TKP kutatási jelentés:

GeoImage Workflow Editing Resources

Elek István

2022. február 20.

Tartalomjegyzék

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 adatféleség, 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. régészet, nyelvészet) 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 hegemóniája (pl. ESRI, Integraph), és a szakterület eddig sosem látott mértékben az adatbázis-technológia világába betagozódott. Az okos telefonok előretörésével már a mindennapi élet részévé vált a digitális térkép, a Google térképkezelő funkcionalitása fejlesztéseinek köszönhetően már bármely felhasználó hozzáférhet villámgyorsan digitális térképi tartalmakhoz, legyenek azok vektortérképek, űrfelvételek, 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 térképészet 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, űrfelvétel, 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 elért 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, fiziológiáját 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 bármely térbeli referenciával rendelkező, űrből, 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 eljárás 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**). Noha a monolit alkalmazás keretében is megvalósítható ez a koncepció, én mégis inkább egy olyan megközelítést tervezek, amely microservices formában valósítja meg a rendszer működését, és amely mindenképpen korszerűbbnek tűnik, annak ellenére, hogy a működési sebességet tekintve egyáltalán nem biztos, hogy felülmúlja a monolit programrendszert. Úgy gondolom, hogy az osztott adatfeldolgozást is jobban támogatja a microservices alapú megközelítés. Sok, nagyméretű kép feldolgozásakor ennek nagy jelentősége lehet.

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 mindkettőt meg szeretném valósítani.

3.2. A desktop alkalmazás

Egy keretprogram vezérli a különböző programrészeket. Ez a Giwer nevű program. Célja a 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.

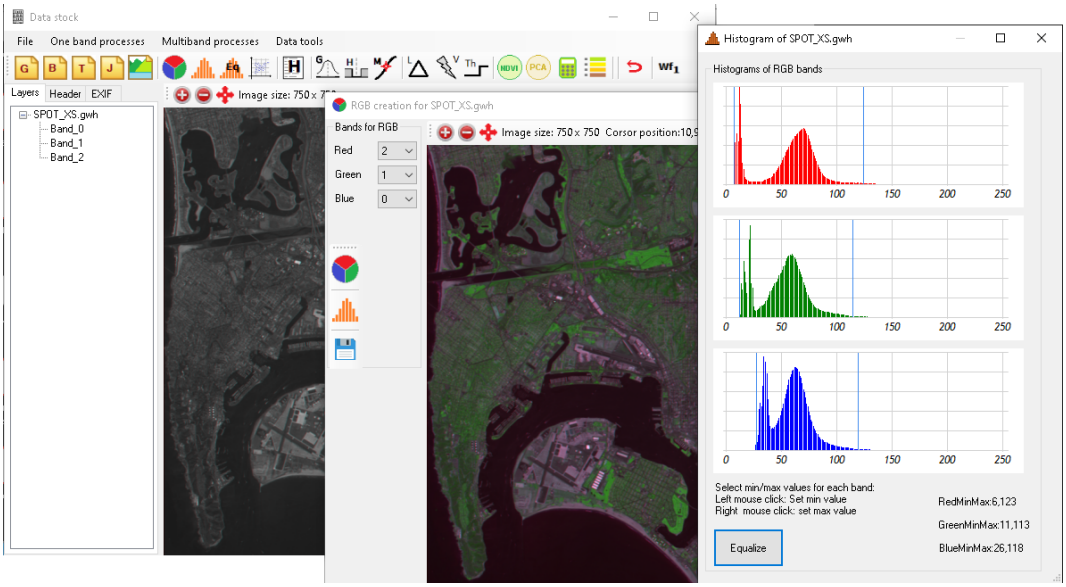
Itt indíthatjuk el a programok konfigurálását végző programrészt (*Config* icon), valamint a program használatát segítő leírást, végül pedig a program metaadatait bemutató információs részt (*About* ikon).

1. ábra. A Giwer keretrendszer

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



2. ábra. A Data stock program egy képernyő képe

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 alrendszert, amely adatbázisban tárolja, rendszerezi a képeket.

3.2.3. Workflow builder

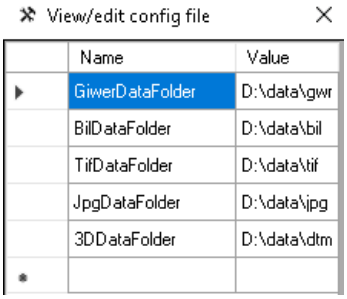
A *Workflow builderrel* a rendelkezésre álló függvényekből tetszőleges munkafolyamatot (workflowt) állíthatunk elő.

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 ak-



3. ábra. A Config viewer/editor képernyő képe

kor 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

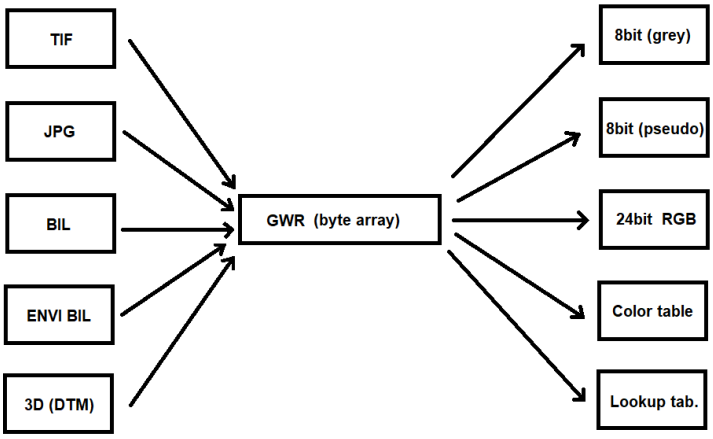
1. A Giwer logikai felépítése

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 mindenképpen 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* formátumra (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 képben.

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*) elkérési 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. Alapelv, hogy minden frekvenciasáv intenzitásértékeit egy-egy bináris fájlban tároljuk. A program ezeket az adatokat egy bytetömbben tartja, amivel gyorsan lehet műveleteket végezni. Mivel a képformátumok 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 adatszerkezeten végzünk 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, hisztogram 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

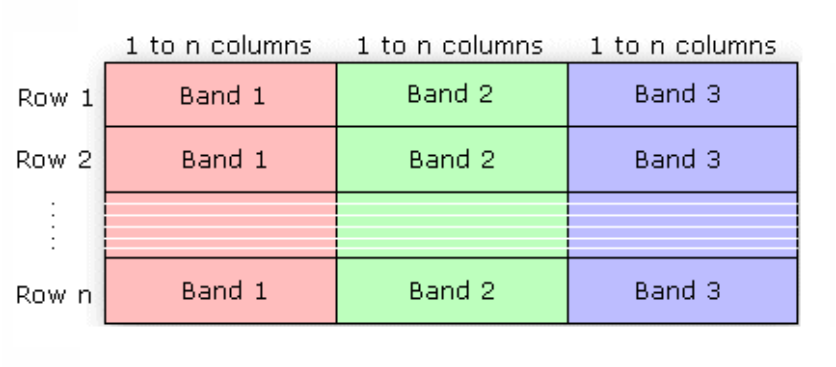
3.2. Támogatott adatformátumok

Háromféle raszteres 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 értékeket tárol, hanem a headerben a kép georeferencia adatait (vetületi rendszer, kalibrációs pontok /többnyire sarok pont koordináták/ egyéb a képalkotásra 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 iránnyal 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

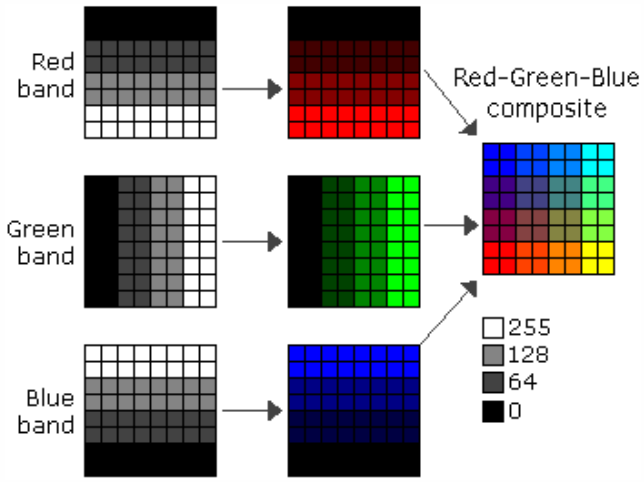
A jól ismert *jpg* formátumú képek beolvasását is támogatja a program. Ez többnyire nem tartalmat georeferencia adatokat. Bizonyos esetekben léteznek hozzá kiegészítő fájlok, amelyek tartalmaznak a vetületi rendszerre, kalibrációs pontokra vonatkozó adatokat. Drón képek esetén ilyen fájlok nincsenek, csak a hagyományos *jpg* fájl. Hasonlóan a tiff fájlokhoz, 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 soksávos űrfelvételek 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

```
0 0 0 0 0 0 0 0 0 0 64 64 128 128 255 255 255 255 255 255 255 255 255 255
0 0 0 0 0 0 0 0 0 0 64 64 128 128 255 255 255 255 255 255 255 255 255 255
64 64 64 64 64 64 64 64 0 0 64 64 128 128 255 255 128 128 128 128 128 128 128 128
64 64 64 64 64 64 64 64 0 0 64 64 128 128 255 255 128 128 128 128 128 128 128 128
128 128 128 128 128 128 128 128 0 0 64 64 128 128 255 255 64 64 64 64 64 64 64 64
128 128 128 128 128 128 128 128 0 0 64 64 128 128 255 255 64 64 64 64 64 64 64 64
255 255 255 255 255 255 255 255 0 0 64 64 128 128 255 255 0 0 0 0 0 0 0 0
255 255 255 255 255 255 255 255 0 0 64 64 128 128 255 255 0 0 0 0 0 0 0 0
```

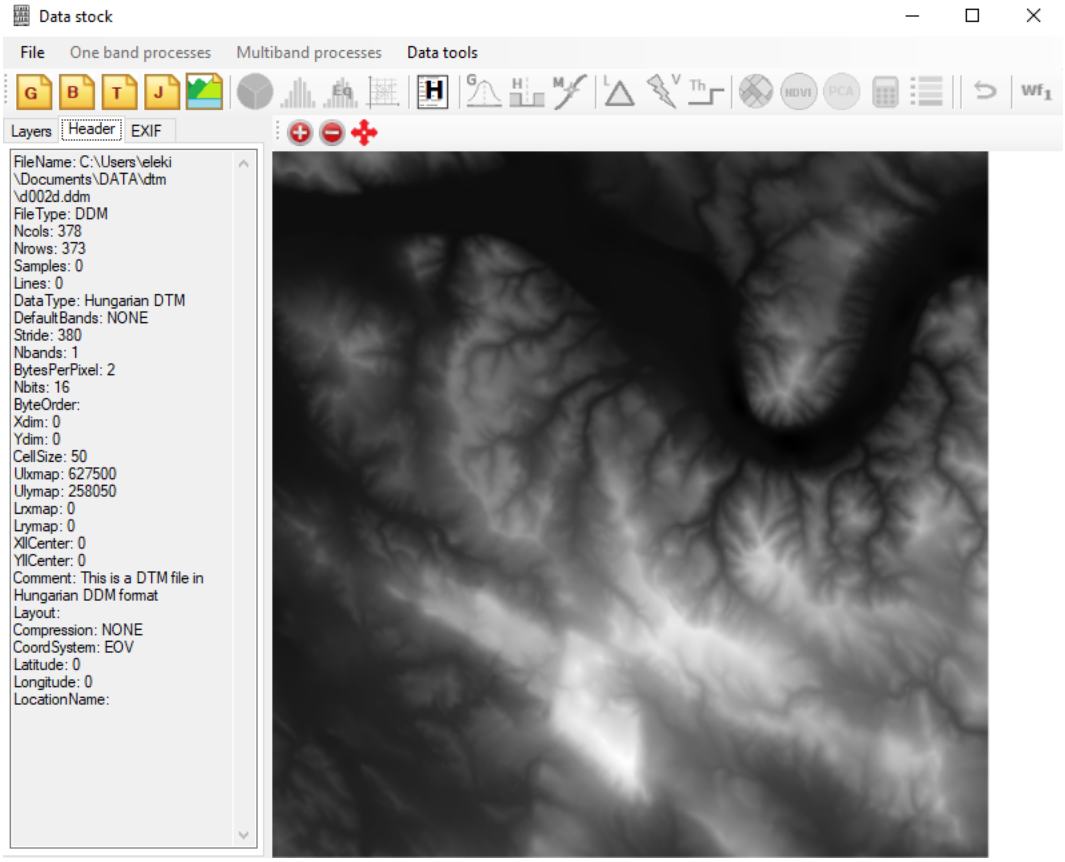


6. ábra. Egy képi példa *BIL* fájl adatstruktúrájára

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* sávonként 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 funkcionalitást:

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 konverziókhoz, manipulációkhoz 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 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 bytearray to a bitmap	byte[ ]	bitmap
bitmap2ByteArray	convert bitmap to a bytearray	bitmap	byte[ ]
getAnRGBBand	get a band data from a bytearray	which band	byte[ ]
getOneBand2Bytes	get a band data from a bytearray	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 bytearray	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 bytearray 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 bytearrays	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 *default*ra van állítva (greyscale), hanem *hipsometric*-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 valamekkora 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 képformátumok metaadatait beolvassa és tárolja. Egységesen, minden képhez ugyanazokat a propertyket, 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 kezelni 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 adatok 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 fájlt (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 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 mindhárom 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. getOneBandtoByteArrayFromBitmap

Ez a függvény egy bitmapből 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

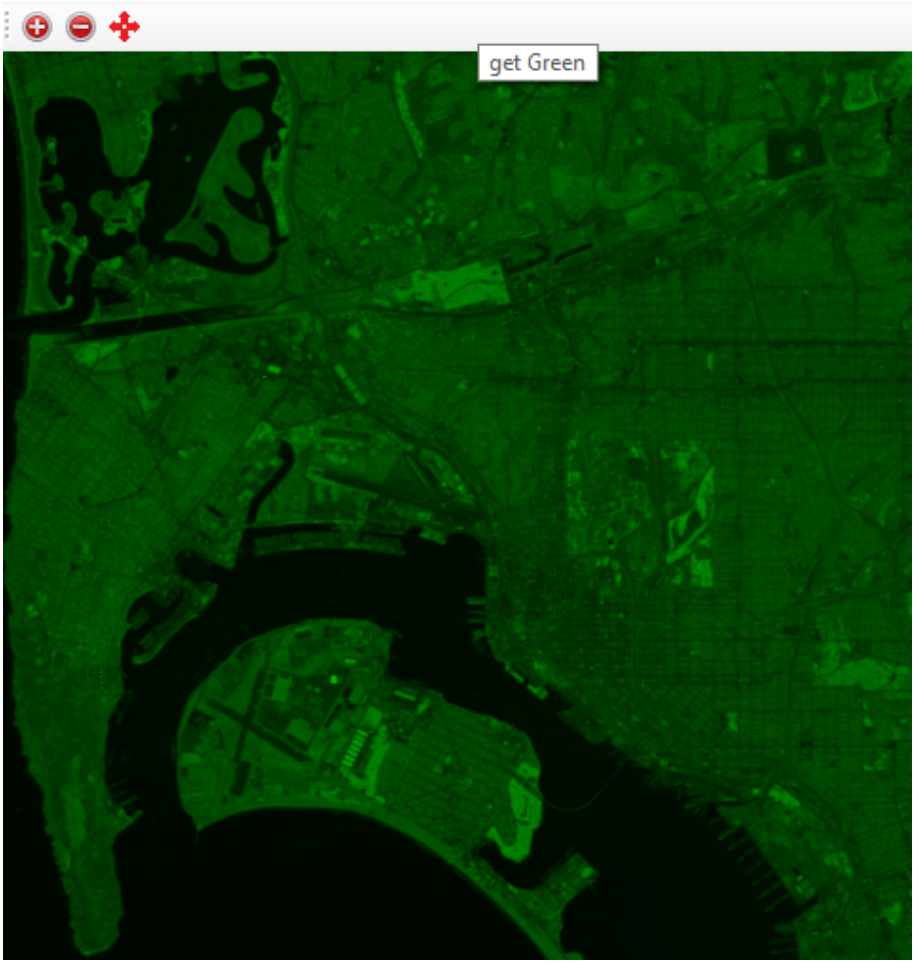
Ez a függvény invertálja at adott bitmap képét

4.2.12. GrayscaleConversion

Ez a függvény egy bitmapet szürkévé alakít.

4.2.13. readGwrFile

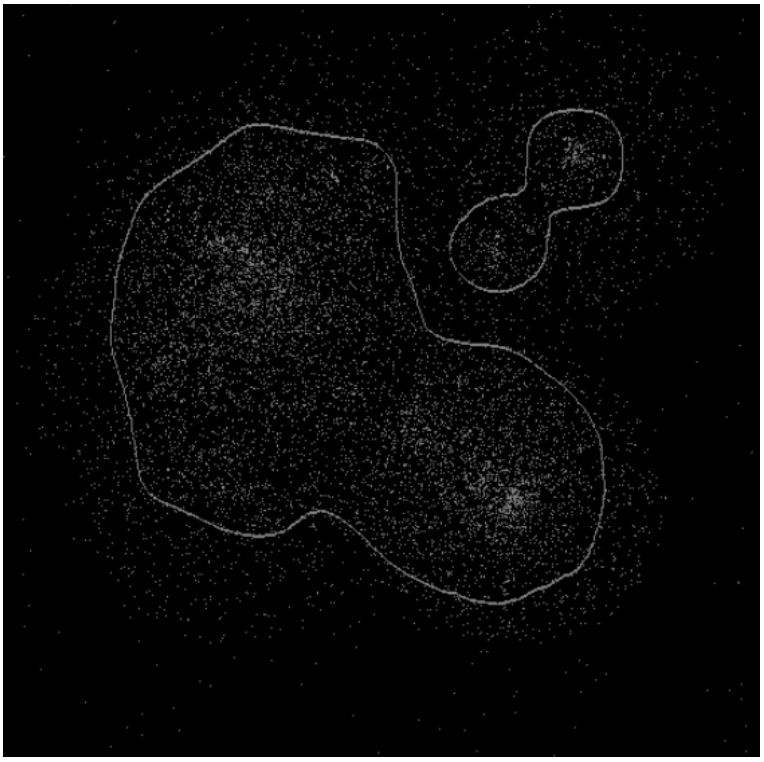
Ez a függvény beolvas egy giwer formátumú fájlt.



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. ConvolSingleBand

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

4.4.7. Convol3bands

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

4.4.8. highPassKernel

Ez a függvény felül áteresztő kernelt számol.

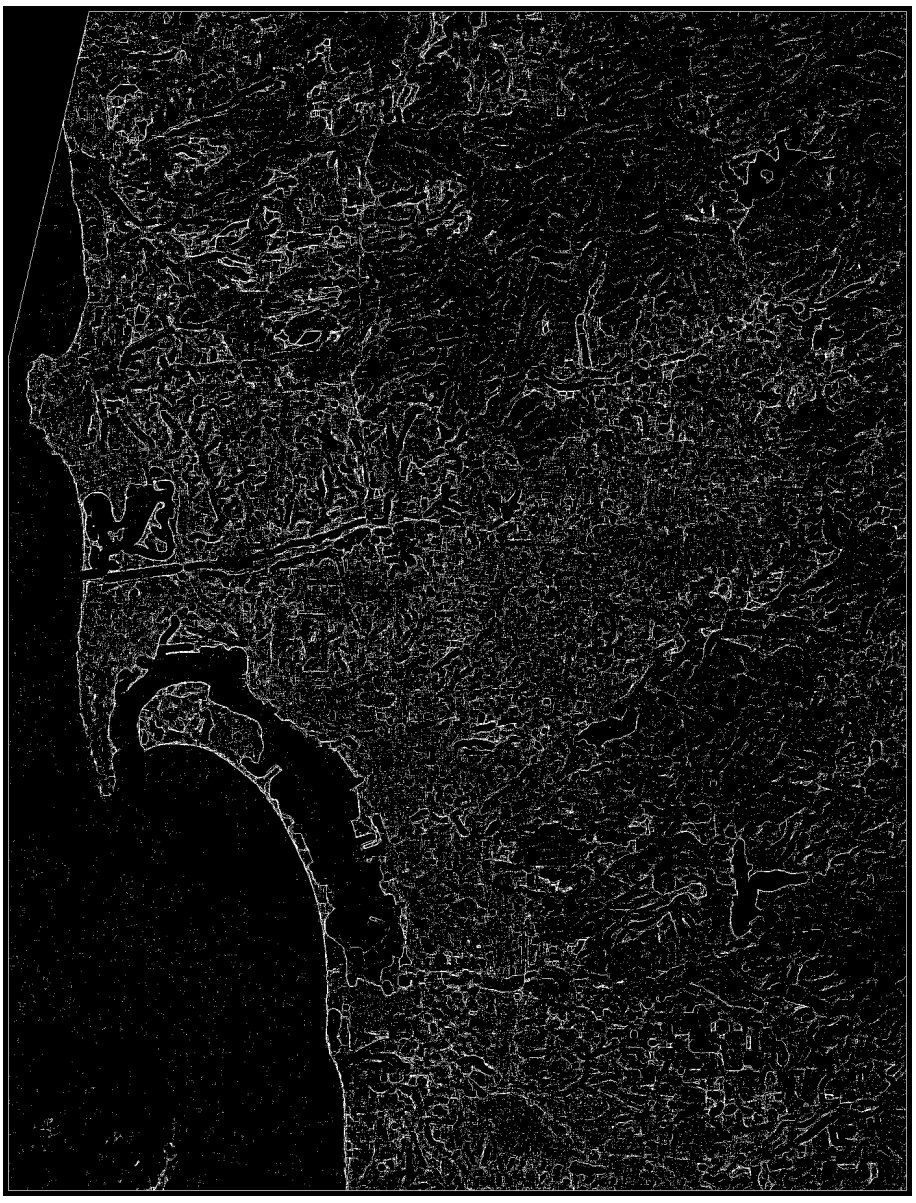
4.4.9. lowPassKernelGauss

Ez a függvény alul áteresztő kernelt számol.

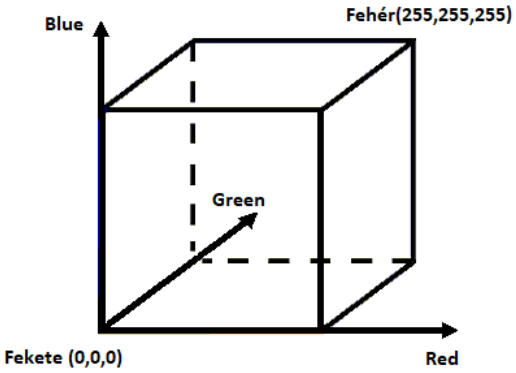
4.4.10. lowPassKernelBox

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 űrfelvételre. A medián szűrés jelentősen mértékben eltüntette a zajokat, így az élek már sokkal világosabban látszanak



12. ábra. Az RGB színkocka

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[] byInR, byte[] byInG, byte[] byInB, 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:

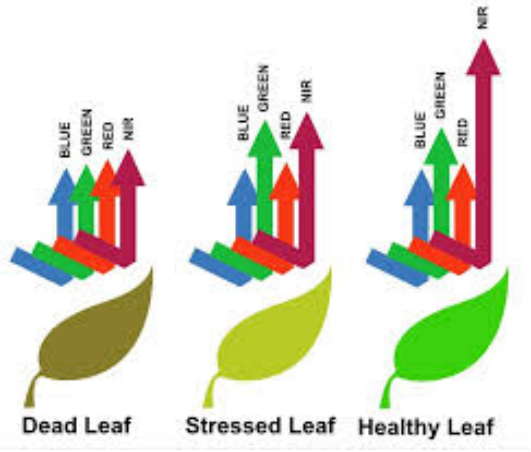
```
byte[] NDVI(byte[] bNIR, byte[] bIR)
```

4.5.3. Cross Plot

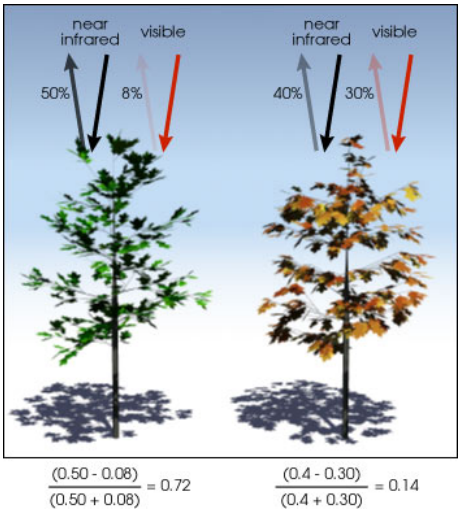
Az a függvény arra szolgál, hogy összehasonlítsuk két tetszőleges frekvenciasáv intenzitás értékeit. Az X-tengelyen az egyik, az Y-tengelyen a másik frekvenciasáv értékeit jelenítjük meg, ahogy azt a 16. ábrán láthatjuk.

4.5.4. compEigen

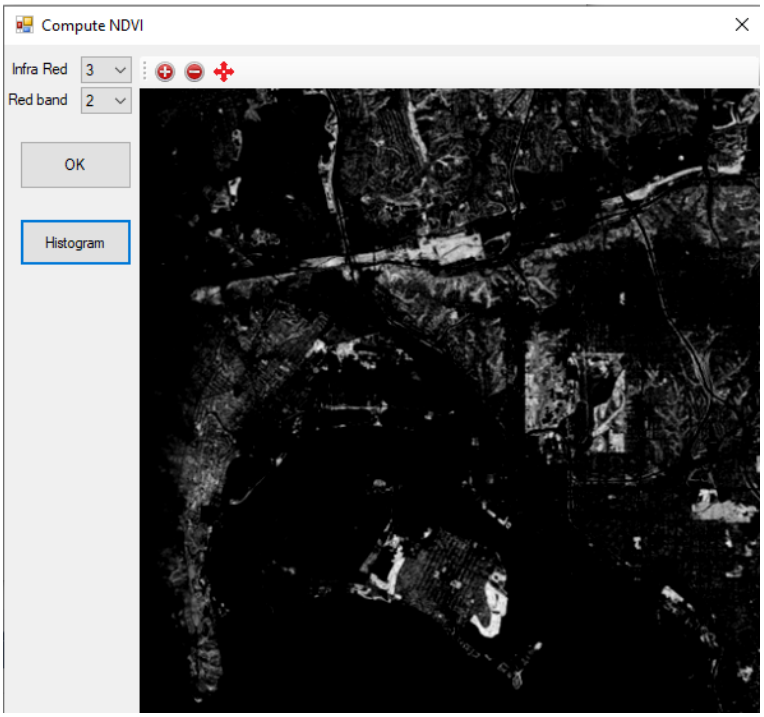
Kiszámítja egy megadott mátrix sajátértékeit és sajátvektorait.



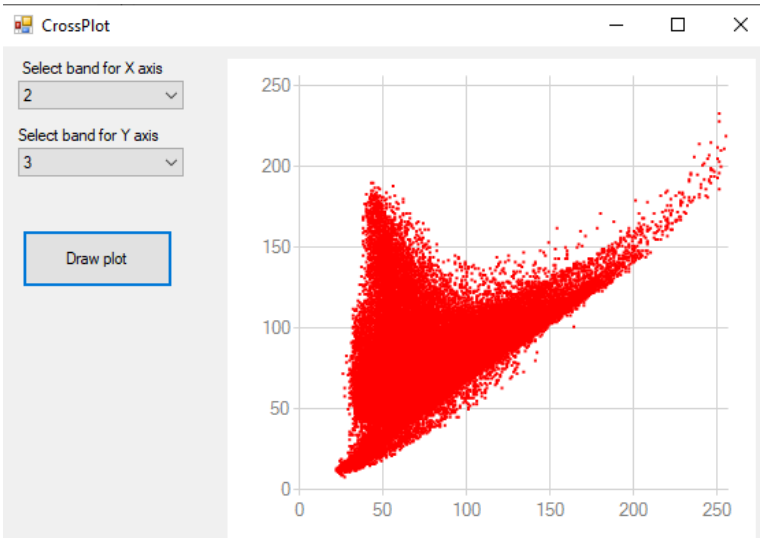
13. ábra. A különböző hullámhosszú fénysugarak visszaverődése különböző állapotú növényekről



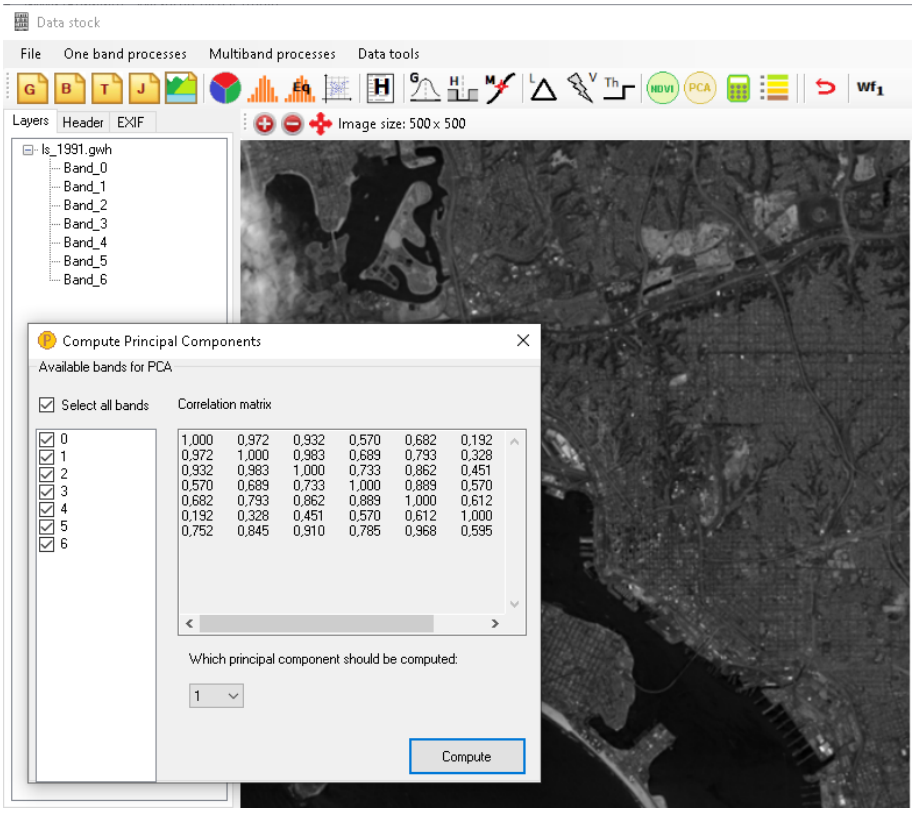
14. ábra. Az egészséges és a beteg növény viselkedése



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 növénnyel fedett részeket. Minél világosabb egy terület annál dúsabb rajta aa vegetáció



16. ábra. Egy példa két különböző frekvenciasáv intenzitásainak összehasonítására



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

4.5.5. compPCA

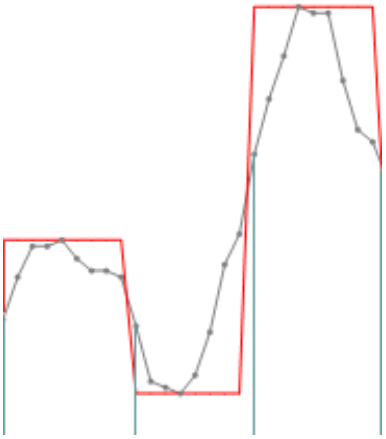
Ez a függvény kiszámítja a megadott byte tömbök által tárolt frekvenciasávok főkomponenseit. A 17. ábrán egy Landsat kép hét frekvenciasávjából számított első főkomponens látható.

4.5.6. Clustering

Az klaszterezési/osztályozási eljárások számos változata létezik. A *Giwer* rendszer saját osztályozó eljárást fog alkalmazni, amely egy 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 detektá-



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.

lása után, azok további szelekciójával fog előállítani szegmenshatárokat. A szegmensek értékei ezeken a határokon belüli pixelek értékei alapján lesznek kiszámítva.

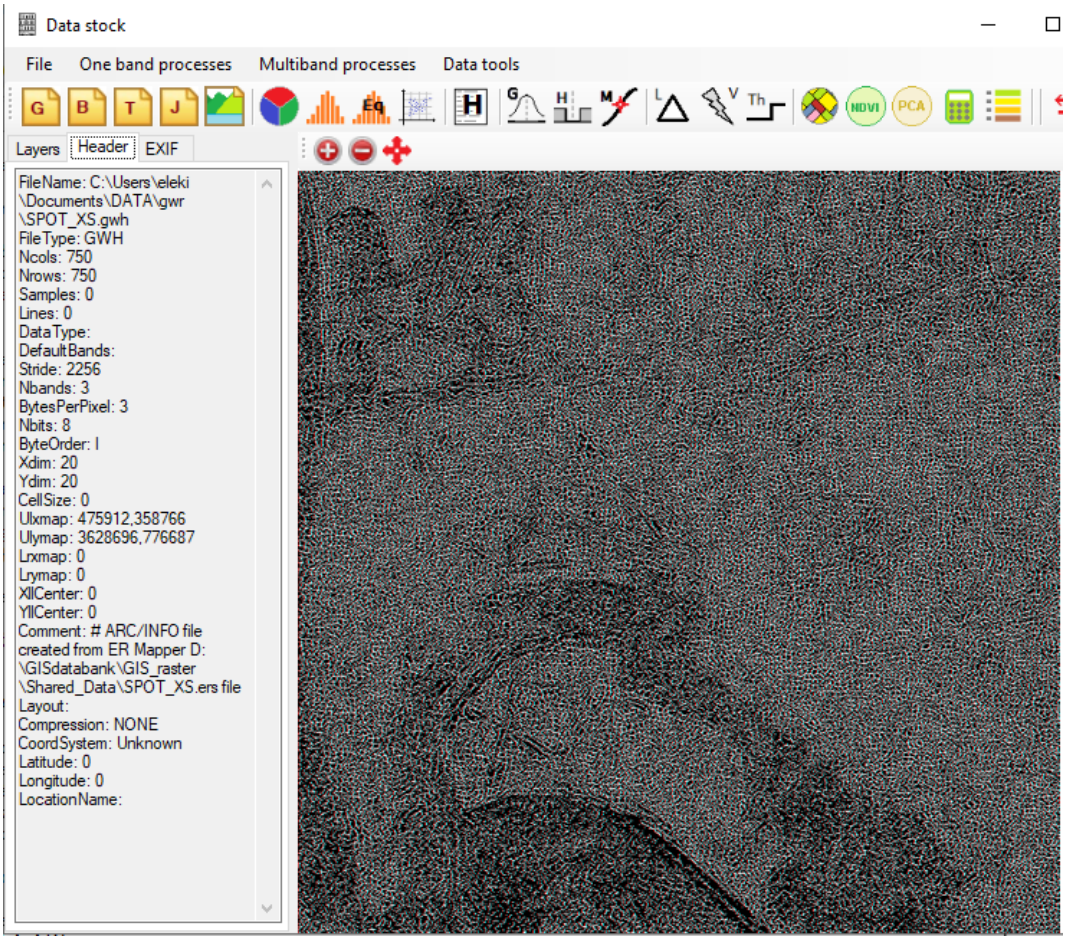
A következőkben ismertetjük az eljárás elvi alapjait. Szegmenshatár ott van, ahol az intenzitás értékében markáns változás van. Tekintsük a legnagyobb intenzitásváltozás helyét a szegmens határának. Erre látunk példát a 18. ábrán. Az a kék vonal azokat helyeket mutatja, ahol az eljárás szegmenshatárt 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 extremuma, ha ezek a szegmenshatárokon belülre esnek. Ha az extremumok a széleken vannak, akkor legyen a jellemző érték a határok közötti értékek átlaga.

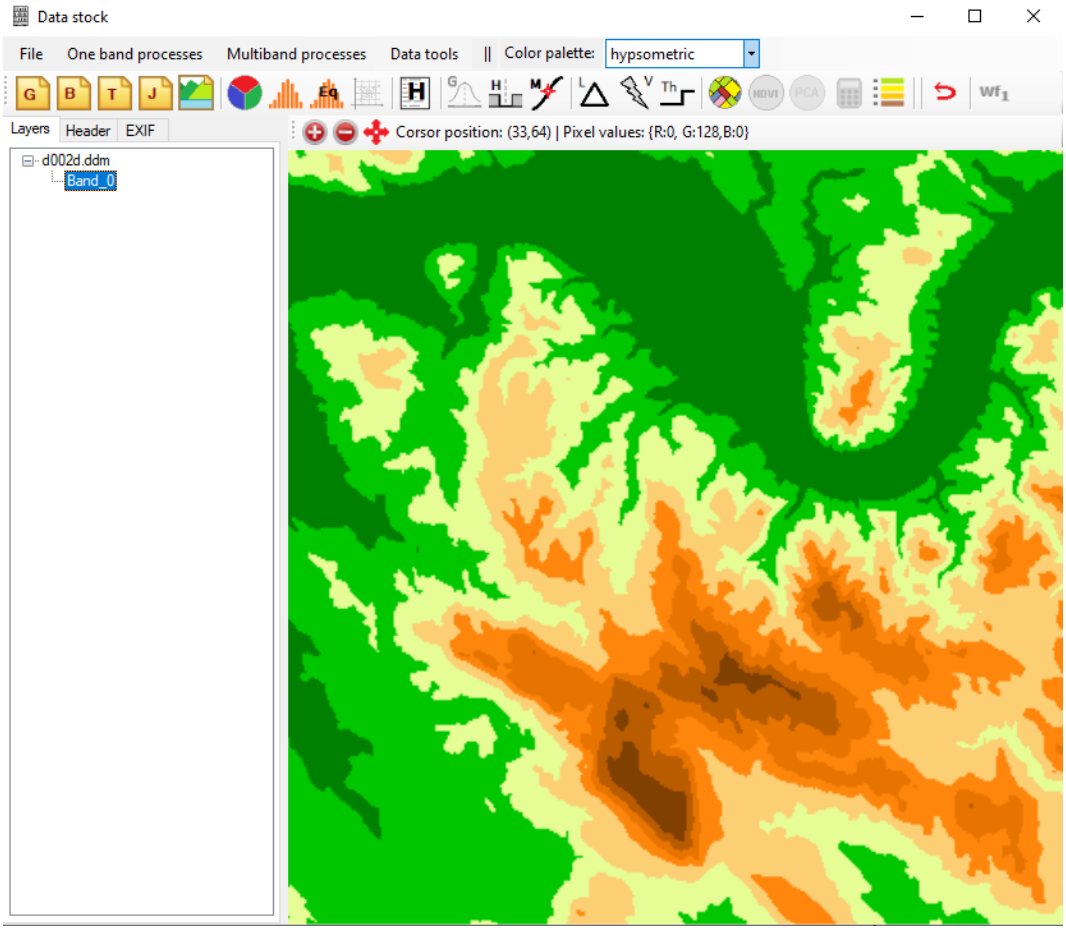
A 19. ábrán egy SPOT kép szegmenshatárait láthatjuk. Ezek a határok a fent ismertetett inflexiós pont keresés alapján lettek megállapítva.

4.6. A DTM osztály

A digitális domborzatmodellre (DTM: digital terrain modell) egy külön osztályt hoztunk létre, mivel az adatok természete jelentősen eltér a képekétől.



19. ábra. Az ábrán egy SPOT kép elemi szegmenshatárait láthatjuk. A kép azért furcsa, mert az eredeti képet EXOR művelettel 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

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 alfanumerikus lekérdező funkciója van, amelynek paramétereit a felhasználó állíthatja be, amint a 21. ábrán látható.

- Where feltételként egy értéket adhatunk meg összehasonlításra
- Where feltételként két értéket (between) adhatunk meg összehasonlításra
- Grafikus lekérdezési lehetőségekkel is felruháztuk a raszter kalkulátort. A cross plot rajzoló ( ) két frekvenciasáv adatait jeleníti meg egy grafikonon, ahol grafikus is szelektálhatunk a pixelek között. Tetszőleges irányú vonalat húzhatunk, és a szerint választhatjuk ki a pixeleket, hogy a cross-ploton ábrázolt sávok intenzitásértékeinek pontjai a vonal felett vagy alatt helyezkednek el (22. ábra). Ennek azért van jelentősége, mert a frekvenciasávok ábrán látható eltérése (többnyire a vörös és infravörös sávokra jellemző) jelez valamit, amire érdemes figyelnünk a képek értelmezése során.

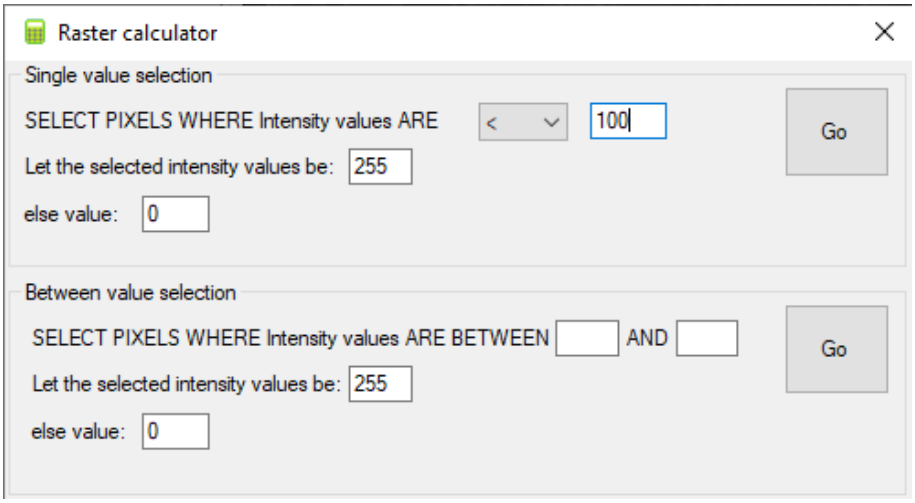
4.8. A Project osztály

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.

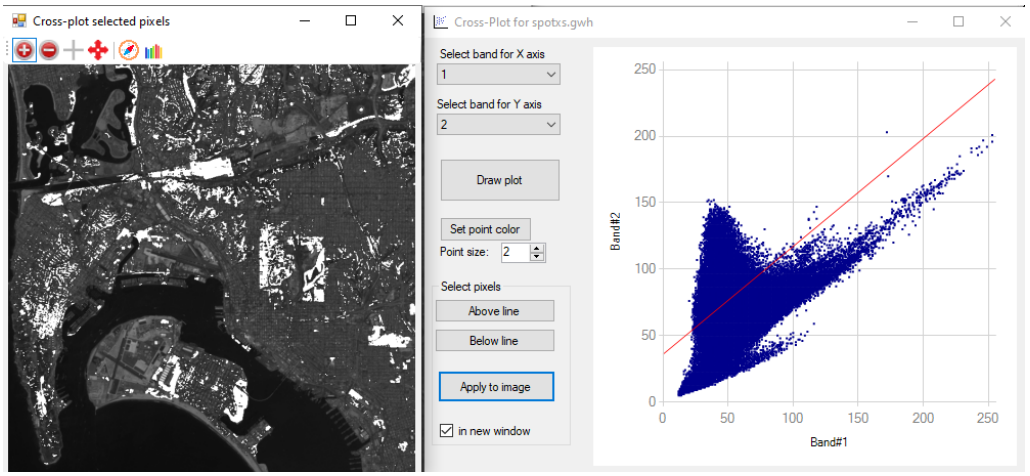
Különösen akkor válik ez igen hasznossá, ha sok drón felvételből kívánunk egy mozaik képet létrehozni.

5. Catalog

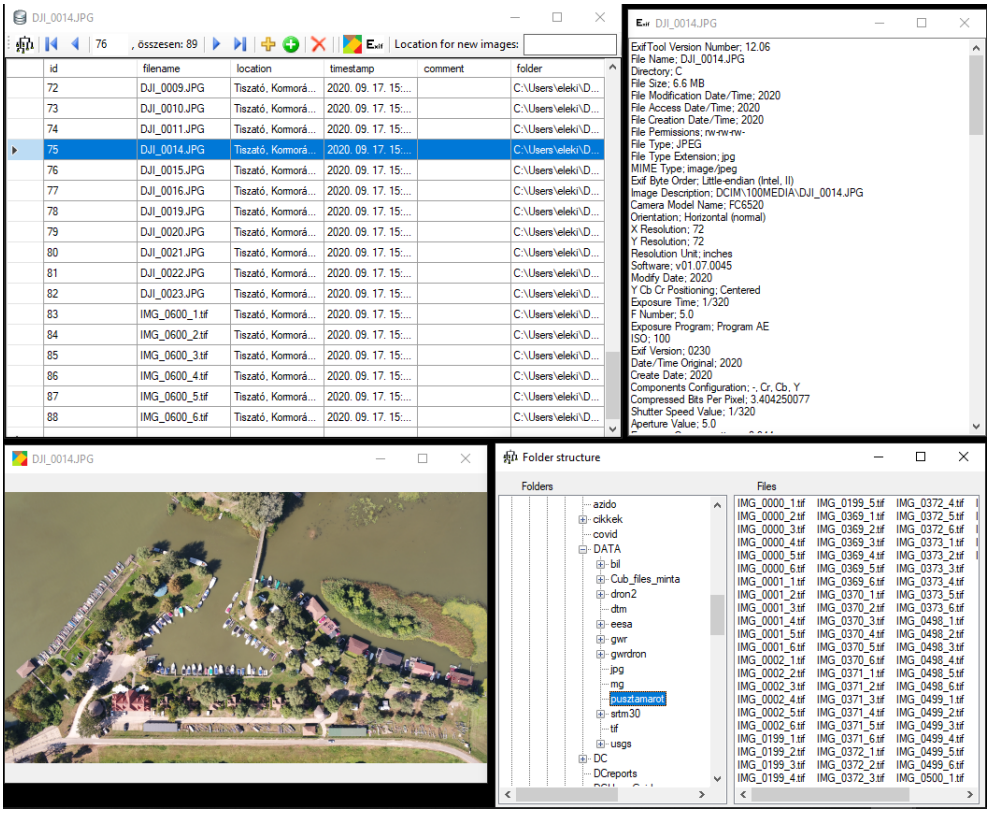
A drónok repülése által olyan nagy mennyiségben keletkeznek képek, hogy ezek kézi áttekintése reménytelen feladatnak mondható. Ráadásul nagy a hibázás lehetősége, ha ezerszámra kell kezelnünk, fejben tartanunk adatokat



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



22. ábra. Szelekció a vörös és a zöld sáv cross-plotja alapján. Az ábra jobb oldalán a piros vonal feletti pontokat választhatjuk ki a kék pontfelhőből. Az eredményt az ábra bal oldalán láthatjuk fehéren



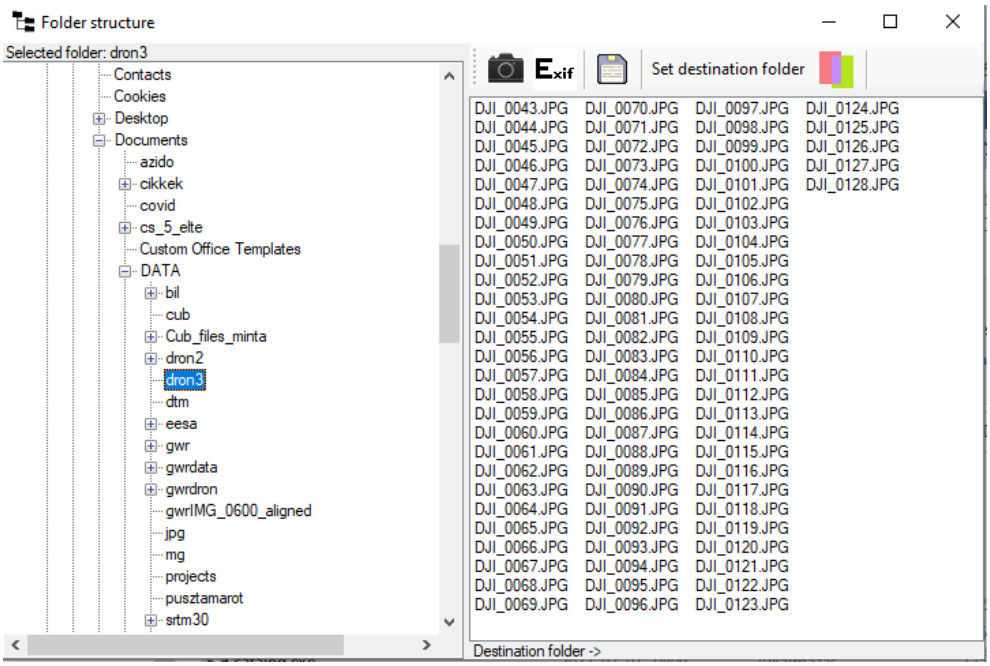
23. ábra. Az ábrán a **Catalog** felületét láthatjuk

a legutóbbi repülés képeiről, azok paramétereiről. Ezért létrehoztunk egy kép katalógust (23. ábra), egy nyilvántartó, kezelő alrendszert, amely adatbázisban tárolja, rendszerezi a képeket. Segítségével még azelőtt tájékozódhatunk a képek minőségéről, elhelyezkedéséről, hogy tényleges elkezdenénk velük dolgozni. Gyakran keletkeznek hibás adatok, amelyeket felesleges alávetnünk az adatfeldolgozásnak.

A drón fotókat a fájlrendszerben gyűjtjük egy előre definiált helyen, ahová a drón adathordozójáról beemelhetjük. (29. ábra).

Az adatbázisban már bent lévő képeket és a hozzájuk tartozó EXIF adatokat is megnézhetjük. Új képeket, egyenként vagy akár tömegesen is hozzáadhatjuk az adatbázishoz. A hozzáadás egyben az adatbázis feltöltését is elvégzi, persze csak azokat az adatokat, amelyek a képekből kinyerhetők (EXIF-ből). Interaktívan is hozzáadhatók adatok, ha azokat a megfelelő mezőbe beírjuk. Egy kijelölt rekordot törölhetünk. Nemcsak a leíró adatok törlődnek, hanem a képek is a fájlrendszerből.

SQL parancsokat is összeállíthatunk, amelyekkel tetszőleges feltétel sze-



24. ábra. A *Folder structure* ablak

25. ábra. Az *Sql editor* ablak

rint kereshetünk (legyűjthetünk) a rendelkezésre álló képek paraméterei alapján. Az 32. ábrán olyan képek legyűjtésének eredménye látható, amelyek a Tiszán készültek, és a kép típusa 'multispektrális'.

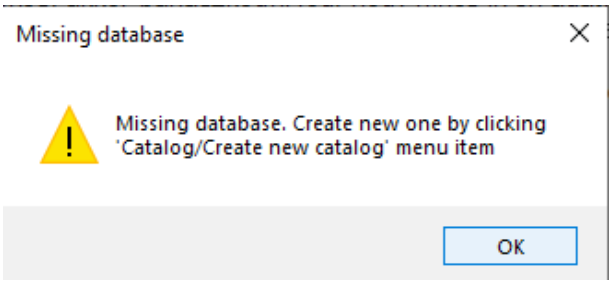
Egy adott mérésre vonatkozó riport fájlt is szerkeszthetünk/megnézhetünk, amelybe olyan adatokat tehetünk bele, amelyeket a mérési körülmények miatt, vagy bármilyen szempontból érdekesnek találunk, de nem az egyes képekhez kötöttek.

A következőknek nézzük meg részletesebben a Catalog működését.

A **Catalog** alrendszer nagy tömegben keletkező képek (csak *jpg* és *tif* formátumú képek) rendszerezésére, adatbázisba szervezésére való (ehhez sqliteot 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 **Catalog**ot olyankor célszerű használni, amikor több száz vagy több ezer képet kívánunk egységesen kezelni, a leíró adataik alapján keresést végezni.

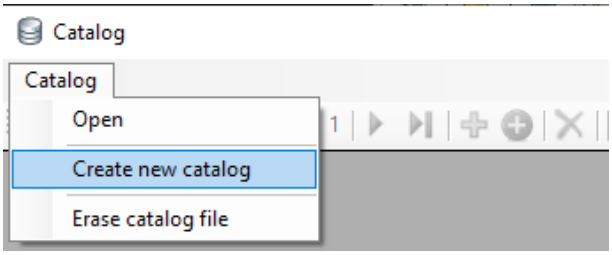
5.1. Első lépések

- 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 (26. ábra):

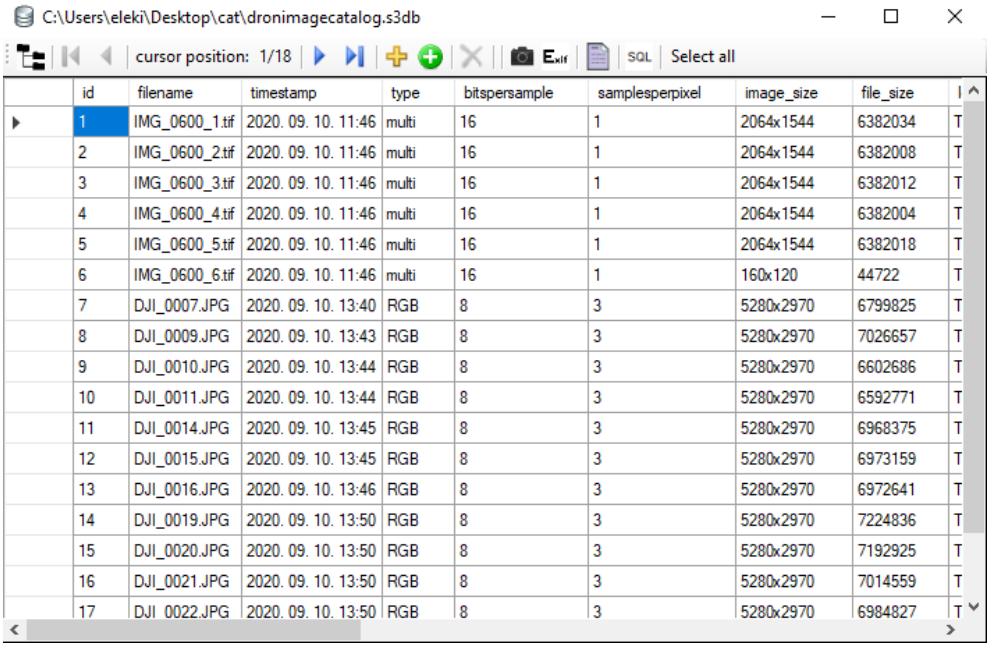


26. ábra. A 'Missing database' üzenet

- Ezután megjelenik a program fő formja, ahol kreálhatunk egy új, üres adatfájl (27. ábra) (a default név 'dronimagecatalog.s3db', de lehet bármi más is)



27. ábra. A *Create new catalog* menü



id	filename	timestamp	type	bitspersample	samplesperpixel	image_size	file_size
1	IMG_0600_1.tif	2020. 09. 10. 11:46	multi	16	1	2064x1544	6382034
2	IMG_0600_2.tif	2020. 09. 10. 11:46	multi	16	1	2064x1544	6382008
3	IMG_0600_3.tif	2020. 09. 10. 11:46	multi	16	1	2064x1544	6382012
4	IMG_0600_4.tif	2020. 09. 10. 11:46	multi	16	1	2064x1544	6382004
5	IMG_0600_5.tif	2020. 09. 10. 11:46	multi	16	1	2064x1544	6382018
6	IMG_0600_6.tif	2020. 09. 10. 11:46	multi	16	1	160x120	44722
7	DJI_0007.JPG	2020. 09. 10. 13:40	RGB	8	3	5280x2970	6799825
8	DJI_0009.JPG	2020. 09. 10. 13:43	RGB	8	3	5280x2970	7026657
9	DJI_0010.JPG	2020. 09. 10. 13:44	RGB	8	3	5280x2970	6602686
10	DJI_0011.JPG	2020. 09. 10. 13:44	RGB	8	3	5280x2970	6592771
11	DJI_0014.JPG	2020. 09. 10. 13:45	RGB	8	3	5280x2970	6968375
12	DJI_0015.JPG	2020. 09. 10. 13:45	RGB	8	3	5280x2970	6973159
13	DJI_0016.JPG	2020. 09. 10. 13:46	RGB	8	3	5280x2970	6972641
14	DJI_0019.JPG	2020. 09. 10. 13:50	RGB	8	3	5280x2970	7224836
15	DJI_0020.JPG	2020. 09. 10. 13:50	RGB	8	3	5280x2970	7192925
16	DJI_0021.JPG	2020. 09. 10. 13:50	RGB	8	3	5280x2970	7014559
17	DJI_0022.JPG	2020. 09. 10. 13:50	RGB	8	3	5280x2970	6984827

28. ábra. A *Catalog* fő formja

- Ezután a 'Open' menüre klikkelve megnyílik az üres adatbázis-fájl.
- Ha már van létező adatbázis (pl. dronimagecatalog néven), akkor megjelenik a tartalma egy táblázatban a program fő formján (28. ábra). Ritka eset, de ha nincs, akkor panaszkodni fog, hogy nincs ilyen adatbázis — mert például 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 'dronimagecatalog' 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.

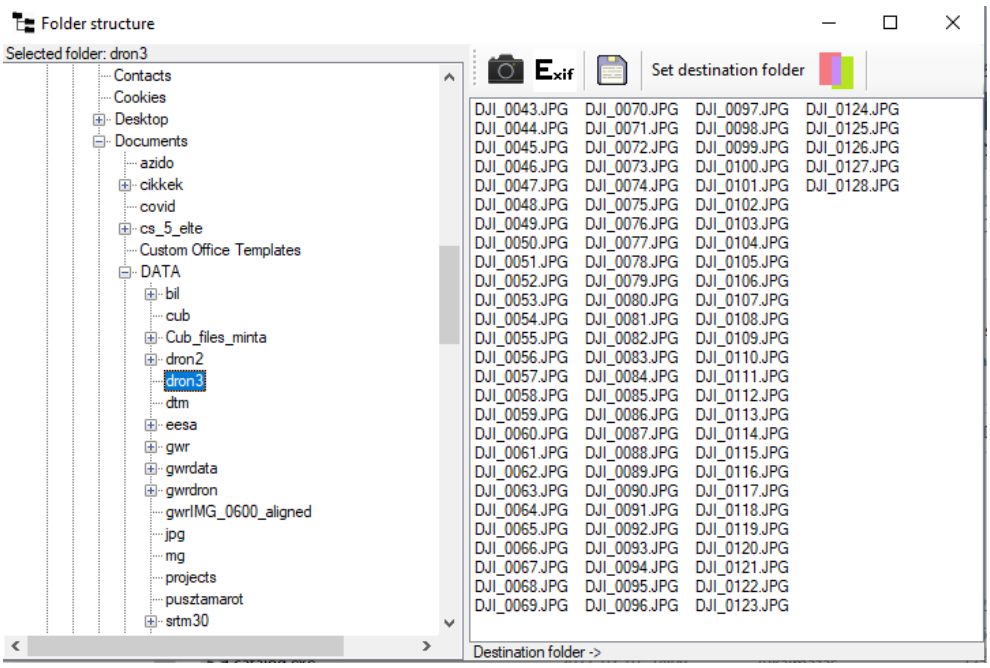
5.2. A fájlrendszer előkészítése

- Kreáljunk egy könyvtárat 'DRON_IMAGES' néven valahol a fájlrendszerben.
- Klikkeljünk az 'open folder tree' ikonra (). Ekkor megnyílik a 'Folder structure' nevű ablak (29. 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 (). Ennek hatására az egész könyvtár tartalma átmásolódik a flash driveról 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.

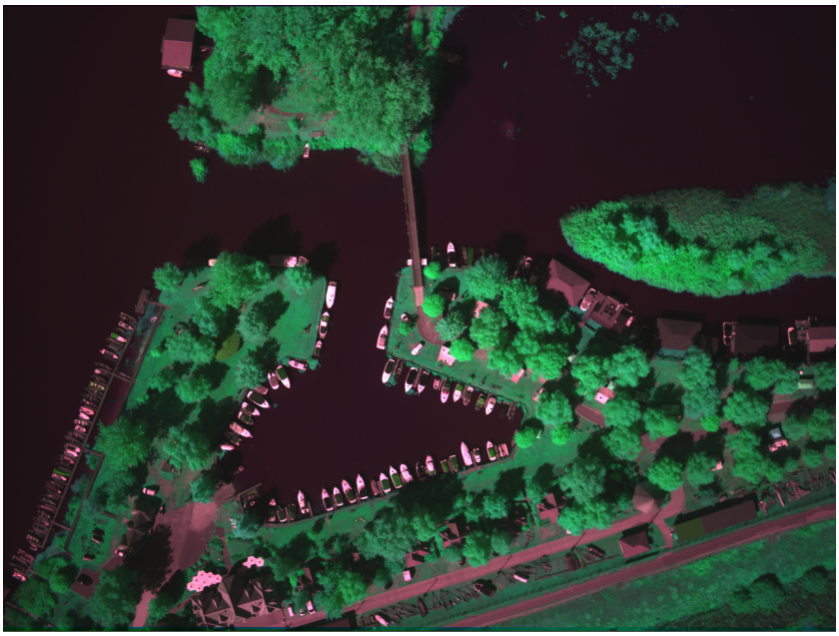
5.3. A Micasense kamera képeinek korrekciója

A Micasense kamera képalkotási hibáit korrigálni kell, mielőtt használatba vennénk. Mivel minden frekvenciasáv képe egy különálló kamerával készült, így ezek képei között kisebb nagyobb elcsúszások vannak. Enélkül a multispektrális képek használhatatlanok.

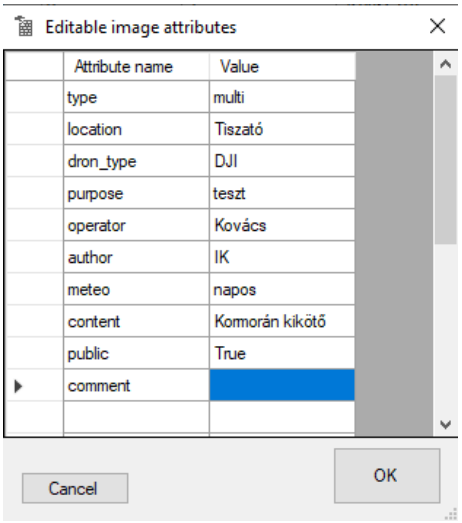
A korrekciót a ikonra kattintva indíthatjuk el. Megnyílik egy fájl dialógus ablak, ahol kiválaszthatjuk a kép fájlokat. Ne felejtsük el, hogy a Micasense annyi kép fájlt készít, ahány sávos a kamera (vagyis 5+1). Az első öt fájl három RGB plusz két infravörös sáv. A hatodik egy termál sáv, amely sokkal rosszabb felbontású, mint a több fájl. Ezt ne vegyük be a kép sávjainak összeigazításába. A 30. ábrán a korrigált kép látható.



29. ábra. A *Folder structure* ablak



30. ábra. A korrigált (aligned) képekből készült RGB, amely a vörös és a két infravörös sávból készült



31. ábra. Az *Editable image attributes* ablak

5.4. 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 kereszt ikonra ( ).

- Bármelyikre klikkeltünk, felbukkan a 'Editable image attributes' nevű ablak (31. 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 tartózó kép. Az 'Exif' ( ) feliratú ikon az aktuális rekordhoz tartozó kép exif adatait mutatja meg egy külön ablakban.

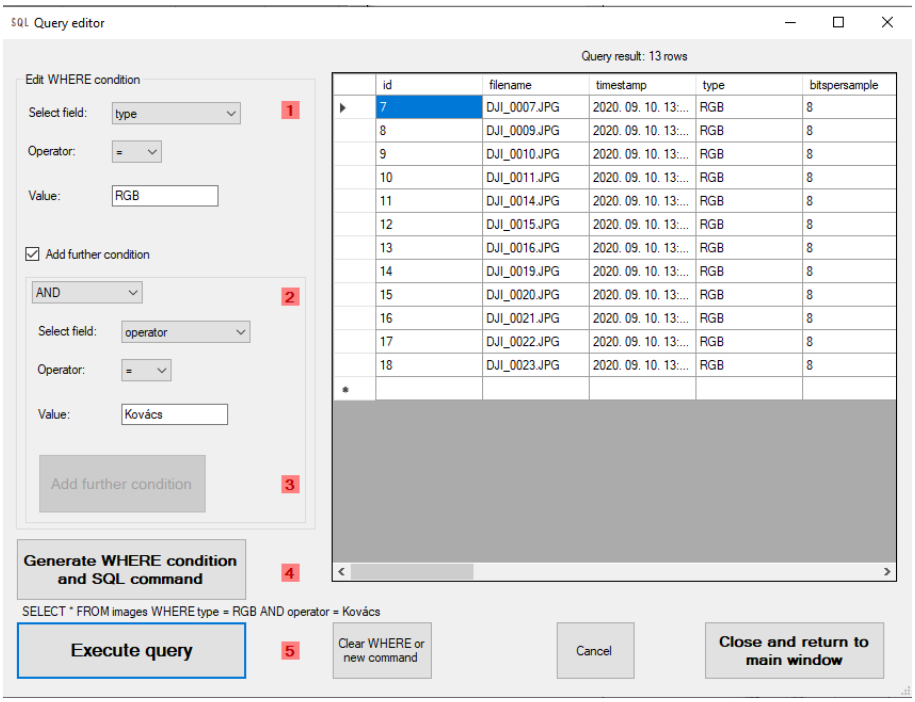
5.5. Funkciók

Az adatokat mutató táblázat felett egy ikonosztáz látható, amelyen a főbb funkciók lettek elhelyezve. A ikon megnyit egy a fájlrendszert nézegető

ablakot, hol megnézhetjük az adatok forrását, mint pl. egy pendrive-ot, ami közvetlenül a drón adattároló eszköze, és amelyen a legfrissebb mérési adatok vannak(29. 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.

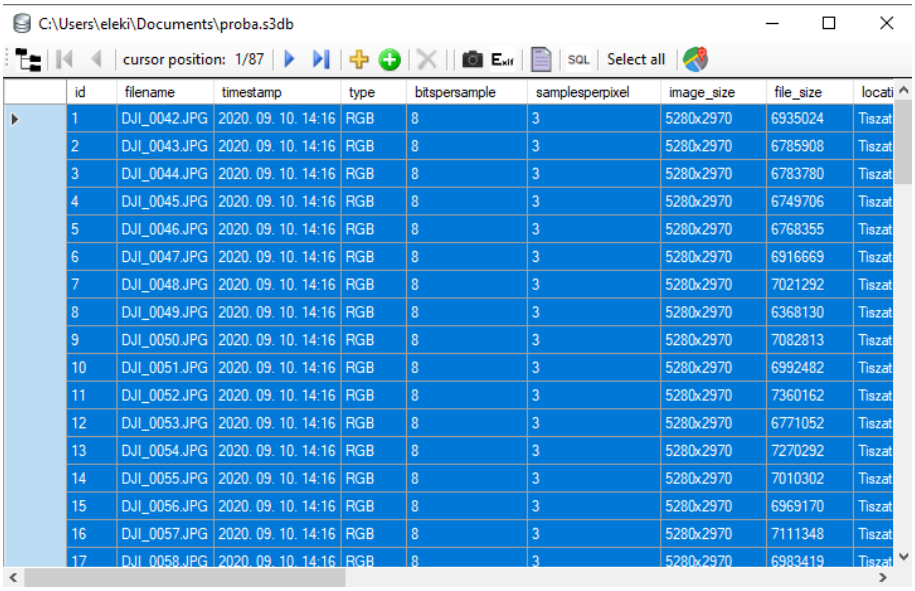
Az adatbázisban már bent lévő képeket a ikonnal, míg a hozzá tartozó EXIF adatokat az Exif ikonnal nézhetjük meg.

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!).



32. ábra. Az *Sql editor* ablak

Az SQL ikonnal SQL parancsokat állíthatunk össze, amelyekkel tetszőleges feltétel szerint kereshetünk (legyűjthetünk) a rendelkezésre álló képek para-



id	filename	timestamp	type	bitspersample	samplesperpixel	image_size	file_size	locati
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

33. ábra. A *Map viewer* ablak

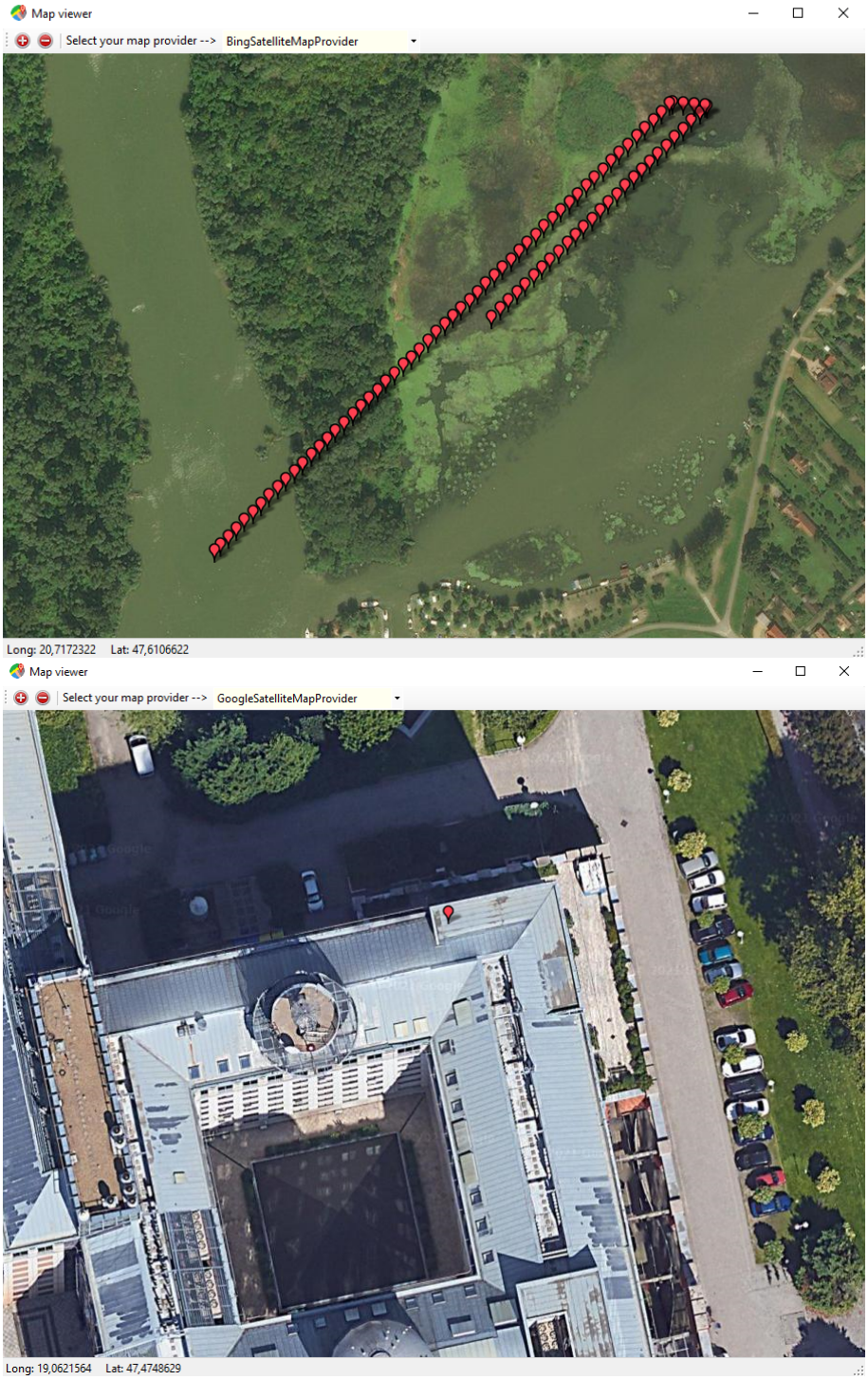
méterei alapján. Az 32. ábrán olyan képek legyűjtésének eredménye látható, amelyek a Tiszán készültek, és a kép típusa 'multispektrális'. Az 'Sql editor' az Sql-t nem, vagy csak alapszinten ismerők számára is használható. (Az Sql-ben járatos felhasználók számára előhívható egy rejtett Sql parancssor, amely azért rejtett, 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 parancssort, amely eltüntethető, ha újra megnyomjuk az F12 gombot. Nemcsak lekérdező, hanem non query típusú parancsok is kiadhatók. A parancs **Enter**rel hajtható végre.)

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

5.6. Lekérdezés

- Az 'SQL' feliratú ikonra klikkelve (SQL) 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.



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

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 klikkeljünk az 'Add further condition' nevű check boxra.

- Ha kész vagyunk egy further feltétellel, klikkeljünk az 'Add further condition'- gombra. Ha az utolsót is hozzáadtuk, akkor klikkeljünk 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, klikkeljünk 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 klikkelve az összes képet legyűjthetjük az adatbázisból, amelyek adatai meg is jelennek az adatrácsban.

- A 'Select all' feliratú gombra a jobb egérgombbal klikkelve az összes képet kijelölhetjük az adatrácsban (33. ábra). Ez olyankor hasznos, amikor térképen akarjuk megjeleníteni a legyűjtött képek centroidjait. Ehhez még rá kell kattintani a  ikonra. Ekkor megjelennek a 'Map viewer' ablakban (34. ábra felső része) a képek centroidjai. Ha úgy klikkeltünk a  ikonra, hogy nem jelöltünk ki egyetlen képet sem, akkor az ELTE Térképtudományi és Geoinformatikai Intézet helye jelenik meg a térképen (a 34. ábra alsó része).

- Sql-ben járatos felhasználóknak közvetlen Sql parancssor is rendelkezésre áll. Ha megnyomjuk az F12 gombot, akkor megjelenik a parancssor. Újból megnyomva eltűnik. Ezt csak biztos tudású SQL szakértő használja, mert nincs UNDO. Igaz, hogy ha elfelejtettük a képek adatbázisba beemelésekor megadni az összes új képre vonatkozó interaktív mező tartalmát, akkor már csak itt tudunk tömeges értékadás végrehajtani (UPDATE), feltéve, hogy meg tudjuk egymástól különböztetni az új képek rekordjait a többitől.

6. Workflow Builder

A **Giwer**-ben igen sok funkciót implementáltunk, melyeket a **DataStock** modulban a menürendszeren keresztül érhetünk el. Interaktívan dolgozhatunk, egy-egy fájlon mindenféle eljárást kombinálva juthatunk el kívánt eredményig.

Amikor azonban több száz kép vár feldolgozásra, akkor ez az egyenkénti feldolgozási mód meglehetősen hosszadalmas és fáradságos munka. Egy-egy terület megismerésével, amit néhány kép interaktív feldolgozásával értünk el, megfogalmazhatunk olyan, a területre általános érvényű feldolgozási módszertant, amit jó lenne az összes képre alkalmazni. Erre a feladatkörre dolgoztuk ki a **Workflow builder**t.

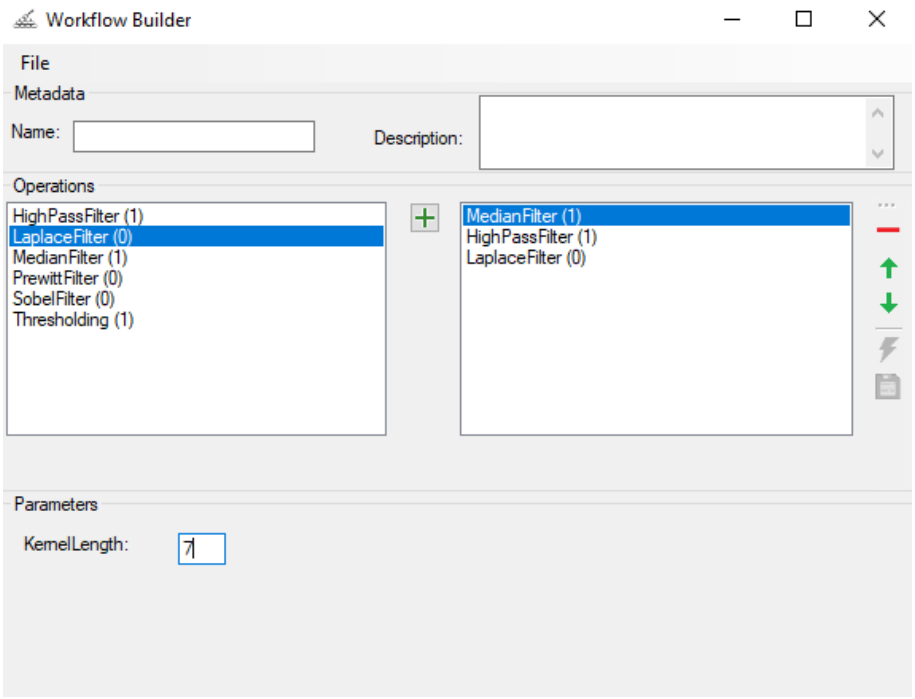
Az implementált függvények közül nem mindegyik való végfelhasználó kezébe, mert valami olyasmit csinál, amit a felhasználó nem használna önállóan (pl. konverzió bitmapből bytetömbbe), de a rendszernek szüksége van rá. Vannak azonban olyan eljárások (szűrők, osztályozók, NDVI, PCA), amire viszont szükség lehet egy feldolgozási folyamat összeállításakor. Ezért összegyűjtöttük ezeket a függvényeket, és a **workflow builder**ben felajánljuk egy listán a végfelhasználónak. Ezek közül választva összeállíthatja a saját munkafolyamatát, amit elmenthet és futtathat. A megoldásra reflektív programozást használtunk.

6.1. Használat

A modult vagy a keretrendszerből indíthatjuk el a **Workflow builder** feliratú gombbal, vagy a DataStock modul *Workflow* menüpontjára klikkelve aktivizálhatjuk. Elindítás után megjelenik a fő ablak (35. ábra).

Az ablak bal oldalán látható az *Operations* feliratú lista, amely az elérhető függvények neveit mutatja. Ebből választhatunk vagy dupla klikkeléssel vagy a megjelölt listaelem után a  jelre klikkeléssel. Ekkor átkerül a kiválasztott eljárás neve a jobb oldali listára. Annyi eljárást tehetünk ide, amennyi csak szükséges.

Mivel az eljárásoknak vannak paraméterei, ezért ezeket úgy adhatjuk meg, ha ráklikkelünk a kiválasztott eljárásra. Ekkor a *Parameteres* ablakban megjelennek a megadandó paraméterek nevei és egy szöveg doboz, ahol megadhatjuk ezeket az értékeket. Az **Enter** leütésével fogadjuk el a begépelt értéket. **Enter** nélkül nem jegyzi meg.



35. ábra. A *Workflow Builder* ablak

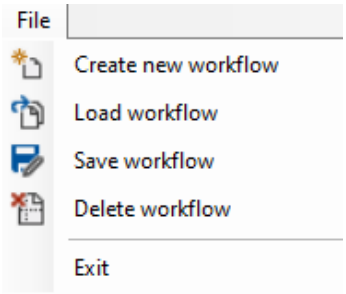
6.1.1. Menürendszer

A menürendszer igen egyszerű. A *File* menü elemeit mutatja a 36. ábra. A *Create new workflow*-val új workflowt hozhatunk létre. Ilyenkor az elérhető eljárások listáját tartalmazó lista kivétel minden törölve lesz. Új eljárás esetén ki kell tölteni a *Name* mezőt, amely a workflow mentéskori nevét hivatott megadni. A workflow fájlok kiterjesztése *wkf*.

Ne hagyjuk üresen a *Description* mezőt, mert később hasznos lehet az ide írt információ. Ha minden paramétert is adatot beírtunk, akkor kattintsunk a *Save workflow* almenüre. Ezzel elmentettük a megadott helyre, ahonnan bármikor visszaolvasható és szerkeszthető.

Ha egy meglévő workflowt szeretnénk használni, kattintsunk a *Load workflow* almenüre. Ekkor megjelennek a wokflow neve, leírása és a kiválasztott függvények nevei. Ha valamely workflow feleslegessé válik, kitörölhetjük a *Delete workflow* menüre kattintással.

A 37 ábra egy workflow fájlt mutat. Ez is egy egyszerű szövegfájl, de a projekt fájlhoz hasonlóan ezt sem célszerű "kézzel" editálgatni, mert könnyű elrontani, és akkor használhatatlanná válhat a workflow. Nem a végfelhasználónak készült, hanem a rendszernek.



36. ábra. A *Workflow Builder* menüi

```
#Description:
Ez itt egy leírás. Nem túl a hasznos ez a folyamat.
#Methods:
GaussFilter (1)
5
MedianFilter (1)
6
Thresholding (1)
7
```

37. ábra. Példa egy egyszerű workflow fájlra. A függvény neve után zárójelben látható, hogy hány paramétert vár az eljárás

6.1.2. Ikonosztáz

A ikont már ismerjük. Ezzel választhatjuk ki a workflowba beemelni kívánt függvényeket az *Operations* nevű listáról. Ha egy függvény feleslegessé vált egy worklowban, akkor a ikonra kattintással lehet levenni a listáról.

A lista elemeit a és nyilakkal tudjuk fel-le mozgatni, ami azért lényeges, mert a workflow a lista sorrendjének megfelelően hajtja végre a műveleteket.

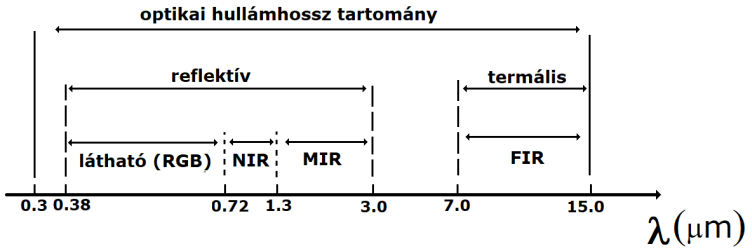
A ikonnal elmenthetjük a workflow fájlt, ami az eljárások nevei mellett a szükséges paraméterek értékeit is tartalmazza. Mentés nélkül nem futtatható a workflow. A futtatást a ikon indítja.

Amíg nem adtunk nevet a workflownak, addig passzív marad a és a ikon. Amíg nem mentettük el a workflowt, addig passzív marad a ikon. Ilyenkor mindkettő szürke színű.

III. rész
Függelék

1. A távérzékelés fizikai alapjai

A távérzékelés megjelenését a globális erőforrások kimerülése okozta, mint például nyersanyagok (olaj), környezeti problémák (környezetszennyezés, fajok kipusztulása) nagyjából a 60 évektől kezdődően. Az űrtechnika és a számítástechnika felgyorsult fejlődése tette lehetővé. Az elektromágneses spektrumnak főként az optikai részét (38. ábra) használjuk (RGB + infra sávok), de előfordulnak hosszabb frekvencia tartományok is, mint pl. a rádióhullámok (RADAR).

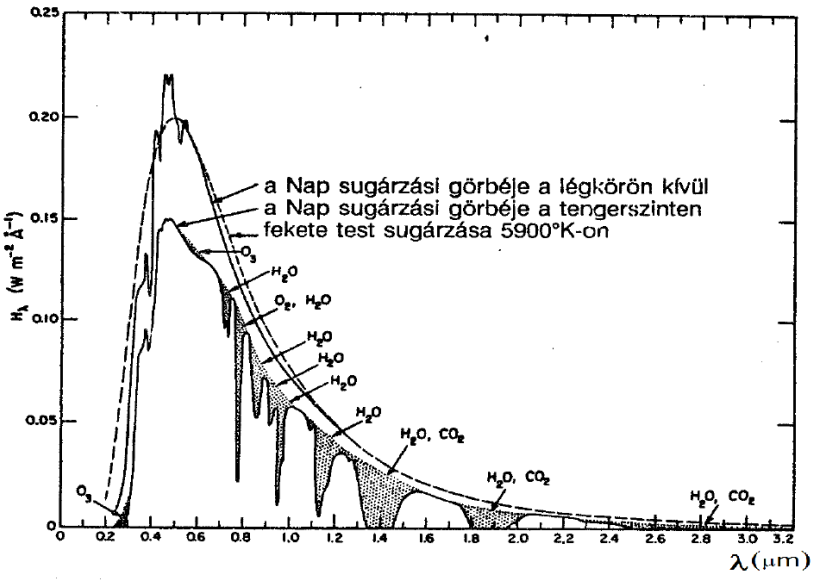


38. ábra. Az elektromágneses spektrum optikai része, ahol az RGB a látható tartományt, NIR a közeli infravörös, a MIR a közepes infravörös, és a FIR a a távoli infravörös tartományt jelöli

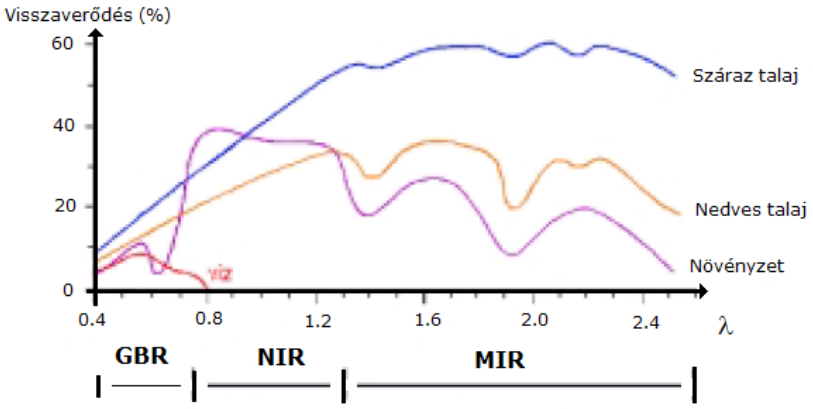
A Napból bejövő elektromágneses sugárzást, amelynek energiaeloszlását a 39. ábrán látható, a légkör jelentősen megváltoztatja (39. ábra), amit figyelembe kell vennünk az eszközök tervezésekor és a képek értelmezésekor. A visszavert elektromágneses sugárzást befolyásolják a felszínt fedő képződmények (víz, növényzet, talajok), ezért azok jelenlétére következtethetünk a visszavert képből (40. 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 (41. ábra).

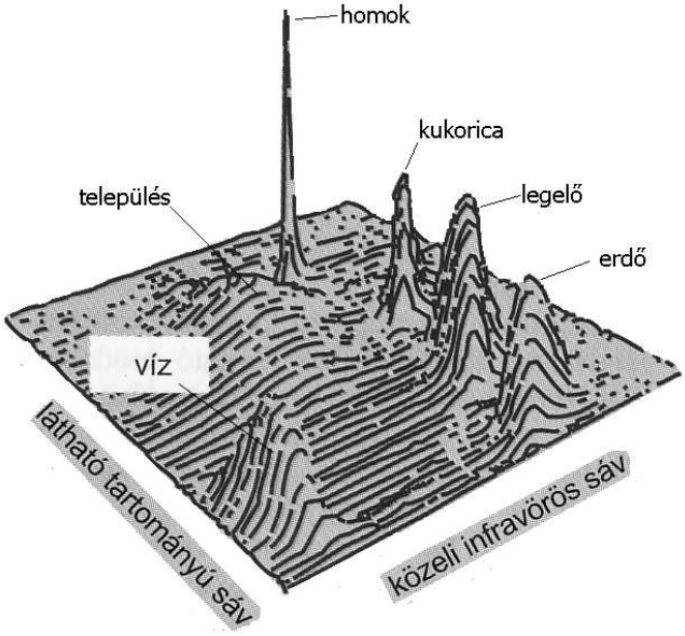
A távérzékelés egyik fő fókuszpontja az agrárium. A cél valamely előre meghatározott anyagminőség detektálása és azok minősítése. Nézzünk néhány lehetséges célkitűzést, vagyis hogy mit akarunk kimutatni. Megállapítunk szakmailag indokolt tematikus kategóriákat, és megpróbáljuk a felszíni



39. ábra. A Nap direkt sugárázásának energiaeloszlása



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



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

képződményeket valamelyik tematikus csoportba besorolni. Nézzünk néhány példát a lehetséges tematikus csoportokra:

1. Aszállyal kapcsolatos kategóriák
    - aszállyal erősen sújtott terület
    - aszállyal 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
    - Bányák, 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 mindig az a szakmai cél dönti el, hogy mit kívánunk kimutatni (pl. gyomos területek felderítése).

2. A főkomponens analízis matematikai alapjai

Sokdimenziós adatrendszerek esetében alkalmazunk dimenzió csökkentő eljárásokat. 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 egységünk, 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:

$$\widetilde{x}_i^j = \frac{x_i^j - \overline{x}^j}{s^j}$$

ahol $\overline{x}^j$ a j-edik vektor elemeinek átlaga (a várható érték becslése), és $\widetilde{s}^j$ 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. Határozzuk meg a korrelációs mátrix sajátértékeit és sajátvektorait, vagyis oldjuk meg a következő sajátérték egyenletet:

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

A sajátértékek $\lambda_1 < \lambda_2 < \cdots < \lambda_p$, a sajátvektorok $\mathbf{v_1}, \mathbf{v_2}, \cdots \mathbf{v_p}$. 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 $j = 1, p$.

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

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

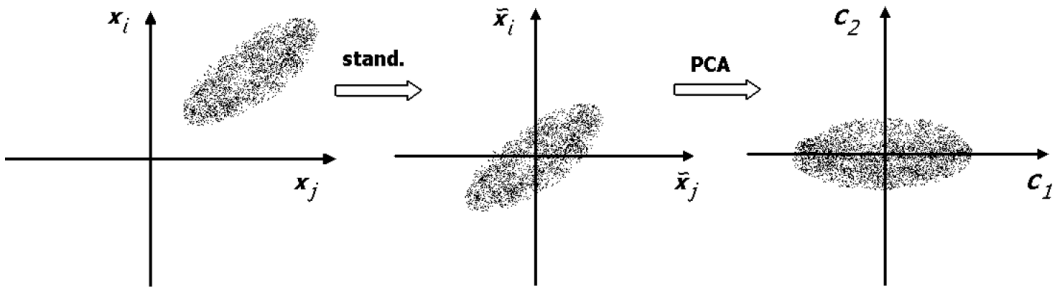
A $\underline{\mathbf{R}}_C$ fontos tulajdonsága, hogy a főkomponensek és a standardizált változók össz-varianciája azonos:

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

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 42. ábra mutatja.

Abban az esetben, ha például az első főkomponens képes magába sűríteni a megfigyelési egységek varianciáinak nagy részét, akkor megtehetjük, hogy az egész adatrendszert csak az első főkomponensével helyettesítjük. Így jelentős mértékben csökkentettük az adatrendszer dimenziószámát, ezzel adatszámát, azaz meggyorsítottuk, 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.



42. 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

Egy másik kézenfekvő kérdés, hogy ha például a megfigyelési egységeink mérési 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 fekvenciatartományt a Fourier-transzformáció köti össze:

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

illetve

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

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 megszorozzuk egy olyan négyszögfüggvénnyel, 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 függvénnyel: $S'(f) = S(f)A(f)$, 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:

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

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:

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

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 fogalommá vált. Nemcsak olyan esetekben használják, amikor a művelet hatása megadható a spektrum valamely függvénnyel történő szorzataként, 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 értékekből 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 összefüggenek, mint az egymástól távoli pixelek. Ezeknek a szűrőknek egy másik fontos jellemzője, hogy nem rekurzívak. Ez azt jelenti, hogy az eljárás csak az eredeti intenzitásértékektől függ, azaz mindig az eredeti képből vesszük a képpont szomszédságát.

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 konvolúvió értéke a t-edik helyen:

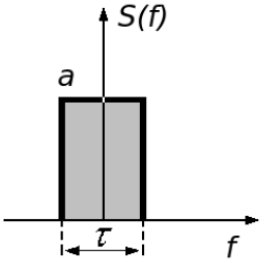
$$h(t) = \sum_{\tau=-\infty}^{\infty} f_1(\tau) f_2(t - \tau) \tag{6}$$

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

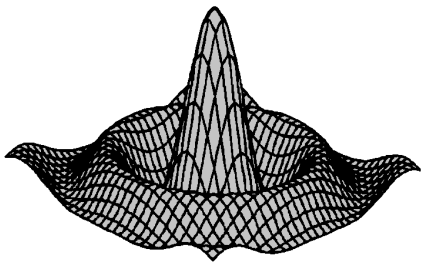
$$h(x, y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f_1(u, v) f_2(x - u, y - v) \tag{7}$$

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 43. ábra, illetve ennek



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



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

inverz Fourier-transzformáltját a 44. 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 44. 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

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

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 négyszögfüggvénnyel 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:

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

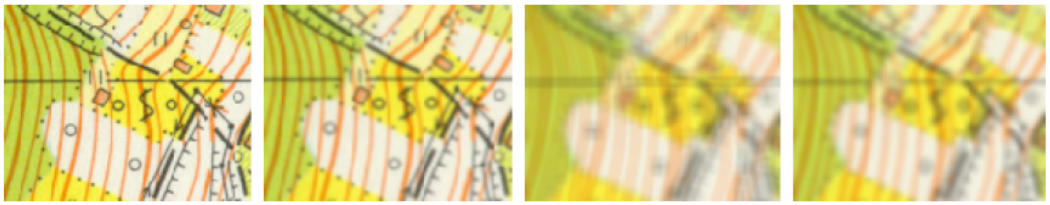
Így a 0 várható értékű, σ szórású Gauss-görbét kapunk, amelyből a kernelt úgy számoljuk, hogy a rácspontokon mintavételezzük a G függvényt. Mivel

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

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



45. ábra. Balról jobbra: Eredeti kép. A kép 3×3-as, Gauss-szűrt változata, σ=1.0. A kép 7×7-es, box-szűrt változata. A kép 7×7-es Gauss-szűrt változata, σ =2.0

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

alakúak lesznek.

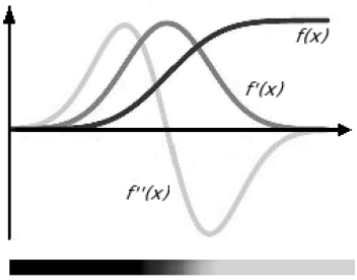
A Gauss-szűrő, mint látható, simító jellegű. A simítás mértéke a szórás nagyságától függ, természetesen nagy szórás esetén a kernel méretét is növelni kell (45. ábra). A Gauss-szűrőnek szintén nem önmagában, hanem más algoritmusokban van szerepe, pl. a Canny-féle éldetektorban használjuk. Átviteli függvénye, 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 határfrekvencia 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 feletti képeket egymásból kivonva a heterogén részek felerősödnek, ez a tulajdonság jól használható a textúrák detektálásánál.

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.



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

3.2. Éldetektorok

Az éldetektorok olyan lokális szűrők, melyek a kép egy pontjában a szomszédos elemek segítségével leírt intenzitásfüggvény deriváltjával dolgoznak (46. 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 hányadossal.

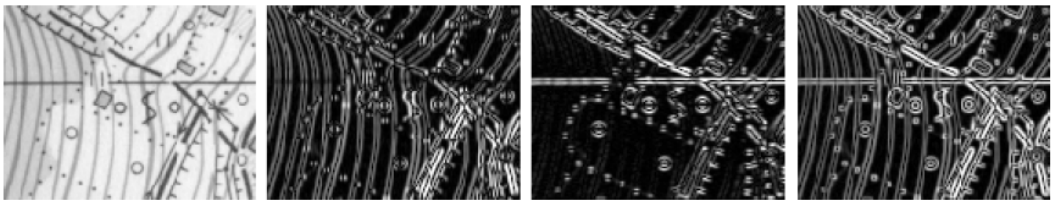
$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

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

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

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

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 (47. ábra). Az eredményt p-vel normalizálni kell.



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

3.2.2. Laplace-szűrő

A Laplace operátor definíciója:

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

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:

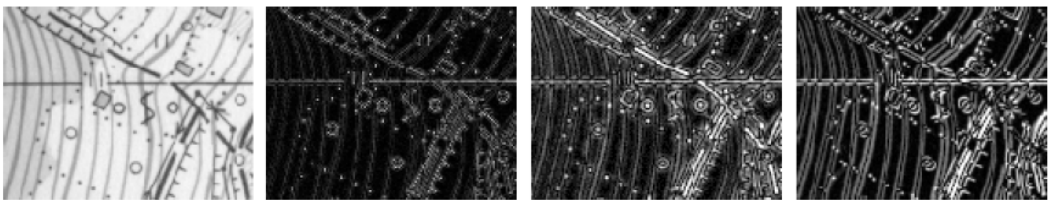
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

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 (48. ábra).

3.2.3. LoG szűrő

A Laplace szűrő előtt egy Gauss-simítást alkalmazva jó éldetektort kaphatunk (48. ábra). Mivel a konvolúció asszociatív, ezért megtehetjük, hogy a Laplace- és Gauss-szűrő kerneljét konvolváljuk, és az eredményként kapott mátrixot használjuk. Ezt a szűrőt szokták „Laplacian of Gaussian" (LoG) nevezni. A LoG-szűrő kevésbé érzékeny a zajra, jó éldetektor. A kernelmátrix a következőképp számolható:

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



48. 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

3.2.4. Emboss szűrő

Az emboss szűrők célja speciális irányú élek detektálása (48. 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-függvénnyel (g) való konvolúciójának a deriváltja 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 mindazokat a helyeket, ahol a h gradiensének értéke meghalad egy 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:

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

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 $M(x, y)$ értékeket. 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 tehetjük meg, hogy az $M(x, y)$-t lemásoljuk egy kimeneti képbe, a $\Theta(x, y)$-ban adott normálvektor által meghatározott szögben lépünk mindkét 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 hagyjunk el, egy határ fölött mindent tartsunk 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 simítsuk 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

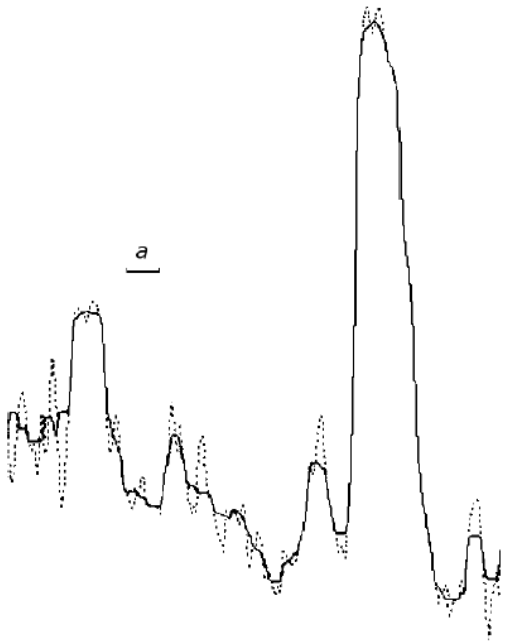
A nemlineáris szűrők azok az eljárások, amelyek ugyancsak egy adott pixel szomszédos pixelei alapján számolják ki a szűrt értéket, de nem a szomszédos pixelek értékeinek valamilyen lineáris kombinációjaként, hanem más módon. Itt nem beszélünk kernelmátrixról, hanem csak kernelről, vagy kernelablakról.

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 (50. 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 49. ábra a medián szűrőnek egy zajos görbére 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



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

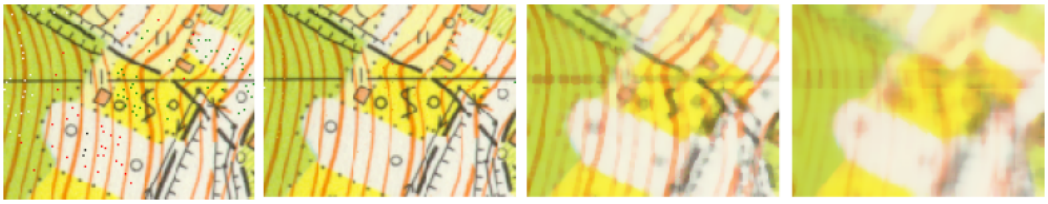
eredmény, amikor a nagy területeket próbáljuk kiemelni. Sajnos az éleket eltolhatja és a sarkokat lekerekíti, de az algoritmus kiegészíthető úgy, hogy ez a hiba ne forduljon elő.

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 elemekből 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 (50. ábra).



50. 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

- 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.

3.4. Szegmentálás, küszöbölés

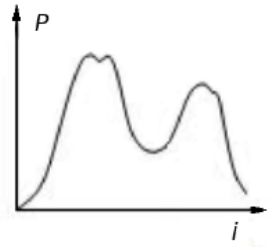
A küszöbölés a szürkeárnyalatos képek szegmentálásának egyik módja. Ilyenkor megadunk néhány küszöbértéket, amelyek intervallumhatárokat fognak jelölni. A küszöbölés speciális esete a kétszintes küszöbölés, a binarizáció, amikor egyetlen küszöbértéket adunk meg, így a pixeleket két osztályba soroljuk.

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 hátterüktől. Mivel két osztályba sorolhatunk be minden pixelt, ezért ez nem sikerülhet mindig tökéletesen, de a jó küszöbölés ezt a lehető legjobban közelíti.

3.4.1. Otsu-féle küszöbölés

Tekintsük meg az 51. 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.



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

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

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

$$\mu_1(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} iP(i)$$

$$\sigma_1^2(t) = \sum_{i=0}^{t} (i-\mu_1)^2 P(i)$$

valamint

$$\mu_2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{255} iP(i)$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{255} (i-\mu_2)^2 P(i)$$

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:

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

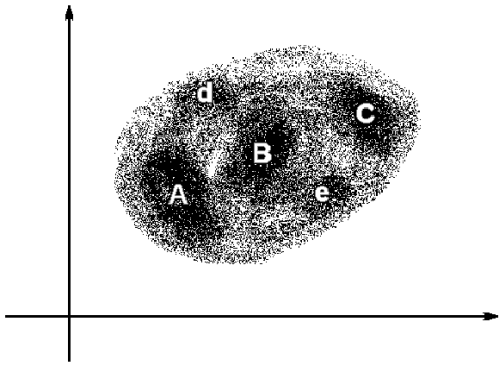
Az optimális szórás számunkra az, ami a lehető legjobban elkülöníti az osztályokat. Mivel a kétféle szórás összege egyenlő a teljes szórással, ezért két, egymással ekvivalens célunk lehet az optimum megtalálásához: minimalizáljuk az osztályokon belüli szórást, vagy maximalizáljuk az osztályok köztit. 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 51. ábra. Az Otsu-féle küszöbölés általánosítható, azaz több szintű küszöbölést is lehet ezzel a stratégiával előállítani.



52. ábra. Csoportok egy kétdimenziós képen

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 raszteres adatmodellt követő esetekben, mint amilyen az űrfotók 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 (52. ábra).

Egy adathalmaz pontjainak az adatrekordok hasonlósága alapján történő diszjunkt csoportokba sorolását klaszterezésnek nevezzük. A csoportosítás jósága alapvetően két dolgon múlik: a jó hasonlóság definíción és egy jó algoritmuson, amely a hasonlóságon alapulva valamilyen kritériumok alapján megállapítja a klasztereket. 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 egymáshoz közel vannak.

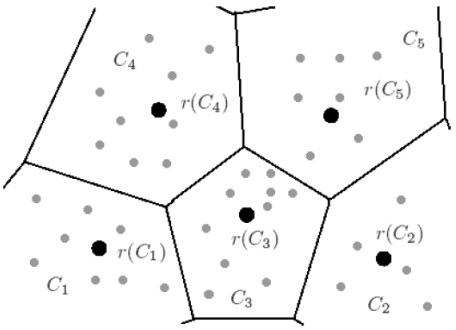
Ez a megfogalmazás elég tág határokat enged meg a klaszterek meghatározására, de valamennyi klaszterező eljárás hátterében megtalálható a távolság mátrix, illetve a klaszterek középpontjától való távolság.

4.1. Particionáló klaszterezés

Feltesszük, hogy a klaszterek egy vektortérben helyezkednek el. A klasztereket súlypontjukkal reprezentáljuk, vagyis a klaszterekhez tartozó adatpontvektorok átlagával jellemezzük (53. 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.



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

A besorolás, súlypontszámítás lépéseit 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 minden belső pontja egy klaszternek felel meg, és azokat a pontokat tartalmazza, amelyek a fában alatta találhatók.

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 adatpont 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:

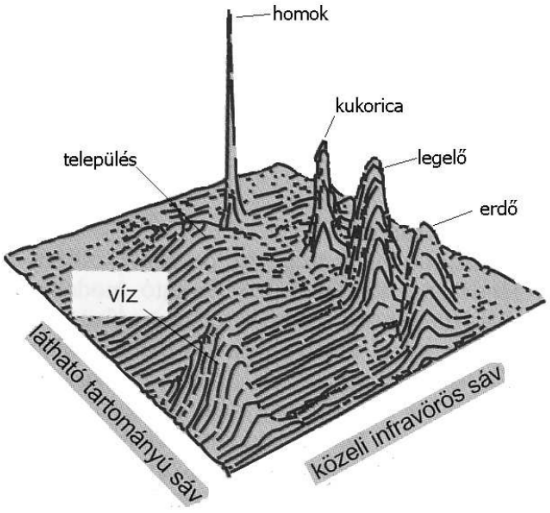
1. Válasszunk ki klaszter középpontokat kiindulásul
2. A pixeleket a hozzájuk legközelebb eső középpontú klaszterbe soroljuk
3. Az újfent besorolt pontok figyelembe vételével kiszámítjuk az új klaszter középpontokat, amik ettől kisebb nagyobb mértékben megváltoznak
4. Az eljárás leállását a középpontok 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 mintaterületeinek kijelölése történhet terepi 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 54. ábrán. (Feltételezhetően annak tudható be a normális eloszlás, hogy a visszaverődés jelensége sokféle folyamat együtteséből tevődik össze. Márpedig ezek akármilyen eloszlást is kövessenek, az összegük eloszlása közelíteni fog a standard normális eloszláshoz. Ez a centrális határeloszlás tétele.)



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

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 lehetségessé. Ez a körülmény teremti meg az értelmét a több száz frekvenciasávban működő hiperspektrális távérzékelés számára.

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

[5] Elek István: Adatbázisok, térképek, információs rendszerek, ELTE Eötvös kiadó, 2010, ISBN 978 963 312 039 2

[6] László István – Csornai Gábor – Fekete István – Giachetta Roberto: Távérzékelt felvételek elemzése, Egyetemi jegyzet, ELTE IK, 2014