálható a távolság mátrix, illetve a klaszterek középpontjától való távolság.

4.1. Particionáló klaszterezés

Feltessük, hogy a klaszterek egy vektortérben helyezkednek el. A klasztereket súlypontjukkal reprezentáljuk, vagyis a klaszterekhez tartozó adatpont-vektorok átlagával jellemzzük (51. ábra). Az algoritmus olyan C klaszter beosztást keres, ahol az adatpontok saját klaszterük $r(C_i)$ súlypontjától mért távolságának négyzetösszege minimális.

$$E(C) = \sum_{i=1}^k \sum_{u \in C_i} d(u, r(C_i))^2$$

Általában előre meg kell adnunk egy k klaszterszámot (vagyis, hogy hány csoportra szeretnénk bontani az adathalmazt). Válasszunk ezután k darab adatpontot. Ezután minden adatpontot a hozzá legközelebb eső klaszter-súlyponthoz tartozó klaszterbe sorolunk. A besorolás eredményeként kialakult új klaszterek súlypontjai lesznek az új klaszterek reprezentáns pontjai. A besorolás, súlypontszámítás lépéseiit addig végezzük, amíg a súlypontok rendszere változik. Akkor állunk meg, amikor a klaszterek elemei és a klaszterek középpontjai már nem változnak az iteráció hatására.

4.2. Hierarchikus eljárások

A hierarchikus klaszterező eljárásokban a adatokat hierarchikus adatszerkezetbe (fába, dendogram) rendezzük. Az adatpontok a fa leveleiben helyezkednek el. A fa

Két alapvető hierarchikus eljárás létezik: az egyik a felhalmozó, a másik a lebontó. A felhalmozó eljárásban kezdetben minden adatelem egy klaszter, majd a legközelebbi klasztereket egyesíti az algoritmus, és a hierarchiában egy szinttel feljebb új klasztert alakít ki.

A lebontó eljárásban kezdetben egyetlen klaszter létezik, amelybe minden adat-pont beletartozik, majd ezt tovább osztjuk. Az újabb klaszterek az előző finomításai lesznek. Az eljárások akkor állnak meg, amikor vagy elérnek egy előre megállapított klaszter számot, vagy a klaszterek közötti távolság egy előre megállapított mértéknél kisebbé válik.

4.3. Képek klaszterezése

A képek osztályozásakor az a célunk, hogy a pixeleket tematikus kategóriákba soroljuk az intenzitás értékeik alapján, mintegy spektrális osztályokat létrehozva. Kétféle osztályozási módszert különböztetünk meg. Az egyik a nem felügyelt (unsupervised classification), a másik a felügyelt (supervised classification) osztályozás. A nem felügyelt esetben megelégszünk spektrális csoportok létrejöttével, amelyeket megkísérlünk megfeleltetni valamely tematikus kategóriának.

A klaszterezéskor kiindulhatunk fix számú osztályból is, de megadhatunk egy környezetet is, például euklideszi távolság alapján, amelynek túllépése esetén új klaszter jön létre.

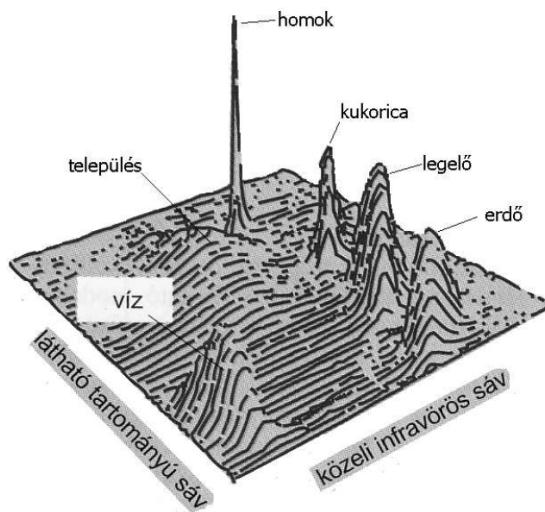
4.3.1. ISODATA eljárás

Az ISODATA eljárás a klaszterek középpontjait keresi meg. Az eljárás a következő módon működik:

1. Válasszunk ki klaszter középpontokat kiindulásul
2. A pixeleket a hozzájuk legközelebb eső középpontú klaszterbe soroljuk
3. Az újfent besorolt pontok figyelembe vételével kiszámítjuk az új klaszter középpontokat, amik ettől kisebb nagyobb mértékben megváltoznak
4. Az eljárás leállását a középpontok mozgása határozza meg. Addig folytatjuk az eljárást (a 2. pontra ugorva), amíg a középpontok helyzete nem változik, pontosabban a mozgásuk egy bizonyos küszöbérték alatt marad

4.3.2. Felügyelt osztályozás

Felügyelt osztályozáskor a kép pixeljeit tematikus kategóriákba soroljuk a tematikus kategóriák mintáiból gyűjtött adatok alapján (vagyis előre tudjuk, hogy hány osztályunk van, és azoknak mik a minőségi jellemzői). A tematikus kategóriák



52. ábra. A különböző anyagok reflektanciáinak eltérése alapján következtetni lehet az anyagminőségre, feltéve, hogy a tematikus osztályok között nincs átfedés

mintaterületeinek kijelölése történhet terapii bejárás alapján, vagy független, más forrásból származó adatok alapján vizuális interpretációval. A mintaterületeket gyakran nevezzük tanuló területnek.

Összehasonlítva a kétféle osztályozási módot látható, hogy a felügyelt osztályozáskor a tematikus kategóriák meghatározása után osztályozzuk a képet, míg a nem felügyelt osztályozáskor a klaszterezés után feleltetjük meg az egyes klasztereket a tematikus kategóriáknak.

Ezen osztályozások fizikai hátterét az a megfigyelés adja (amelyet akár a távérzékelés alapösszefüggésének is nevezhetünk), hogy egyes tematikus osztályok, minőségi kategóriák pixeljei jellegzetes csoportokat alkotnak, amint azt a kategóriák reflektancia értékeinek eloszlása is jól mutatja a 52. ábrán. (Feltételezhetően annak tudható be a normális eloszlás, hogy a visszaverődés jelensége sokféle folyamat együtteséből tevődik össze. Márpedig ezek akármilyen eloszlást is kövesssenek, az összegük eloszlása közelíteni fog a standard normális eloszláshoz. Ez a centrális határeloszlás tétele.)

A tapasztalat azt mutatja, hogy a különböző frekvenciasávokra másként reagálnak ezen minőségi kategóriák anyagai, így minél több frekvenciasávban állnak rendelkezésünkre képek, annál több minőségi kategória megállapítása válik lehetővé. Ez a körülmény teremti meg az értelmét a több száz frekvenciasávban működő hiperspektrális távérzékelés számára.

5. Szegmentálás

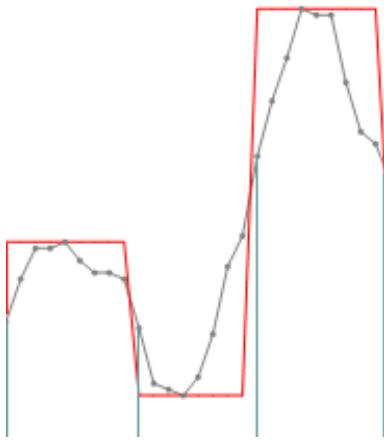
Egy kép azonos tulajdonságú pixeljeinek homogén területekre történő felbontása, csoportosítása a szegmentálás. Fontos megkülönböztetni a klaszterezéstől, noha látszólag hasonlóak. Míg a klaszterezés eredménye az ugyanabba a csoportba tartozó pixeleket, akkor is egy csoportba sorolja, ha azok területileg diszjunkt elhelyezkedésűek. A szegmentálás azonban különálló szegmensként kezel diszjunkt elhelyezkedésű, de amúgy a jellemzők alapján azonos minőségű területeket.

Jelentősége kiemelkedő a számítógépes látás terén, de hasonlóan fontos a távérzékelésben is. Mindkét terület a felületek leírására, és alakfelismerésre vagyis a kép kiértékelésére használja. A képek feldolgozásának három fő szintjét különböztetjük meg:

1. A kép tulajdonságainak meghatározása: éldetektálás, spektrális jellemzők, textúra. E feldolgozás eredménye többnyire egy újabb kép, amely az eljárásainkkal kinyert tulajdonságokat (is) tartalmazza.
2. A képen látható objektumok tulajdonságainak meghatározása. A kiindulási adatok az előző feldolgozási fázis eredményei. A folyamat eredményeként a kép tartalmának egy kezdetleges, szimbolikus leírását kapjuk, amely főleg a képen látható alakzatok jellemzőit foglalja magába (felületek, kontúrok, stb.).
3. A kép értelmezése, vagyis a képen előbb előállított objektumok felismerése

Kétféle megközelítéssel élünk:

1. Globális eljárások:
 - Thresholding (küszöbölés) hisztogram alapján
2. Lokális eljárások
 - Határvonalak detektálása éldetektálással
 - Homogén régiók detektálása
3. Homogén régiók keresése
 - Leggyakrabban intenzitás (szürkeségi szint) alapján
 - Színelemzés alapján
 - Textúra elemzés alapján



53. ábra. Az intenzitás görbék inflexiós pontjaiban jelöljük a szegmenshatárokat. Szegmenshatáron belül egy jellemző értéket rendelünk a szegmenshez, ami a két szegmenshatár közötti szupréumum értéke (minimum vagy maximum), kivételes esetben a két határ közti átlagos intenzitás érték. A szürke vonal az intenzitás görbe, a világoskék függőleges vonalak a szegmenshatárok, a piros görbe pedig a szegmensek értékeit mutatja

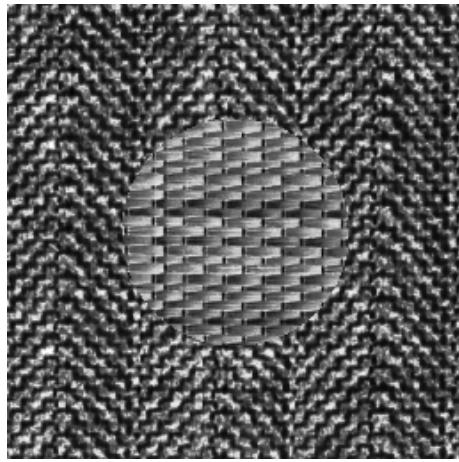
5.1. A Giwer rendszer éldetektálási módszere

A *Giwer* rendszer egy éldetektáláson alapuló szegmentálási eljárást fog használni, amely elsődleges élek detektálása után, azok további szelekciójával fog előállítani szegmenshatárokat. A szegmensek értékei ezeken a határokon belüli pixelek értékei alapján lesznek kiszámítva.

Szegmenshatár ott van, ahol a legmarkánsabb éleket detektáljuk az intenzitás görbén (53. ábra). A határok között azonos értéket adunk meg a szegmens intenzitás értékére, amely a határok közti intenzitásértékek szupréumuma. Ennek akkor van jelentősége, amikor nagy felbontású képekre kívánunk osztályozást végrehajtani, ugyanis a pixelekre alkalmazva a klaszterezést igen nagy futásidők állhatnak elő az adatrendszer óriási mérete miatt. Ha ugyanezt a klaszterezési módszert a szegmensekre hajtjuk végre, akkor nagyságrendileg kisebb adatmennyiséggel nézünk szembe. A távérzékelési szakértők véleménye szerint igen nagy felbontású képek esetén nem tulajdonítható érdemi fizikai jelentés egyetlen pixel értékének, sokkal inkább a szegmenseknek.

5.1.1. Soksváros képek szegmentálása

A hagyományos multispektrális képek mellett egyre gyakoribbak a hiperspektrális képek, amelyek akár több száz frekvenciasávból állhatnak. Ezekre alkalmazva az osztályozó eljárásokat lehetetlen mértékű futásidőket kapunk. Ennek elkerülése érdekében kétféle eljárást alkalmazunk.



54. ábra. Példa szabályos textúrájú poligonokra. A poligonok spektrális jellemzők alapján nem különböztethetők meg.

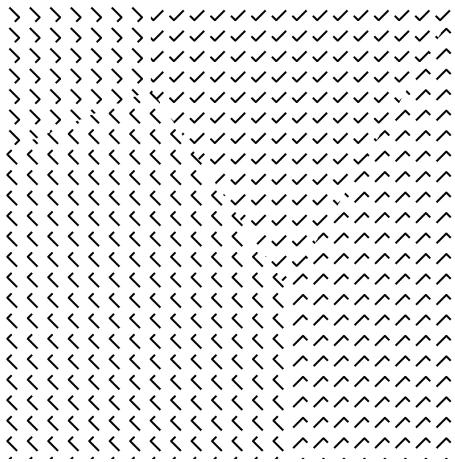
- Kiszámítjuk az összes (vagy egy részhalmaz) frekvenciasáv első főkomponensét, amely a kép össz-varianciája legnagyobb közös részét tartalmazza, és ezzel az egy „sávval” helyettesítjük az egész képet. Ezzel voltaképpen irányított, optimális adatvesztést hajtunk végre, cserébe viszont jelentősen csökkentjük az adatrendszer dimenziószámát.
- Az így kiszámított első főkomponensre alkalmazzuk a fent ismertetett szegmentáló eljárást, majd a szegmensekre hajtjuk végre az osztályozást.

6. Textúra elemzés

A képek szegmentálása (vagyis kvázi homogén területekre bontása) nemcsak az intenzitás értékek feldolgozásával oldható meg. Amikor spektrális tulajdonságok (pl. színkontrasztok) alapján nem különböztethetők meg szomszédos területrészek, olyankor használjuk a textúrális jellemzőket szegmentálásra.

A 54. és 55. ábrákon szabályos textúrák láthatók. Mindkét képre jellemző, hogy a különböző poligonok intenzitás eloszlása megegyezik. Míg a 54. ábrán azonnal felismerhetők a poligonhatárok, addig a 55. ábrán már korántsem ilyen egyszerű a helyzet. Hosszabb nézegetés után ismerhetők csak fel egyes határok.

Érdekes, hogy egyes alakzatokból álló poligon textúrákat azonnal felismerünk, míg mások esetén csak hosszasabb szemrevételezés után vagyunk képesek észrevenni az eltérő textúrákat. Ebben a kérdéskörben alapvető eredményeket ért el egy híres, magyar származású amerikai fizikus, Julesz Béla ([3]).



55. ábra. Példa szabályos textúrájú poligonokra. A textúrát ugyanannak az alakzatnak az elforgatott változatai adják.

Hivatkozások

- [1] Julius T. Tou, Rafael C. Gonzalez: Pattern Recognition Principles, Addison-Wesley, 1974
- [2] Távérzékelt felvételek elemzése, Egyetemi jegyzet, ELTE, 2011
- [3] Julesz Béla: Dialógusok az észlelésről, Typotex, 2000
- [4] Gonzalez - Woods: Digital image processing, Prentice-Hall, Third edition, 2008