Eötvös Loránd Tudományegyetem
Informatikai Kar

MSc. diplomamunka

Scannelt raszteres térképek vektorizációja

Készítette:
Nemes Krisztián
Térképész MSc.

Témavezető:
Dr. Elek István
habilitált egyetemi docens
ELTE IK Térképtudományi és Geoinformatikai Tanszék

2012. június 5.

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

DIPLOMAMUNKA-TÉMA BEJELENTŐ

Név: **Nemes Krisztián** EHA-kód: NEKPAAT.ELTE

Tagozat: **nappali** Szak: **térképész MSc**

Témavezető neve: Elek István

munkahelyének neve és címe: ELTE-IK Térképtudományi és Geoinformatikai Tanszék 1117 Budapest, Pázmány Péter sétány 1/A

beosztása és iskolai végzettsége: habilitált egyetemi docens

A dolgozat címe: **Scannelt raszteres térképek vektorizációja**

A dolgozat témája:

Napjainkban a térképek rajzolása már számítógéppel, vektorosan történik. Ehhez sokszor alaptérkép szükséges, mely beillesztése legtöbbször scannelés útján történik. Ugyan ez a (raszteres) térkép már használható, mégis, vektorizált állapotban sokkal alkalmasabb, és időt takaríthatunk meg vele, ha egyes részei már automatikusan vektoros állománnyá alakulnak. Ám a teljesen automatizált vektorizáció még nem 100%-os, szükség van emberi beavatkozásra egy-egy következő lépés eldöntéséhez. Dolgozatomban összesítem a vektorizáció területén szerzett új ismereteket, emellett egy saját programot készítek, ami képes lesz alapvető műveletek elvégzésére, megkönnyítve a digitalis térképkészítést.

A témavezetést vállalom:

.....
(a témavezető aláírása)

Kérem a diplomamunka témájának jóváhagyását.

Budapest, 2011. december 1.

.....
(a hallgató aláírása)

A diplomamunka-témát az Informatikai Kar jóváhagyta.

Budapest, 2011. december 1.

.....
(a témát engedélyező tanszék vezetője)

Előszó

Napjainkban a térképek rajzolása már számítógéppel, vektoros formában történik. Ehhez sokszor papír alapú alaptérkép szükséges, mely beillesztése legtöbbször scannelés útján oldható meg. Ugyan ez a (raszteres) térkép már használható, mégis, vektorizált állapotban sokkal alkalmasabb, hiszen az objektumokhoz *attribútumok*[1] köthetőek és nem utolsó sorban időt takaríthatunk meg vele, ha egyes részei már automatikusan vektoros állománnyá alakulnak. Ám a teljesen automatizált vektorizáció még nem 100 százalékos, szükség van emberi beavatkozásra egy-egy következő lépés eldöntéséhez. Diplomamunkám ezzel a folyamattal, a digitalizálással, és az azt követő vektorizációval foglalkozik. A vektorizáció célja a térkép szerkesztésének megkönnyítése, felgyorsítása. A téma ötletét Szarvas András térképésztől kaptam, akivel volt szerencsém találkozni egy kurzus keretein belül. Mesélt a térképkiadás jelenlegi helyzetéről és a lehetőségekről. Ekkor esett szó a vektorizációról is, ami igencsak felkeltette érdeklődésemet. Szerencsére nem kellett messzire mennem támogatásért, hiszen a tanszéken Elek István már foglalkozott a témával és rögtön fel is ajánlotta a segítségét.

Dolgozatomban összesítem a vektorizáció területén szerzett új ismereteket, emellett egy saját programot készítek, ami képes lesz alapvető műveletek elvégzésére, megkönnyítve a digitalis térképkészítést.

[1] A dőlten szedett szavak/fogalmak magyarázata a Fogalomtárban található.

Tartalomjegyzék

Ábrák jegyzéke

1. fejezet

Bevezetés

Egy új térkép többnyire két fontos összetevőből áll [2, 37–40. dia], melyek kiválasztása nagyban függ a készülő térkép típusától, céljától. Ez a két összetevő a következő:

- topográfiai adatok (általában topográfiai térkép mint alaptérkép) a megfelelő háttértartalommal,
- és a célhoz köthető tematikus adatok, céltartalommal. Ezek lehetnek már meglevő tematikus térképek, írásos források stb.

Topográfiai anyagok általában a jól megválasztott *méretarányú* topográfiai térképekből nyerhetők, azok *generalizálásából*, ha szükséges. Az térképes alapanyagok általában két módon állhatnal rendelkezésünkre: raszteres vagy vektoros állományban.

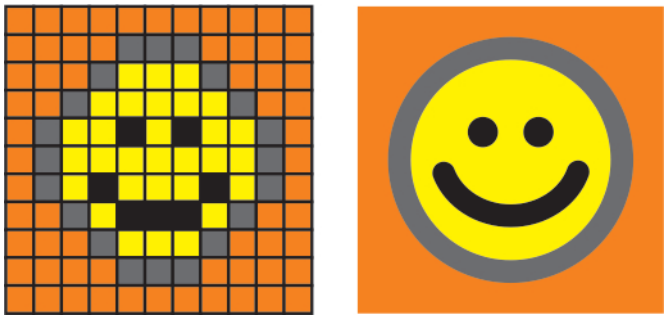
1.1. Raszteres (tér)kép [4, 33–34. o.]

A raszteres képek alapja a raszteradatmodell. E modell egyszerű elveken alapul, maga a kép egy rácshálóval van lefedve, mely rácsháló elemeihez (*pixeljeihez*) hozzárendelünk egy rá jellemző adatot (például színt). Raszter alapú képeknél nem a pixel mérete, hanem a *felbontás* a leginkább használatos jellemző.

Egy raszteres (tér)kép állományának a következő alapvető adatokat kell tartalmaznia:

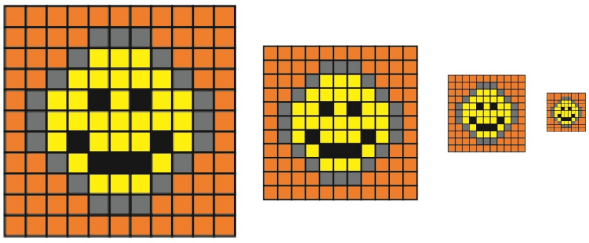
- A raszterháló geometriai jellemzői: sorok és oszlopok száma, pixelméret, térkép esetén esetleg a térképi koordináta-rendszer transzformációs paraméterei. Amennyiben nem téglalap alakú területről van szó, akkor rendelkeznünk kell a határoló vonal koordinátáival.
- Annyi attribútum, ahány pixelből áll a térkép. Tehát minden pixelhez kell tartozzon egy attribútum. Természetesen maguk a raszteres állományok ettől eltérő struktúrában is tárolhatják a pixelekre utaló információkat.

A raszteres adatmodell hátránya az így létrejött igen nagy tárolandó adattömeg, hiszen minden egyes raszterpont attribútuma tárolódik. Továbbá a térképi elemek a megszokott pont/vonal/terület formában nem hozzáférhetők, egyedileg nem azonosíthatók. Tipikusan csak raszter formában tárolható grafikus adatok például a (légi)fényképek, űrfotók. A raszterformátumú állományok két legfontosabb jellemzője a már említett felbontás, az ábrázolt terület nagysága és az egy pixelhez tartozó attribútum tárolási módja: egy bináris változót (0/1, igen/nem) ábrázoló kép egy pixele egy biten tárolható, míg egy *valós színeket* tartalmazó színes kép esetében egy pixelhez 16,7 millió (2^{24}) féle attribútum tartozhat, amit 3 bájton lehet tárolni (alapszínenként 1 bájton). Igazán professzionális alkalmazások esetén a színek száma 2^{32} is lehet, ilyenkor a három színkomponens mellett még egy átlátszósági tényezőt is figyelembe veszünk.



1.1. ábra. Egy raszteres és egy vektoros kép (a megfelelő szemléltetéshez erősen túlozva)

Bár a vektoradatmodell napjainkban gyakrabban alkalmazott a térképészetben, raszteres adatok kezelésére is sokszor szükség van. A fentebb már említett űrfelvételek és légi fényképek a legjellemzőbb raszteres adatok, amikkel a kartográfusok találkoznak. A raszteralapú térképek főbb hátránya, hogy a képernyőn sokkal kevésbé nagyíthatók és kicsinyíthetők (1.2. ábra), mint a vektoralapúak. Ennek oka, hogy a raszteralapú térképek erős nagyításban pixelekre bomlanak, a kép szétesik, míg kicsinyítés esetén a képpontok "összevonódnak", és így az apróbb részletek elveszhetnek.



1.2. ábra. Egy raszteres kép nagyítása és kicsinyítése

Ugyanakkor a raszterformátum legnagyobb előnye az egyszerű bevitel: scanner, esetleg digitális kamera segítségével gyorsan digitális adatsorrá alakítható egy papírtérkép, viszont ennek további manipulálása, megváltoztatása rendkívül nehéz feladat, hiszen a térképi elemekhez nem tudunk a pont/vonal/felület struktúrában hozzáférni (mint

ahogy a vektoros adatmodellnél majd látjuk). Scannelés útján csak színinformációhoz jutunk a pixelek attribútumaként. Például egy út, egy folyó attribútumok alapján való kiválasztása igen nehéz feladat egy scannelt raszteres térképen. A raszteres állományok sajátossága, hogy egy adott pontban (pixelben) csak egyféle színinformáció tárolható, azaz például egy vízfolyás és egy út keresztezésében az a szín fog csak megjelenni, amely a *rajzi struktúrában* feljebb volt. Egy-két speciális raszteres formátum ennél bonyolultabb tárolást is lehetővé tesz (raszteres rétegek, átlátszóság).

Fontosabb raszteres formátumok [4, 65–70. o.]

- TIFF

  A TIFF raszteres állományformátumot az Aldus, a Microsoft és a professzionális színes képfeldolgozásban érdekelt scannergyártók fejlesztették ki 1986-ban elsősorban az asztali kiadványszerkesztés, illetve a digitális adatcsere szempontjait figyelembe véve. Ez a formátum az általános célú professzionális képfeldolgozás legelterjedtebb, platformfüggetlen lehetőségévé vált, köszönhetően annak is, hogy más területeken (távérzékelés, térinformatika, CAD) is használják. Széles körű elterjedését nagy mértékben elősegíti az állomány belső struktúrája. Nyitottsága révén a formátum rendkívül könnyen bővíthető. Magában a TIFF állományban tetszőleges szöveges információ is tárolható, kihasználva a formátum adta lehetőségeket.

  Néhány érdekesebb TIFF bővítés:

  – GeoTIFF, melyben az egyes pixelekhez a valós földrajzi helyhez kapcsolódó információk (transzformációs paraméterek, vetület, alapfelület) köthetők, a távérzékelésben és a térinformatikában alkalmazható elsősorban;

  – RichTIFF, mely eredetileg a professzionális DTP-eszközök (scannerek, levilágítók) egyik legismertebb gyártójának, a Crosfield cégnek a bővítménye, abból a célból, hogy az újságokban megjelenő képek adatcseréjét elősegítse.

  Már a nyolcvanas években is felmerült az igény egy olyan egységes raszteres formátum kialakítására, mely képes a térinformatika igényeihez igazodni. Mára a GeoTIFF már be tudja tölteni ezt a státuszt. A kezdeti internetes egyeztetések után 1995-ben egy SPOT konferencia alkalmából állapodtak meg az érintett cégek (USGS, Intergraph, ESRI, ERDAS, SoftDesk, MapInfo, NASA/JPL) a formátum egységesítésében. Mivel a TIFF raszteres formátum nyílt struktúrájú, így igazából azokban a bővítményekben kellett megegyezni, amelyek a TIFF állományon belül a térképészeti információkat tartalmazzák. Ezeket alkalmazhatjuk űrfotóknál, scannelt légifotóknál, digitális domborzatmodelleknél, vagy akár elemzések végtermékeinél is. A fő cél, hogy a raszteres állományhoz koordinátákat, vetületi információkat csatoljunk, és ezzel lehetővé tegyük, hogy a raszteres információ is a térinformatikai

rendszerek egyenrangú része legyen. A GeoTIFF nem módosítja a standard TIFF elemeket, mint például a színes megjelenítést vagy a belső adattömörítést és teljes egészében megfelel a TIFF 6.0 szabványban leírtaknak, sajátos bővítményei nem ellentétesek azzal. Olyan eljárásokat alkalmaz, amely nem használja ki a TIFF "titkos" képességeit, ezért könnyen készíthető hozzá megfelelő megjelenítő szoftver. Ha szükséges további funkciók beépítése, ezt a TIFF rendszer nyitottsága minden bizonnyal lehetővé teszi.

A GeoTIFF rendszer publikus, a teljes specifikáció elérhető az interneten. A TIFF nem nyomtatóvezérlő vagy oldalleíró nyelv, nem szándékozik általános adatcsere formátummá sem válni. Kifejlesztésekor elsődleges célnak tekintették a formátum funkciógazdagságát, hogy igazodhasson az eltérő tudású képfeldolgozó programok és a szintén rendkívül eltérő tudású scannerek (és hasonló eszközök) képességeihez. Az alkalmazott szoftvereknek sem jelent gondot, ha olyan TIFF jellemzőkkel kerülnek szembe, amelyeket a szoftver alkotóinak szándékai nem támogatnak, ilyenkor csak az alapvető (a szoftver által támogatott) jellemzők alapján végzik el a megjelenítést. A TIFF szándékai szerint platformfüggetlen, nem kötődik operációs rendszerekhez, processzorokhoz, fordítóprogramokhoz, értelmezőkhöz.

A TIFF állomány maximális mérete 2^{32} byte (több, mint 4 Gbyte), ez a méretkorlát az állomány tényleges nagyságára utal, valamilyen belső tömörítést alkalmazva valójában még ennél is nagyobbak lehetnek a TIFF állományok. A belső struktúrától függően beszélhetünk Microsoft, illetve Motorola (Apple-MacIntosh) byte sorrendről (általában az utóbbiba tartoznak a különféle munkaállomásokon létrehozott TIFF állományok is). A jelenlegi verzió az 6.0-s TIFF formátum, mely már olyan különleges formátumokat, effektusokat is támogat, mint például:

- 16 bites szürkeárnyalatos kép, 48 bites színes kép;
- raszteres kép hatékonyabb tárolása, mely elsősorban a nagyfelbontású képek gyorsabb hozzáférését segíti elő,
- javított *RGB* színkezelés;
- JPEG tömörítés;
- az RGB-től eltérő színrendszerek: CMYK, Lab, YCbCr (YUV).

A belső tömörítés többféle lehet, természetesen ezen módszerek mindegyike nem használható az összes korábbi TIFF formátumokban. Szerkezetét tekintve a TIFF állomány három fő részre osztható: az első egy rövid fejléc, a második egy könyvtár, ahol az állományban alkalmazott összes mező megtalálható, a harmadik rész tartalmazza az egyes mezők adatait. Ez a struktúra azt is lehetővé teszi, hogy egy állományban egyszerre több raszteres képet tároljunk, oly módon, hogy azok között tulajdonképpen semmilyen azonosság sincs (eltérő méret, felbontás, *színmélység*).

- BMP

  A BMP formátumnak négyféle változata van. Kettő a Windows grafikus környezetben (régi és új formátum) és kettő az IBM OS/2 operációs rendszere alatt. A Windows grafikus keretrendszer, illetve operációs rendszer a raszteres állományokat ebben az eszközfüggetlen raszteres formátumban tárolja. Az eszközfüggetlenség azt jelenti, hogy a módszer, ahogy az állományban tárolódnak az egyes pixelek színei, függetlenek a monitor színmegjelenítési módszerétől. Az állományformátum jellemzője, hogy minden egyes BMP állomány tartalmaz egy fejlécet, amely magában foglalja a színtáblát, illetve a színmélységet (hány bites képről van szó). A színtáblában a színek fontossági (gyakorisági) sorrendben jelennek meg, ami elősegíti gyors megjelenítésüket olyan eszközöket használva, melyek egyébként csak jóval kevesebb szín bemutatására képesek. A lehetséges színmélységek: 1 bites (fekete-fehér, monokróm), 4 bites (16 szín), 8 bites (256 szín), 24 bites (16,7 millió szín).

- JPG

  Mára a web legfontosabb grafikus formátumává vált a szintén platformfüggetlen JPEG File Interchange Format (JFIF). Rendkívül tömör formátum, de a tömörítés nem veszteségmentes, a kép minősége és az adathordozón elfoglalt terület nagysága egymással fordított arányban van. A legtöbb szoftver lehetővé teszi a felhasználó számára a tömörítési arány kiválasztását. A tömörítési mód, az ún. JPEG eljárás, nem áll szabadalmi oltalom alatt. A JPG állományok mentése, illetve betöltése alatt megy végbe a kódolás, illetve a dekódolás, de a gyors processzorok korában az ebből adódó sebességcsökkenés általában már észrevehetetlen. A rendkívül jó tömörítési arány titka, hogy a felbontástól függetlenül a képet 8×8 pixel nagyságú elemi területek alapján elemzi és átlagolja, tulajdonképpen az emberi szem számára kevéssé érzékelhető kis különbségeket kiszűri. Belső színkódolása a legtöbb raszteres formátumtól eltérően nem RGB alapú, hanem YCbCr. Mindenképpen legalább 256 színű (vagy szürkefokozatú) palettát tartalmaz, így elsősorban fényképek elterjedt formátuma. Vonalas jellegű rajzok (például bizonyos fajtájú térképek) ilyen formátumban történő tárolása nem az ideális megoldás. Elvileg támogatja a 32 bites színmélységet is, de leginkább a 8 és a 24 bites színmélység tekinthető szabványosnak. Szintén nem tekinthető szabványosnak az ún. progresszív JPG formátum sem, ahol a megjelenítés fokozatosan finomodik a végső kép megjelenéséig. A digitális fényképezés terjedésével a formátum jelentősége egyre növekszik, hiszen itt alapvető fontosságú, hogy a viszonylag korlátozott kapacitású háttértárolóra minél több képet lehessen lementeni. Az új változat a JPG2000 már támogatja a wavelet funkciókat is, mely további jelentős állományméret-csökkenést tehet lehetővé.

  A JPG esetében is lehetséges a kép földrajzi koordinátákhoz rögzítése, olyan módon,

amelyet tulajdonképpen minden más raszteres állomány esetében alkalmazhatunk, de a legelterjedtebb a JPG-nél. Az ún. "ESRI World" állomány egy egyszerű szöveges állomány, melynek neve megegyezik a JPG állomány nevével, kiterjesztése jgw, esetleg jpgw. Ez a szöveges állomány csak hat számértéket tartalmaz, amelyek a vetületi egyenletek paraméterezésére utalnak.

1998-ban fejlesztették ki a JPEG 2000 formátumot, amely lehetővé teszi a veszteségmentes tömörítést is. A legfontosabb újítás maga a belső tömörítési algoritmus, mely itt már wavelet alapú. Lehetséges speciális adatok tárolása a képpel együtt, erre 256 csatornát kínál a formátum. Szintén sokat javítottak a színkezelésen, ennek megfelelően mind az adatbeviteli (scanner), mind a megjelenítési oldalon (képernyő, nyomtató) a felhasznált eszközhöz igazodik a színek használata.

- PNG

  Ezt a viszonylag új raszteres formátumot a web jobb kiszolgálása érdekében alkotta meg a W3 Consortium erre a célra megalakított csoportja. Az 1.0-s verzió 1996. októberében született meg. A formátumot úgy alkották meg, hogy egyesítse a GIF és a JPG formátumok egyes előnyeit (tömörség, veszteségmentes tömörítés, átlátszóság). Az alkalmazott tömörítési algoritmust nem védi szabadalmi oltalom, így széles körű elterjedésének ez sem szabhat gátat. Ennek ellenére a böngészők közül a Netscape-nek csak az 1998-ban megjelent 4.5-ös változata támogatta először a formátumot, de a Microsoft Internet Explorer esetében annak Windows98-ba beépített változata csak külső program segítségével tudta megjeleníteni, természetesen az újabb verziók már külső segítség nélkül is be tudják tölteni. Támogatja a korlátozott színpalettájú, a szürkefokozatos és a valós színes megjelenítéseket színkomponensenként 1, 2, 4, 8, 16 bit mélységben (színmodellektől függően nem minden esetben az összeset). A webes adatátvitelre tekintettel teljes állományintegritás-ellenőrzést biztosít és képes a gyakori átviteli hibák felderítésére.

1.2. Vektoros (tér)kép [4, 30–33. o.]

A vektoros adatstruktúra lényege, hogy a grafikus objektumokat jellemző pontjaik koordinátáival tároljuk. Alapvetően háromféle objektumtípus létezik a vektoros rendszerekben, de szükség szerint létezhetnek további speciális objektumok is (szöveg, blokk stb.):

- Pont

  A pont a térképen méretlehelyesen általában nem ábrázolható objektum, aminek helyét koordinátáival kell definiálni. Ha az objektumot meg akarjuk különböztetni a többitől, akkor meg kell különböztetni az eltérő típusba sorolható többi pontszerű

objektumtól. Ennek a legegyszerűbb, leghagyományosabb módja a térképjelek alkalmazása (1.3. ábra). A térképjel az objektum valós kiterjedésénél nagyobb területet fed le a térképen, mivel leggyakrabban ún. egyezményes jeleket használnak pontszerű tereptárgyak jelölésére (pl. jellegzetes fa, gyárkémény, emlékmű).

1.3. ábra. Néhány térképi jel (a jobb láthatóságért felnagyítva)

A vektoradatmodellben egy pont leírásához a következő alapvető információkat kell tárolni (zárójelben az opcionális információk találhatók):

- (Azonosító)
- Attribútum
- X és Y koordinátapár
- (A koordináta-rendszer paraméterei, utalás vetületre, alapfelületre)

- Vonal, poligon

  A vonalszerű objektumokat töréspontjaik koordinátáival tároljuk. Ily módon egy vonal pontok sorozatára vezethető vissza. A grafikus megjelenéstől függetlenül minden esetben csak a vonalas jel tengelyvonalát tároljuk egyszerű sokszögvonalként, illetve megfelelő szoftver esetén a professzionálisabb megjelenítés érdekében *Bézier-görbékkel*. A megjelenítés azonban jóval bonyolultabb problémákat vet fel, hiszen a vonalas objektumok esetében szinte minden esetben méreten felüli ábrázolást alkalmazunk. Például egy I. rendű főutat egy 1 : 10 000 méretarányú térképen 0,12 centiméter vastag vonallal ábrázolunk az utak hierarchiája miatt a nyomtatott térképen. Azonban ezt az értéket visszaszámolva a méretarány függvényében azt kapjuk, hogy ezen főút valódi vastagsága 1200 centiméter, azaz 12 méter lenne. Azonban tudjuk, hogy ez igen nagy szám a valódi értékhez képest.

  Egy vonal leírásához a következő alapvető információkat kell megadni (zárójelben az adatstruktúra megvalósítási módjától függő információk találhatók):

  - (Azonosító)
  - Attribútum
  - Pontazonosító 1, pontazonosító 2, pontazonosító 3, ...
  - (A koordináta-rendszer paraméterei)

- Felület

  A felületek tulajdonképpen a vonalakra vezethetők vissza, hiszen minden felületet vonalak határolnak. A felületek már rendkívüli bonyolultságúak lehetnek: több,

egymással nem határos felület ábrázolhat egy logikai egységet (pl. Japán szigetei); lehetnek a felületben lyukak. A térinformatikában gyakran elengedhetetlen olyan információk tárolása is, hogy a felület határainak másik oldalán milyen felületi objektum található. Egy felület leírásához a következő információkat kell megadni (zárójelben az adatstruktúra megvalósítási módjától függő információk találhatók):

- (Azonosító)
- Attribútum
- A felületet felépítő határoló vonalak azonosítói: vonalazonosító 1, vonalazonosító 2, vonalazonosító 3, ...
- (A koordináta-rendszer paraméterei)

Különleges problémát jelentenek a nem-folyamatos felületek, vagyis amelyek lyukakat, szigeteket tartalmaznak. Ilyen esetben a fentinél még összetettebb információkat kell megadni a felület pontos leírásához. A felületek, vonalak és pontok egymásra épülésének rendszerét topológiának nevezik. A topológia megvalósítására nincs általános érvényű szabvány, minden egyes nagyobb szoftvergyártó kialakította saját megoldását.

A vektoradatmodell egyik problémája éppen ebből adódik: a komplex topológia sokszor problémát okozhat a különböző szoftverek közötti adatcserében, de akár egyes térképmanipulációk megvalósításában is egy adott szoftveren belül. A vektoros adatstruktúra másik problémája a tárolt objektumok méretarányhelyes rajzi megjelenítése. Pontszerű objektumok esetében egy pont koordinátái tárolódnak adatként, de az ehhez tartozó objektum grafikus megjelenítése a térképen (a térképjel) lefoglal egy bizonyos nagyságú területet. Hasonló a probléma a vonalas objektumok esetén is: a vonalas objektumok töréspontjait tároljuk, de a grafikai megvalósítás mindenképpen valamilyen grafikai attribútumokkal bíró vonal a térképen, melynek vastagsága minden bizonnyal nagyobb lesz a terepi objektum szélességénél. Generalizálási probléma is felmerülhet: gondoljunk csak egy olyan egyszerű esetre, hogy hogyan ábrázoljunk a térképen egy közvetlenül egymás mellett futó műutat és vasutat. A koordináták alapján (azaz a valóságban) ezek egymástól való távolsága esetleg csak néhány méter, de a grafikai megjelenítés helyigénye miatt méretaránytól függően ennek a távolságnak a térképen a sokszorosára kell nőnie.

Ha a vektoros állományokat a képernyőn is meg szeretnénk jeleníteni, akkor figyelembe kell venni a képernyő korlátozott felbontóképességét és a térképi adatállomány részletességét. A monitor raszteres elven működik, a felbontóképesség, mint technikai korlát megszabja a képernyőn egyszerre megjeleníthető (pontosabban észlelhető) adatmennyiséget. A vektoros térképek esetében, az alkalmazott struktúrától függően, folyamatosan nagyítható-kicsinyíthető a képernyőn látható kép. Ha a vonal- és felülettípusú objektumokat sokszögvonalként tároljuk, akkor az egyre növekvő nagyítási fokozatokban a vonalak szögletessé válnak, mintegy arra utalva, hogy már elértük (meghaladtuk)

a felvételi pontosságot. Ha a vonalakat Bézier-görbék formájában tároljuk, akkor ilyen jellegű visszajelzést még közvetett módon sem kaphatunk.

Alapvető fontosságú, hogy ismerjük annak a térképnek a méretarányát, amelyik a digitális térkép alapjául szolgált: a vektoros térképeket leginkább úgy jellemezhetjük, hogy megadjuk, milyen méretarányú hagyományos térképnek felel meg az adattartalma, a részletessége. Például egy 1 : 1 milliós méretarányban digitalizált térkép alapjául szolgáló papírtérképek eredetileg 1 : 500 000 méretarányúak voltak, a digitális térkép generalizálás révén jött létre. Ha ezt a térképet a képernyőn 1 : 100 000 méretarányúra nagyítjuk fel, akkor a vártnál kevesebb részletet fogunk kapni, és rajzi vonalai ebben a rendkívül felnagyított méretarányban már láthatóan sokszögvonalakká esnek szét. A digitalizálási méretarányból következő probléma úgy is megoldható, ha az eltérő részletességű információk külön rétegeken helyezkednek el, pl. a kis folyók csak akkor jelennek meg a képernyőn, ha a nagyítás (tulajdonképpen a méretarány) elér egy előre definiált küszöbértéket (azaz méretarányt). Ezért fontos, hogy a vektoros adatbázis legalább annyi rétegből álljon, ahányféle térképi objektumtípust ábrázol. Ez a lehetőség is csak bizonyos méretarány-tartományban alkalmazható megfelelően, hiszen az egyes térképi objektumok ábrázolásának részletessége a digitalizáláskor már eldőlt, ez tovább nem finomítható. Az állami topográfiai alaptérképek vektorizálása ma a legtöbb országban alapvető fontosságú feladat. A legfejlettebb országokban ezek a vektoros térképek már többféle méretarányban is elkészültek, sőt jónéhány országban már a teljes méretarány-sorozat digitalizálása megtörtént és a naprakészen tartás is már teljes egészében digitális alapokon működik. Magyarországon jelenleg az 1 : 50 000 méretarányú katonai topográfiai térkép az egyetlen, amelynek a teljes tartalma, az ország területét hiánytalanul lefedve, vektoros digitális formában is hozzáférhető.

Fontosabb vektoros formátumok [4, 72–74. o.]

- DWG

  A DWG (Autodesk Drawing) formátum az Autodesk natív, belső (bináris) állományformátuma, mely népszerűségét a cég AutoCAD nevű mérnöki tervező programjának köszönheti. A DWG formátum általában az új programverziók megjelenésekor kicsit megváltozik, továbbfejlődik, de természetesen minden újabb verzió olvassa a régebbi változatokat. Az utolsó jelentős változás a 12-es és a 13-as verzió (1995) között volt: a 13-as verzió formátuma gyökeresen megváltozott a korábbi verziókhoz képest, mind a DWG, mind a DXF formátumot tekintve. A DXF formátum CAD és GIS környezetben a poligonalapú grafikus információk szabványa. Bináris változata a DXB, tömörebb, gyorsabb értelmezést tesz lehetővé, de egyre kevésbé használják. A DWG és a DXF formátumok gyakorlatilag ugyanazt a grafikus információt tárolják, de a DXF állományok – lévén tiszta szövegállományok – jóval nagyobb méretűek, igaz adatcsere szempontjából a DXF

sokkal fontosabb. Az AutoCAD a 13-as verzió óta már támogatja a Bézier-görbéket, a speciális 3D-s bővítéseket, sőt a raszteres állományok is beágyazhatók. Az AutoCAD 2002 DXF formátuma az előző változatokéra épül. Az állomány alapelemei a csoportkódok és az ehhez tartozó értékek. Az adott kód már meghatározza a hozzá tartozó érték típusát is. A kódokat a könnyebb áttekinthetőség kedvéért célszerű csoportokba szervezni. A struktúra a következő:

- fejléc (header): általános információkat közöl (pl. verziószám, rendszerváltozók);
- osztályok (classes): tartalma zömmel előre meghatározott;
- táblázatok (tables): többféle információt tartalmazhat táblázatos formában (rétegek, stílusok, vonaltípusok);
- blokkok (blocks): speciális objektumfajta, alkalmas pl. összetett térképjelek definiálására;
- entitások (entities): a grafikus objektumokat írja le;
- objektumok (objects): a nem grafikus objektumokat sorolja fel;
- előkép (thumbnailimage): ez az opcionális rész a rajz egy előképét tartalmazza, hogy látható legyen a DXF állomány közelítő tartalma a teljes betöltés előtt.

A DXF fájl részei (szekciói) elhelyezkedés és elnevezés szempontjából egy verzió vonatkozásában kötöttek, de a verziószám növekedésével újabb csoportok jelenhetnek meg: például az objektumok szekció, mely a többszörös vonalak paraméter hivatkozásait is tartalmazza, vagy az osztályok szekció csak a 13-as verziótól kezdődően található meg a formátumban. Ez a bővítési lehetőség azonban kétélű fegyver, mivel a bővített DXF-eket a régebbi verziók, illetve a transzformációs programok nem értik. Ezt megkerülendő mind az exportnál, mind az importnál lehetőség van arra, hogy a fájl csak az ún. entitás szekciót tartalmazza. Összefoglalva:

- a fájlban található összes információt csak az AutoCAD megfelelő verziója képes kihasználni;
- ez a 2 dimenziós GIS rendszerek szempontjából nem igazán problematikus, mivel a változók, illetve táblázatok jelentős része a kótázást, a megjelenítést és a rajzi minőséget szolgálja, az utóbbi célokra a GIS szoftver saját eszközökkel rendelkezik, kótázásra pedig a GIS-ben alig van szükség;
- ugyanakkor a DXF alkalmas a teljes adatmodell átvitelére, ami egyrészt a 3 dimenziós objektum leírásban, másrészt a tömbökkel jellemzett összetett objektumokban, a rétegszerkezetben és – az újabb (13-assal kezdődő) verziókban – az attribútum táblázatok alkalmazhatóságában jelentkezik. A fentiekből két dolog következik: a DXF konvertáló programokat mindig az adott GIS

szoftver adatmodell, illetve grafikai szintjén kell megírni, másfelől tudatában kell lennünk annak, hogy a DXF konvertáló programok ezt a követelményt különböző szinteken teljesítik. A DXF rendkívül szigorú és bonyolult struktúra, ezért viszonylag nehéz olyan programot kézszíteni, amely szabályosan értelmezhető állományt állít elő. Ha az AutoCAD nem szabványos vagy értelmetlen szekciókat, sorokat talál, kihagyja a problémás részek értelmezését.

- HPGL

  A HPGL plottervezérlő nyelv a plotterek működési sajátosságaiból adódóan csak egyenes vonalak rajzolására képes, a görbéket sokszögvonalakra bontja. Szintén komoly korlátozás, hogy felületkitöltésre csak különféle sraffozási lehetőségek állnak rendelkezésünkre. Maga a nyelv tulajdonképpen rendkívül egyszerű utasításokból áll: vonalhúzás (írófej a papíron), tollmozgatás (írófej a papír felett), illetve a fejlett plotterek esetében tollcsere (eltérő színek használata). Elvileg 255 féle színű toll használható, gyakorlatilag a drágább modellek is csak 8 tollat tudtak akkoriban egyszerre kezelni. A nyelv bemutatkozása a HP 7475A típusú plotter megjelenéséhez köthető, fejlettebb változata az egyelőre kevéssé elterjedt PGL/2. A HPGL formátum tiszta ASCII állomány. Ma már mind a különféle gyártóktól származó olcsóbb lézernyomtatók, mind a tintasugaras nyomtatók zömmel értik, illetve támogatják a PCL nyelvet. Az újabb és fejlettebb HP nyomtatók megjelenésével maga a PCL nyelv is folyamatosan változik, bővül.

- DGN

  CAD és GIS területen széles körben elterjedt ez a formátum, az eredeti programot az Intergraph, majd a Bentley cég fejlesztette. Tulajdonképpen nem is egy formátumról (DGN) van szó, feltétlenül meg kell említeni az ún. CEL állományokat is. A DGN állományok az ún. designfájlok, melyek a grafikus elemeket továbbá a nem grafikus adatokat tartalmazzák, beleértve a felhasználó által definiált elemeket is. A cellakönyvtárakban a designállományokban elhelyezett cellák definíciói találhatók. A cellaleírások egymásba ágyazhatók. A MicroStation legújabb változata a V8 (2001) már olyan szinten támogatja a DWG formátumot, hogy ezzel megpróbálja a felhasználókat átcsábítani az AutoCAD-ről. Sajnos maga a DGN formátum is jelentősen megváltozott ebben a V8-ban, ami a más programokkal való kompatibilitást nehezíti. Ez a változás már várható volt, mert a DGN magja már régóta változatlan volt. A legfőbb újdonság a DGN esetében, hogy nincs korlát a pontosság (az AutoCAD-del szemben a sokkal kisebb pontosság volt az egyik nagy hátránya a MicroStation-nek), a rétegek száma (max. 63 helyett ezentúl 4 milliárd), a komplex láncokban lévő komponensek száma és az állományméret (itt 32 MB volt a maximális méret) tekintetében. A korábbi MicroStation verziók közül talán az 5-ös volt a legnépszerűbb. A V8 legfeljebb a V7 formátumában engedi meg a DGN

állományba mentést (természetesen egyes speciális funkciók elvesztése árán), de azért megnyitja a korábbi verzió állományait. Az "erőviszonyokat" jól mutatja, hogy az AutoCAD-ben nincs lehetőség DGN formátumba mentésre, míg a MicrosStation korábbi verziói is lehetővé tették a DXF vagy DWG formátumba való konverziót.

2. fejezet

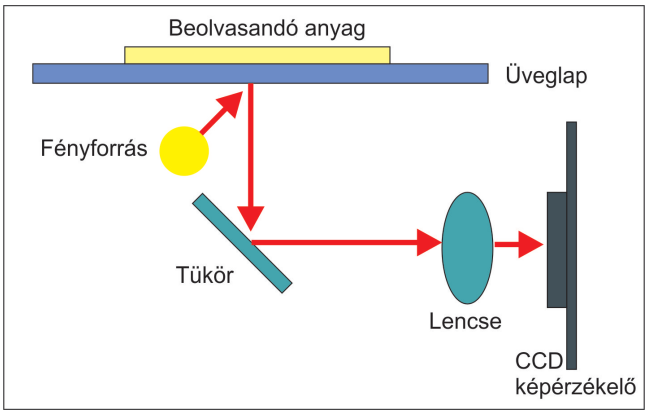
Adatok bevitele

2.1. Scannerrel

A scanner működési elve */vázlatos bemutatás/*

Bár a scannereknek több típusa létezik (síkágyas, dob, kézi, lapáthúzós), az általunk leginkább alkalmazott eszköz a síkágyas scanner [6]. Ezen scanner alapelemei a következők:

- az olvasófej, amely egy fénycsövet és egy tükröt tartalmaz,
- az üvegfelület, amelyre a beolvasandó anyagot tesszük,
- az érzékelő,
- a fejmozgató motor,
- az elektronika.



2.1. ábra. A scanner működése

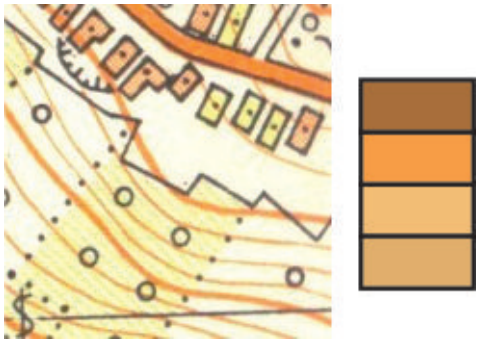
Az olvasófejet a léptetőmotor bordásszíj segítségével mozgatja fémsíneken az üveglap alatt. A fejegység fénycsöve alulról megvilágítja a beolvasandó anyagot, majd a visszavert fényt a tükör (egyes eszközöknél több tükör is lehetséges) segítségével egy lencsén

keresztül (amely a kép kicsinyítését végzi) a scanner belsejében található, fix pontra rögzített *CCD érzékelőre* fókuszálja. Majd az érzékelő digitális képpé alakítja a beérkező fényt (2.1. ábra). A fénycső előnye a széles spektrumú fény kibocsátása, így viszonylag jó színhűség érhető el.

Scanneléskor előforduló hibák

Scannelni kívánt anyag hibái

Mint már említettem, az új térkép készítéséhez többnyire analóg, már kinyomtatott térképet használnak alapul. Figyelembe kell venni ennek az alaptérképnek a minőségét is, ami nyomtatás után változhat [5, 7. o.]. Maga a nyomtatás úgy történik, hogy a különböző színű rétegeket egymásra nyomják. Ebből következik, hogy egy már kinyomott kék színre ha következőleg sárgát nyomtatnak, akkor az így előálló kék *színárnyalata* más, mintha ugyanerre a kék rétegre egy piros szín kerülne. Emberi szemmel könnyen megállapítható, hogy ugyan arról a kékről van szó, ám vektorizáláskor a program automatikusan már nem képes felismerni a látszólag kétfajta kék ugyan azon voltát. A 2.2. ábrán látható főszintvonal is jó példa erre.



2.2. ábra. Elméletileg egyszínű főszintvonal pixeljei különböznek egymástól

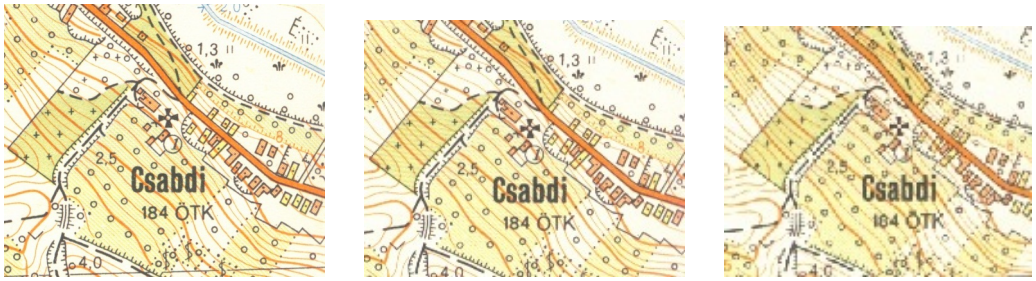
Eszköz működése közben fellépő hibák

A fentebb említett scanner működése közben különböző hatások érhetik az eszközt: a tükrök a mozgás hatására idővel elállítódhatnak, szennyeződhetnek, és ezáltal a beolvasott kép minősége, élessége romlik. A lapolvasók beolvasott képének minősége igen eltérő. Ennek oka többek között a fejléptető motor, mely vibrációkat kelt. Ez – alig észrevehetően – berezonáltatja az egész készüléket, ami miatt a kép minősége/élessége szintén romlik. A rezonanciák keltette káros bemozdulást ki lehet védeni a készülék megfelelő súlyával (minél nehezebb egy scanner, annál kevésbé befolyásolja károsan a rezgés), valamint a megfelelő kialakítású gumitalpakkal. A fejegységet mozgató sínrendszer minősége is lényeges, hiszen minimális görbületek vagy anyaghibák megint pontatlan olvasást eredményezhetnek. Ott van még a tükör, az érzékelő, az elektronika és az optika (lencse). Minél egyenletesebb a tükör felülete, minél zajmentesebb az érzékelő és az elektronika, s minél

jobb minőségű a lencse, annál jobb minőségű lesz a beolvasott kép. Nem mindegy, milyen az adott lapolvasó optikai felbontása. Egy másik probléma lehet a por megjelenése az eszköz belsejében. Ezt lehetetlen elkerülni, hiszen hiába gondoljuk, hogy a scanner belseje tökéletesen védett a külső hatásokkal szemben, tapasztalhatjuk, hogy kis idő múltán a leglehetetlenebb helyen is megjelenik. Sőt, sokszor még emberi hajszálat is találhatunk az eszközben.

Emberi mulasztás során fellépő hibák

Emberi mulasztáson értem a felhasználó figyelmetlenségét. Egy lehetséges hiba a nem megfelelő felbontás beállítása scanneléskor. A scannerek maximális felbontóképessége ma már nem lehet akadály. Azonban nekünk felhasználóknak tudnunk kell, mikor, milyen célra mekkora felbontást állítunk be. Az alábbi képsoron ugyanazon terület látható, más-más felbotás mellett.



2.3. ábra. Ugyan azon terület 300, 150 és 100 dpi-vel scannelve

A scannelni kívánt anyag behelyezésekor is követhetünk el hibákat:

- Az anyagot célszerű az üvegfelülettel práhozamosan behelyezni a scannerbe. Megnehezíti a feldolgozást, ha bizonyos szöggel elforgatva tesszük ezt meg.
- Figyelnünk kell, hogy a scanner fedelét lecsukva nehogy gyűrődjön a scannelni kívánt állomány,
- vagy épp le ne csússzon a beolvasandó felület az üvegfelületről.
- Scannelés közben tartsuk fixen az anyagot, mert az esetleges elcsúszás miatt elmosódhat a tartalom.

Megfelelő minőség beállítása

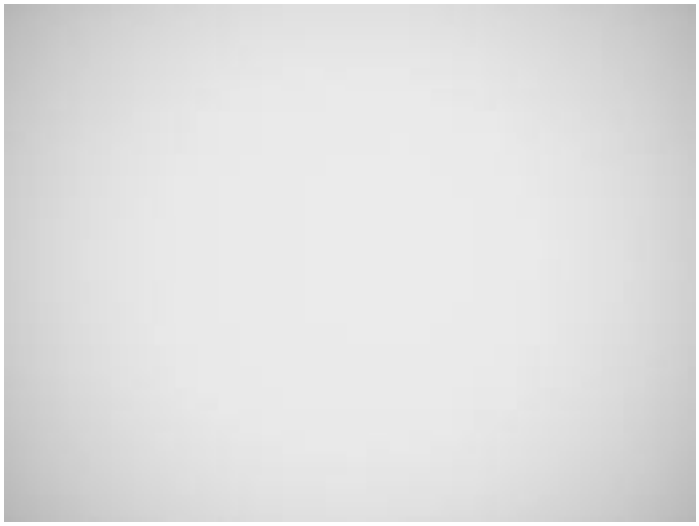
A megfelelő minőség célhoz köthető. Otthoni felhasználásra nincs értelme fölöslegesen valódi 42 bites vagy afölötti színmélységű scannert vásárolni, ugyanis a kijelzők (pl. monitor) többsége amúgy is csak 24 bites színmélység megjelenítésére képes. A magasabb bitértéken való működés több számítást vesz igénybe, ezáltal lassabb is. Térképek scannelésekor amúgy sem a színmélység a mérvadó, sokkal inkább a digitális állomány

majdnai felbontása. A szokványos, asztali felhasználásra (fényképek, térképek beolvasása, dokumentumok digitalizálása) tulajdonképpen bőven elég az optikai 300 és 600 *dpi* felbontás. Azonban filmek, negatívok, vagy diák scanneléséhez már minimálisan elvárható az 1200 dpi.

A lapolvasó legfontosabb értéke a beolvasott kép digitális másának minősége, azaz az eredetinek megfelelő színkép, a jó képélesség, a zajmentesség. Másik fontos tényező az idő. Nem mindegy, hogy adott feladatot egy vagy tíz perc alatt sikerül elvégezni. Ez a felbontással arányosan növekszik. Ezért a scannerek lapolvasási ideje is igen fontos tényező.

2.2. Digitális fényképezővel

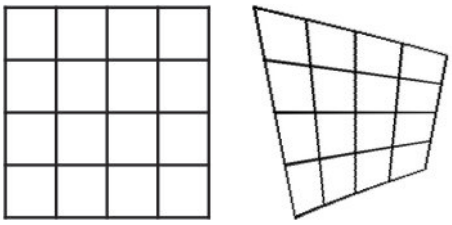
Analóg térképet digitalizálhatunk akár digitális fényképezőgéppel is, ám a fellépő nehézségek miatt nem érdemes ezt a módszert választanunk. A három fő jelentkező probléma a perspektív torzulás, a radiális torzulás és a vignettálás jelensége, ami a 2.4. ábrán[1] látható. A kép szélei felé sötétedés figyelhető meg. Ez vakuval történő fényképezés során jelentkezik, és – ellentétben a scannerrel, ahol egyenletes a fényeloszlás az adatbevitelkor – a majdani vektorizálás során komoly nehézségeket okozhat. Például az elméletileg mindenhol azonos színű vízrajz a térkép széle felé sötétebb lesz, ott a pixelek más intenzitás-értéket kapnak, nem lesz homogén, ezért pontosabb utófeldolgozást igényel.



2.4. ábra. A vignettálás jelensége

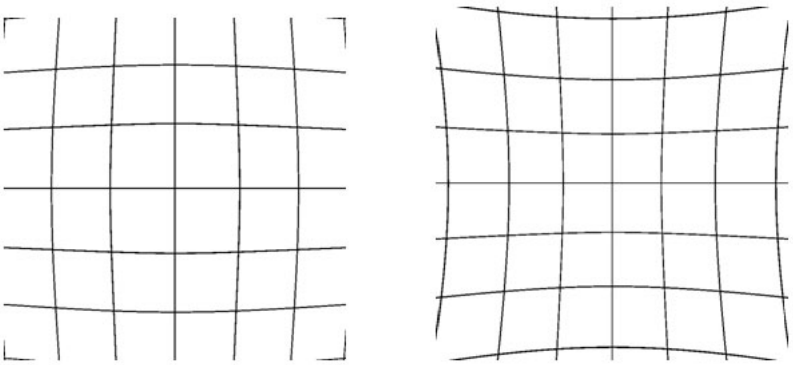
A perspektv torzulás akkor keletkezik, ha a fényképezés pillanatában a fényképező tengelye nem pont merőleges a térképlapra (2.5. ábrán erős túlzással). Ezt kiküszöbölhetjük megfelelő rögzítéssel.

[1]Forrás: `http://pixinfo.com/img/Olympus/SP-565UZ/a/v_end_open.jpg`



2.5. ábra. A normál és a perspektív torzulással terhelt kép

A radiális torzulásnak két típusa van: az úgynevezett hordó és tűpárna (2.6. ábra[2]). Hatásuk a fényképező lencséjének típusától függ. A hordó-effektus általában a nagy látószögű fényképezőknél jelentkezik, míg a tűpárna-effektus a kis látószögűeknél. Ezen hatások főleg a légifényképezéskor hatnak a képre, hétköznapi fényképezés során elhanyagolható a hatásuk.



2.6. ábra. A hordó-effektus és a tűpárna-effektus

A fentebb felsorolt problémák megoldása az ortofotó készítése. Ennek leírása azonban nem képezi a szakdolgozatom részét, és a téma szempontjából nem is fontos.

[2]Forrás: http://1.bp.blogspot.com/_CqfX2aO_9nY/SOuW-rrBamI/AAAAAAAAOxk/1tIXXYx0cmg/s400/tech_hemi_08.jpg,
http://3.bp.blogspot.com/_CqfX2aO_9nY/SOuW-B5NoYI/AAAAAAAAOxc/VartClH5IxE/s1600-h/tech_hemi_09.jpg

3. fejezet

Raszteres képek átalakítása

Scannelés után a keletkező digitális állomány – ahhoz, hogy vektorizálható legyen – általában át kell essen egy előfeldolgozáson. Az előfeldolgozás során kiszűrjük a keletkezett képi hibákat, egységes pixel-csoportokat hozunk létre, kiemeljük az éleket, topológiát építünk, majd vektorokkal létrehozzuk a kívánt vonalakat, felületeket.

3.1. Digitális szűrők

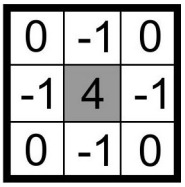
Ahogy a 2.2. ábrán is láthatjuk, egy vonalat többféle intenzitású pixel épít fel. Sőt, képformátumtól függően az adott formátumhoz tartozó tömörítő eljárás miatt különféle foltok is keletkezhetnek a képen. Ha a scanner üveglapja nem volt tiszta, esetleg karcos volt a scannelés alatt, akkor a keletkezett állomány minőségét ez is befolyásolhatja, természetesen negatívan. Ahhoz, hogy ezeket a hibákat eltüntethessük, különböző szűrési eljárások alá kell vetni a képet.

5	3	8	11	12	3	7	6	⋯
6	2	7	4	9	8	2	0	
1	13	9	10	6	5	3	9	
8	3	5	4	4	2	7	1	
10	7	0	2	3	8	6	5	
⋮								

3.1. ábra. A raszteres kép pixeljei intenzitásértékekkel

Ezek a szűrők a következő képpen működnek: adott a raszteres kép, ahol minden egyes pixel egy intenzitásértékkel bír (3.1. ábra). Adott továbbá egy $n \times n$-es[1] *kernel*mátrix (3.2. ábra), ami végigfut a kép minden során, és a kernel közepére eső aktuális pixel

[1] A kernelmátrix mérete tetszőleges lehet, de ajánlott a $(2n+1) \times (2n+1)$-es mátrix alkalmazása, ahol $n \in \mathbb{Z}^+$.



3.2. ábra. A 3 × 3-as kernelmátrix

értékét módosítja valamilyen eljárás útján. Ezen eljárásoknak több változata is van, az alábbiakban bemutatom a legfontossabakat.

Megjegyzés. A 3.1. ábrán a vörös keret a kernel aktuális helyét mutatja a képen, míg a vörös kitöltésű négyzet azt a pixelt jelöli, amire a kernel épp hatni fog. A kernelt ábrázoló 3.2. képen a szürke négyzet esik rá a raszterkép piros négyzetére. A kernelmátrixban található értékek a kernel funkciójától függenek.

Ha jól szemügyre vesszük a 3.1. ábrát, észrevehetjük, hogy ha a kernel közepe a kép szélén levő pixelek valemelyikén áll, akkor maga a kernel lelóg a képről. Ez elkerülhetetlen, csak részben tudjuk korrigálni a problémát. Ilyenkor az lehet egy megoldás, ha a kép szélein lelógó pixeleket a legközelebbi (szomszédos) pixelekkel pótoljuk.

3.2. Lineáris szűrők

A lineáris szűrők – másnéven konvolúciós szűrők – úgy működnek, hogy a kernelmátrix értékeivel súlyozottan számítják ki az aktuális pixel új értékét, mint átlagot a szomszédos pixelek figyelembevételével.

$$\sum_{i=x-n,j=y-n}^{i=x+n,j=y+n} f(i,j) \cdot w(i+n,j+n) \tag{3.1}$$

A fenti képlet az általános eljárást mutatja, ahol az $f(i,j)$ a kép aktuális pixelje, a $w(i+n,j+n)$ pedig a kernelmátrix aktuális értéke. x és y változók a kernel pillanatnyi középpontjának koordinátái. $n = 1$ esetén a kernelmátrix nagysága 3, míg $n = 2$ esetén 5.

Szeparábilis szűrők

A szeparábilis szűrők olyan szűrők, ahol a kernelmátrix felbontható egy sor- és egy oszlopvektor szorzatára. Ekkor a konvolúció végeredményét sokkal gyorsabban ki tudjuk számolni, mert gyorsabb a szűrő lefutása. Ilyen esetben először kiszámítjuk a kép konvolúcióját a sorvektorral, majd az eredményt konvolváljuk az oszlopvektorral.

$$w\left(i,j\right) = u\left(i\right) \cdot v\left(j\right), \tag{3.2}$$

ahol $w(i,j)$ a kernelmátrix, $v(i), u(j)$ pedig a sor- és oszlopvektorok.

Box szűrő

A box szűrőben a kernel értékei megegyeznek. A kernel végigfutása nyomán a kép pixelei átlagolódnak. Hatása a képre, hogy a képet simábbá teszi, a kiugró intenzitásértékek megszűnnek. Jó tulajdonsága a szűrőnek, hogy az éleket megtartja.

Gauss szűrő

A Gauss szűrő is egy simító szűrő, kerneljében a Gauss elolszlás értékei szerepelnek, a kernel nagyságától függően. Minél nagyobb a kernel mérete (n), annál hatékonyabb a simítás. A szűrő egy érdekes tulajdonsága, hogy vele gyorsan elő tudjuk állítani a kép kicsinyített változatait. Ez úgy működik, hogy a képre alkalmazzuk a szűrőt, majd minden második sort és oszlopot elhagyva megismételjük a műveletet. Ekkor a képekből előáll az úgynevezett Gauss-piramis. Ha az egymás feletti kép-rétegeket kivonjuk egymásból, akkor a kép heterogén részei jól felerősödnek, ami jól használható a textúrák detektálásánál.

Nemszeparábilis szűrők

A nem szeparábilis szűrők olyan szűrők, ahol a kernel mátrixa nem írható fel két vektor szorzataként.

Laplace szűrő

A Laplace szűrő kerneljének értékeit az alábbi képlet adja meg.

$$L\left(f\left(x,y\right)\right)=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}. \tag{3.3}$$

Ezt diszkretizálva kapjuk a következő képletet:

$$f''\left(x,y\right)=f\left(x-1,y\right)+f\left(x+1,y\right)+f\left(x,y+1\right)+f\left(x,y-1\right)-4\cdot f\left(x,y\right), \tag{3.4}$$

amiből pedig az alábbi kernelmátrixot kapjuk:

0	1	0
1	-4	1
0	1	0

3.3. ábra. A Laplace szűrő kernelmátrixa

A szűrő a kép eredeti és simított változatának különbségét adja meg, tehát jól mutatja az intenzitás-változásokat és a hibákat. Gauss szűrő után alkalmazva kiváló éldetektáló.

Emboss szűrő

Az emboss szűrő is az intenzitás-változásokat detektálja, a kerneljében egy-egy átellenes sarokpontban 1 és -1 van. Attól függően, hogy melyik sarokpontban vannak ezek az értékek, az arra merőleges élekre reagál főleg.

1	0	0
0	0	0
0	0	-1

3.4. ábra. Az emboss szűrő kernelmátrixa

Kép élesítése

A fentebb felsorolt szűrők segítségével már lehetőségünk van a kép élesítésére is. Ehhez kell az eredeti kép és annak simított változata. Először az eredeti képből "kivonjuk" a simított változatot, majd az így eredményül kapott képet hozzáadjuk az eredeti képhez. Képletben ez a következőképpen néz ki:

$$g(x,y) = f_e(x,y) + (f_e(x,y) - f_s(x,y)), \tag{3.5}$$

ahol g az újonnan kapott éles kép, f_e az eredeti kép, f_s pedig a simított változata. Simító szűrőnek alkalmazhatjuk bármelyiket a fentebb felsoroltak közül.

3.3. Nemlineáris szűrők

A nemlineáris szűrők olyan eljárások, melyek hasonlóan működnek az eddig felsorolt lineáris szűrőkhöz, a különbség azonban, hogy a szomszédos pixelekből számolt értéket nem lineáris kombinációval számolják, hanem más módszerrel. A kernelmátrix fogalma helyett csak kernelablakról beszélhetünk, értékei nincsenek. Az alábbiakban bemutatok néhány nem lineáris szűrőt.

Rank szűrők

Rank szűrők esetében a kernelablak alatt található pixelértékeket nagyság szerint növekvő sorba állítjuk, majd ezen sorrend alapján választjuk ki az új pixelértéket a kernelablak közepén található pixel helyére.

Medián szűrő

A leggyakrabban használt rank szűrő a medián szűrő. Lényege, hogy a nagyság szerint sorba állított pixelértékek közül kiválasztja a nagyság szerinti középső értéket, és ez lesz

az új pixelérték. Például a 3.5. ábrán a kernelablak alatti számokból az új pixelérték az 5 lesz, hiszen ha sorba állítjuk az értékeket, akkor kapjuk, hogy $2, 4, 4, 4, 5, 8, 9, 10, 11$, ahonnan a középső érték az 5 lesz.

5	3	8	11	12	3	7	6	···
6	2	7	4	9	8	2	0	
1	13	9	10	11	5	3	9	
8	3	5	4	4	2	7	1	
10	7	0	2	3	8	6	5	
⋮								

3.5. ábra. Kernelablak medián szűrőnél

A medián szűrő is simító hatású, jól használható az úgynevezett salt & pepper típusú hibára. Ekkor ugyan is az oda nem illő, kiugróan magas értékkel bíró pixeleket a sorba állításnál a sor szélére tolja, így azok kiesnek. Jó példa a salt & pepper típusú hibára egy koszfolt a scanner üveglapján, ami egy fekete pontként jelenik meg az állományban.

Előnye a medián szűrőnek, hogy $2n + 1$ méretű kernelablak esetén a k-nál vékonyabb vonalakat eltünteti. Ez a nagy területek kiemelésekor hatásos lehet. Hátránya, hogy az éleket eltolhatja és a sarkokat lekerekíti, de ez orvosolható probléma.

Konzervatív szűrő

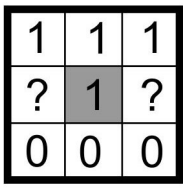
Ez is egy simító szűrő, mely hasonlóan működik a medián szűrőhöz. Az eljárás során itt is sorba rendeződnek a kernelablak alatti pixelértékek, kivéve a kernel közepe alatti pixel értékét. Ekkor megkapjuk ennek a pixelnek a szomszédságában levő pixelértékek minimum és maximum értékét, majd megvizsgáljuk, hogy a középső pixel értéke beleesik-e ebbe a halmazba. Ha igen, akkor megtartja az értékét, ha nem, akkor ha az értéke kisebb volt, mint a halmaz szélső értéke, akkor megkapja ezt a minimumot, ha nagyobb volt, akkor a maximum értéket kapja.

3.4. Élvékonyítás

Miután a felületek is vonalakból épülnek fel, így célszerű vektoros vonalakat előállítanunk a vektorizálás során. Ehhez azonban vigyáznunk kell arra, hogy a végrehajtott procedúrák után is megmaradjon a térképen a topológia, és az objektumok végpontjai. Ugyanakkor – lentebb láthatjuk is – ahhoz, hogy a már vektorizált vonalak jól fussanak, le kell csökkentenünk a vastagságukat 1 pixelre. Ehhez élvékonyító eljárás szükséges.

Morfológiai élvékonyítás

Ezen eljárás bitképen dolgozik, egy kernelablak fut végig a képen, minden egyes pontban egy maszkot illeszt a képre, és annak elforgatási lehetőségeit vizsgálja. Maga a maszk (3.6. ábra) három féle értéket hordoz: háttér, objektum és figyelmen kívül hagyott. Akkor "áll jól" a maszk a képen, ha a háttér alatt `hamis`, az objektum alatt `igaz`, a figyelmen kívül hagyott alatt pedig tetszőleges érték szerepel.



3.6. ábra. A maszk vagy kernelablak értékei

A fentebbi képen a értékek között az 1 jelenti az objektumot, 0 a hátteret, míg a ? a figyelmen kívül hagyott. A maszk forgatásával összesen 8 állás lehetséges: ennek 45°-kal való forgatása. Minden pozícióban megvizsgáljuk, hogy a maszk illeszkedik-e, és ha illeszkedik, akkor az adott – kernelablak közepén levő – pixel értékét töröljük, vékonyítjuk a vonalat. A forgatási ciklus akkor áll le, ha már egyik elforgatási szög mellett sincs változás, egyik sem illeszkedik a lehetséges 8 irány közül. Ekkor az ablak a következő pixelre lép. Láthatjuk, hogy ez a folyamat igen időigényes.

3.5. Színek kiemelése

A vektorizálás egyik fontos előfeltétele a színek felismerése és csoportosítása. A bedigitalizált (tér)képek általában RGB színmodellt használnak. Az előzőekben (2.2. ábra) már láthattuk, hogy scannelés után az elméletileg homogén területek (pl. települések narancssárga kitöltése) nem lesznek homogének, az egyes pixelértékek között jókora eltérések is lehetnek. Hiába épül fel a térkép csak 8-10 színből, ha scannelés után ez az érték megsokszorozódik. Ezért célszerű az ilyen pixeleket egységesíteni, csoportosítani. Erre léteznek különféle módszerek. Mindenek előtt jó, ha tudjuk a vektorizálandó térkép nyomdai színeit. Ez hasznunkra válhat a csoportok létrehozásában. Jelentkező nehézség, hogy a térkép szerkesztésekor használt rétegek (pl. vízrajz, névrajz, *hipszometria*) színei a nyomtatás után összemosódnak.

Egyszerű színdetektálás

A színek csoportosításának legegyszerűbb módja, ha létrehozunk csoportokat (pl. szintvonal, főút, erdő), és hozzárendeljük a kívánt pixelértékeket (pl. szinvonalnál barna, erdőnél sötétzöld) RGB szerint. Ez függ a térkép típusától, színeitől is. Majd a pixelenként végigfutunk a képen, és megvizsgáljuk az adott pixel intenzitásértékét, és

megpróbáljuk elhelyezni a megfelelő csoportban. A nehézséget a már említett inhomogenitás és a képfájl mérete okozza, hiszen minél nagyobb a digitális állomány, annál tovább tart az eljárás. Az inhomogenitás hatását valamelyest csökkenthetjük, ha beállítunk egy színtolerancia-értéket az adott csoporthoz. Például a szintvonal barna színéhez nem egy konkrét RGB-számhármast rendelünk, hanem mindhárom konponenshez egy minimum és egy maximum értéket úgy, hogy az e halmazba eső pixelértékek még megfeleltethetők legyenek a szintvonalnak. Azonban vigyáznunk kell a halmaz méretével, mert ha túl nagy a tolerancia, akkor akár a nem szintvonalat reprezentáló pixelek is ebbe a csoportba eshetnek.

Erre a módszerre alapozva készítettem egy saját eljárást, ami a 5.3. fejezetben kerül bemutatásra.

Bejáráson alapú színdetektálás

Még hatékonyabb módszer az inhomogenitásra, ha a színdetektálást egy pixeltől kiindulva annak környezetében végezzük, egyre távolodva tőle. Ehhez célszerű egy interaktív eljárást készítenünk. Kiválasztunk a képen egy területet, például erdőt, majd az ebben a területben található pixelek közül kiválasztunk egyet. Ez a pixel biztosan erdőt reprezentál. A program ettől a ponttól kezdve végignézi annak környezetét, és ha hozzá hasonló pixelt talál, akkor azt egy csoportba sorolja vele.

3.6. Színek levétele a képről

Már tudjuk, hogy nagy területek vektorizálásához a legjobb, ha homogénné tudjuk tenni őket. Azonban számolnunk kell a felületben található más objektumokkal is. A felületekben legtöbbször névrajz szerepel, ami általában fekete színnel lett nyomtatva. A vektorizálás jobb eredménye érdekében célszerű ezeket a névrajzi elemeket (és természetesen egyéb zavaró objektumot) eltávolítani a felületről. Ennek egy egyszerű módja, ha megnézzük, milyen színű pixel van a fekete pixelhez legközelebb, majd kicseréljük a pixel értékét arra a színre. Ennek a módszernek viszont nagy hátránya, hogy nem biztos, hogy a kiválasztott "szomszéd" pixel a legmegfelelőbb, hiszen egy pixelnek akár 8 szomszédja is lehet. Ilyenkor a felület széle hullámossá válhat.

3.7. Textúrák kezelése

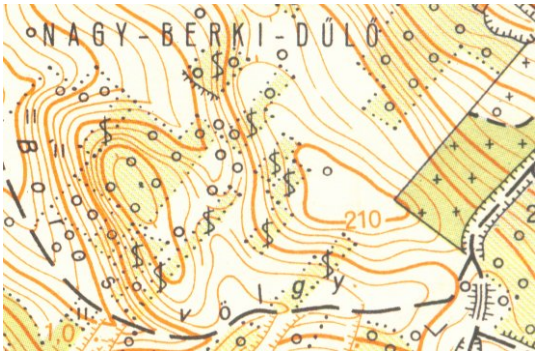
Eddig csak – elméletileg – homogén felületekről esett szó, azonban előfordulhat olyan is (pl. mocsaras területek), ahol a felület valamilyen textúrát kap, például egy egyszerű csíkozást. Ezt a felületet visszavezethetjük homogén felületté. Lássuk, hogyan.

Egyszerű textúra kezelés

Az ilyen felületeket több szűrésnek kell alávetnünk. Első lépésként sarokmegőrző medián szűrőt futtatunk rajtuk, majd a jobb hatás érdekében háromszor sarokmegőrző box szűrőt. Ez után a kisimított felület már vektorizálható lesz.

3.8. Mintaillesztések

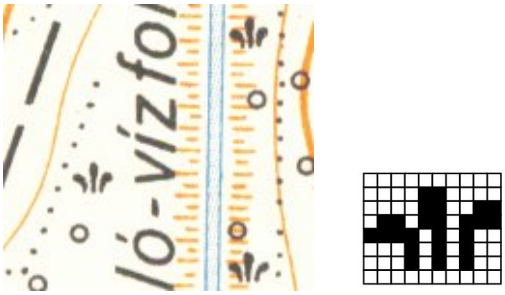
Ahogy az 1. fejezetben is szó esett róla, az újonnan készített térképek alapja legtöbbször valamilyen topográfiai térkép, célszerű tehát ezt vektorizálni. A topográfiai térkép a földfelszínt általában – persze ez méretarányfüggő – földhasználat szerint csoportosítja és jeleníti meg [3, 74–75. dia]. A csoportokat (rét, szőlő, gyümölcsös, szántó stb.) valamilyen felületi jel szerint különíti el (3.7. ábra). Ezeket is fel kell ismernie a programnak, és tudnia kell vektorizálni őket.



3.7. ábra. Felületi jelek a térképen

Egyszerű raszteres mintaillesztés

A modern kori térképek esetében a jelkulcs már egységes, így scannelés után az egyes minták (felületi jelek) jobbára egységesen digitalizálódtak. Ahhoz, hogy a program felismerje őket, készítenünk kell egy olyan maszkot, ami a jel alakját tartalmazza (3.8. ábra).



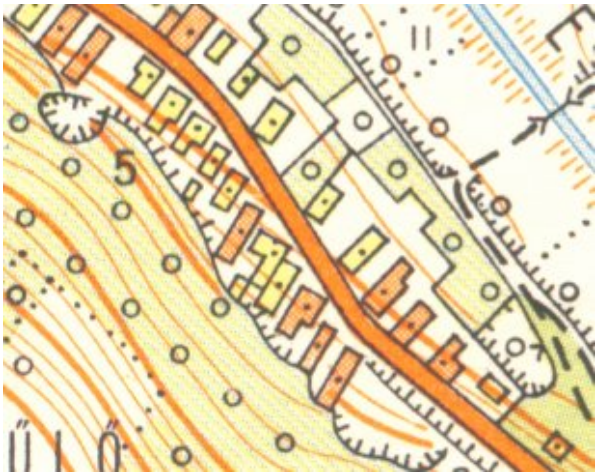
3.8. ábra. A térképi jel és a hozzá készített maszk

Ezek után végigfuttatjuk a maszkot – mint kernelablakot – a képen, és ahol a maszk megfelelően fedi az alatta levő jelet, oda egy új, vektorizálhatóság szempontjából jobb ábrát helyezhetünk. Nehézségek a régebbi térképeknél adódnak, ahol még nem volt egységes jelkulcs, nem biztos, hogy minden ugyanazon objektumot jelölő szimbólum egyformán néz ki.

Megjegyzés. Természetesen a fent említett eljárást egyéb objektumokra, akár a 1.3. ábrán bemutatott térképi jelekre is alkalmazhatjuk.

Eltérő irányú minták felismerése

Nem esett szó az olyan jelkulcsi elemekről, melyek mérete, iránya nem egységes, eltérő lehet. Ilyen például megfelelő méretarányban az épületek alakja. A 3.9. ábrán ezen felül egy újabb problémát látunk, az eltérő kitöltést. Egyes térképeken az épületek kitöltése a magasságuktól függ, máshol a tűzállóságuk szerint csoportosítják őket.



3.9. ábra. Épületek jelölése a térképen

Épületek kiemelésére és vektorizálására érdemes úgy előkészítenünk a képet, hogy minden más zavaró tényezőt eltüntetünk. A következő lépésben a megfelelő színre állítva a kernelt végigfuttatjuk a képen, és megkeressük a hasonló színű pixeleket, majd találat után megállapítjuk, az adott színű pixelekből álló halmaz hány pixelt tartalmaz. Már ezzel is szűrhetjük az eredményt, hiszen az épületek legfeljebb egy bizonyos darabszámú pixelből épülhetnek fel a képen, így az ennél több pixelből álló halmazt kitöröljük. A megmarad halmazokra a forgó érintők módszerével meghatározzuk a legkisebb befoglaló téglalapot, ami egyben meg is adja az épület helyét és irányultságát is.

4. fejezet

Vektorizálás

4.1. Automatikus vektorizálás

Miután előkészítettük a képet, a következő lépés maga a vektorizálás. Ez két lépésben történhet: először végrehajtjuk az úgynevezett nyersvektorizálási folyamatot, ahol előállítjuk a felületeket. A vonalkövetés-funkció nehézsége okán poligon-növesztést alkalmazunk, ennek segítségével alakítjuk ki a felületeket. A módszer lényege, hogy a program a már előzőleg csoportosított pixeleket összekapcsolja felületté. Így az eltérő intenzitású pixelek (erdő, kert, beépített terület stb.) külön felületeket alkotnak. Egy poligon addig növekszik, míg "be nem kebelezte" a szomszédságában levő összes azonos intenzitású pixelt.

Ennek a folyamatnak az eredményeképpen egy hézag- és átfedésmentes lefedettségű poligon struktúrát állítottunk elő. Ez a *topológiailag helyes tárolás* miatt fontos.

Azonban az eredményül kapott poligonmezőben – ahogy az eredeti térképen is – benne találhatóak a különféle megírások, jelek is. Ezek eltávolítása már nagyobb feladat. Ez lenne a következő lépés.

Célszerű a különböző típusú objektumoknak (pont, vonal, felület) külön réteget elkészíteni, és azokat egyenként vektorizálni, majd a folyamat végeztével újra egy képpé összeilleszteni a rétegeket a megfelelő sorrendben. Az összetettebb jeleket maszkok segítségével különíthetjük el és vektorizálhatjuk.

A létrehozott vektoros állományt már csak el kell mentenünk egy olyan formátumban, amit a legtöbb szoftver felismer.

4.2. Fellépő hibák

Természetesen a gyakorlatban nem ilyen egyszerű a konverzió. A vektorizálás során különféle problémák lépnek fel, amit csak külső (emberi) segítséggel tud megoldani a program. Ilyen hibák, nehézségek lehetnek például:

- Ha egy felületen belül egy másik felület található. Például egy szántó terület

közepén egy kisebb erdő. Egy lehetőség, ha az erdőt egy másik rétegre helyezzük, és a szántó "alatta" folytatódik. Azonban, ha egy olyan szoftvernél használjuk a kapott vektorizált állományt, ami a felületek nagyságával (mekkora a területe?) is foglalkozik, akkor ez egy rossz megoldás.

- A szintvonalérték-számoknál a szintvonal futása megszakad, a program csak külső segítséggel tudja folytatni a megfelelő vonalat, hiszen a vonal folytatása messzebb van, mint a tőle kisebb vagy nagyobb értékű szintvonalak, így megeshet, hogy átugrik egy másikra (piros nyíl a 4.1. ábrán).

- Karakterek, szövegek felismerése. Általában a felületek neveit a térképen szórtan, ívelten tüntetik fel. A térképolvasó tudja, melyik karakter melyik névhez tartozik. A program viszont csak betűnként tudja vektorizálni a szöveget, nem létesít közöttük kapcsolatot. Ráadásul egy ívelt névrajzi elemet sokkal nehezebb felismertetni a programmal.

- Vonalak vékonyításánál egy több pixel vastagságú vonal esetén nem biztos, hogy a program a vonal közepét hagyja meg.

- Azonos színű vonalak találkozásakor a programnak 2, esetleg több irány között kellene választania.

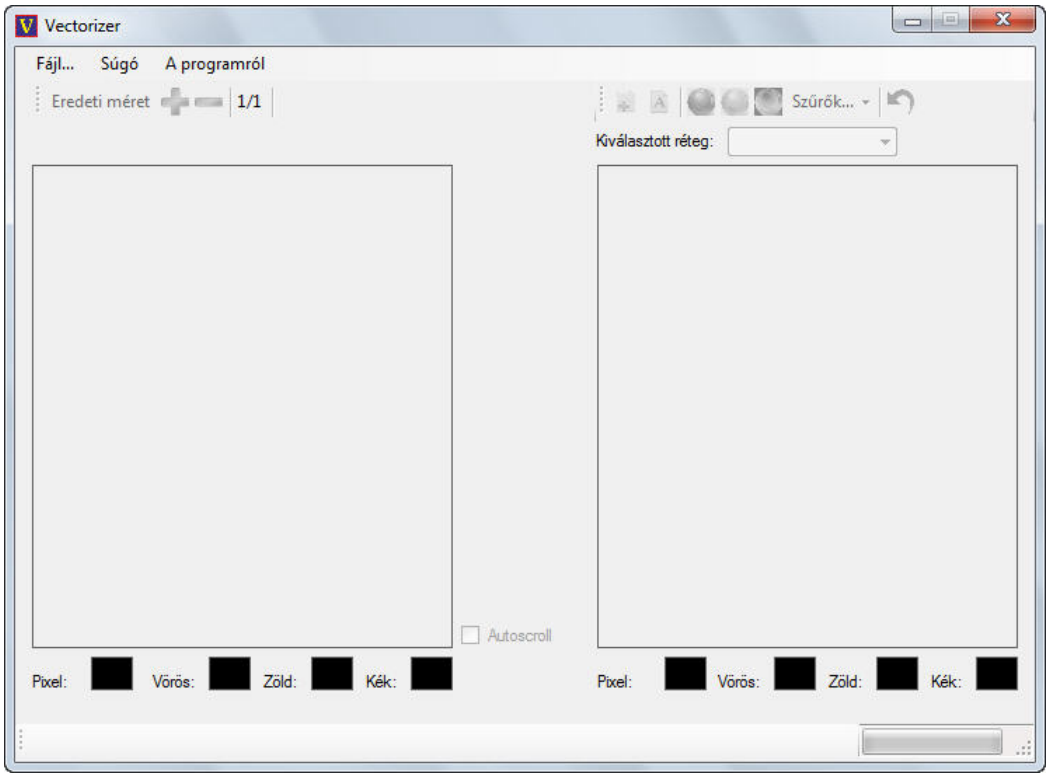
4.1. ábra. A program rosszul ismeri fel a vonal futásának folytatását

A felsoroltakon kívül számos más hiba is előfordulhat vektorizálás közben, hiszen minden térkép más. A megoldás egyenlőre a félautomatikus vektorizálás, ami hozzáértő kezekkel hatékonyabban állíthat elő vektoros térképet, mintha mindent csak hagyományosan, kézzel rajzolnánk meg.

5. fejezet

Saját program bemutatása

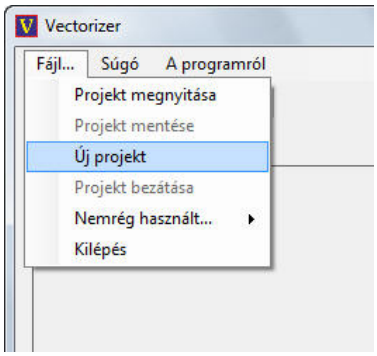
A diplomamunkámhoz készítettem egy programot, mely alapvető képmanipuláló funkciókkal bír. Ezen funkciók alkalmazásával a használt alaptérkép tetszés szerint átalakítható, ezáltal megkönnyítve a felhasználását a rajzolásban. A program az ingyenes Microsoft Visual Basic 2010 Express-ben íródott, a programkód a mellékletben megtalálható. Bár a projekt a Vectorizer nevet kapta, a program nem képes konkrét vektorizáló folyamatokra, csak a módosítani kívánt térképet készíthetjük elő rá. Maga a vektorizáló folyamat leprogramozása véleményem szerint túlmutat a diplomamunka terén végzendő feladat határain. Tehát a programot a vektorizálást segítő alkalmazások kategóriájába sorolnám.



5.1. ábra. A főképernyő

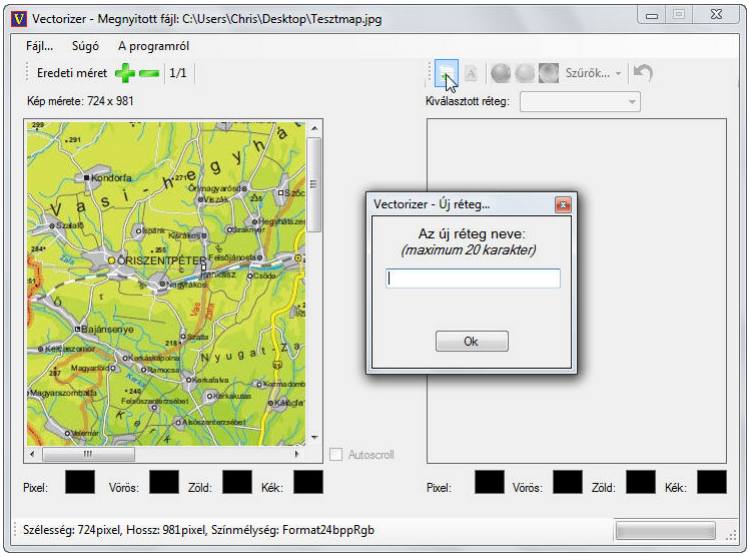
5.1. Alapok

A program indítása után a főképernyő (5.1. ábra) töltődik be. A program használatához először egy új projektet kell létrehoznunk. Ehhez kattintsunk a "Fájl..." gombra, majd a lenyíló menüben az "Új projekt" gombra (5.2. ábra).



5.2. ábra. Új projekt létrehozása

Kiválaszthatjuk a manipulálni kívánt képfájlt a következő formátumokból[1]: *bmp, jpg, tif, png.* Ezt követően a program betölti a képet a bal keretbe. Ahhoz, hogy manipulálni tudjuk, létre kell hoznunk egy új réteget. Ezt a jobb oldali keret feletti menüsorban az "Új réteg" gomb segítségével tehetjük meg. Meg kell adnunk a réteg nevét (5.3. ábra), majd a jobb keretbe betöltődik a kép. Lehetőségünk van több réteget is létrehozni. Ennek felső korlátja 10 darab. A program mindig a legújabbat mutatja, de természetesen kiválaszthatjuk a számunkra szükséges réteget. Ehhez kattintsunk a rétegek nevét tartalmazó lenyíló menüre.



5.3. ábra. Új réteg létrehozása

[1]Vegyük figyelembe, hogy a program legfeljebb 24 bites képeket kezel.

5.2. Egyszerűbb funkciók

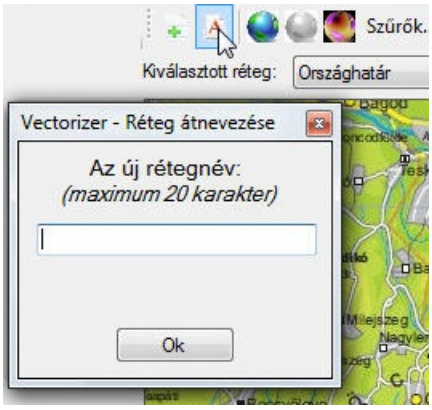
Nézzük a menük egyes elemeinek tulajdonságát. A bal oldali menüvel (5.4. ábra) a betöltött képet tudjuk nagyítani (mintegy rázoomolni), vagy épp kicsinyíteni. A bal oldali kép nagyítása hatással van a jobb oldali képre is. Emellett ha az egeret a képen mozgatjuk, kiírja annak aktuális képkoordinátáit.

5.4. ábra. Bal menüsor

A jobb oldali menüsor (5.5. ábra) többnyire a manipulálni kívánt rétegekre hat. Az első gomb már említve volt, egy új réteg létrehozására szolgál.

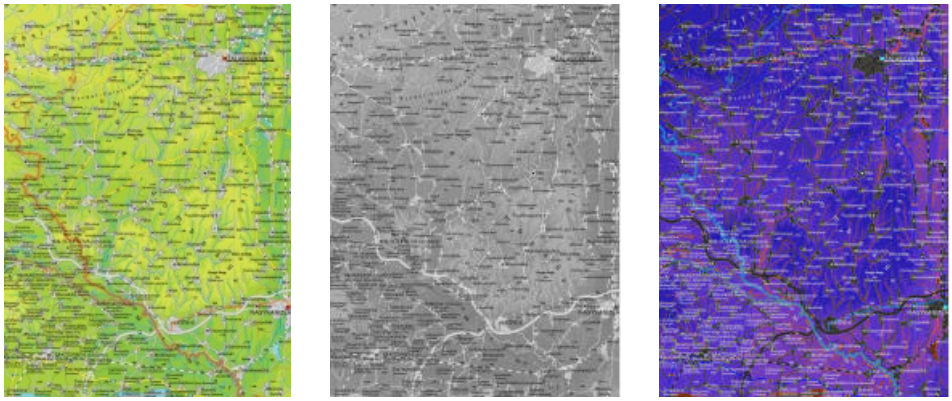
5.5. ábra. Jobb menüsor

Mellette található a "Réteg átnevezése" gomb, amivel a kiválasztott réteg nevét tudjuk megváltoztatni. Rákattintva megjelenik egy ablak (5.6. ábra), ahova beírhatjuk a réteg új nevét. Ez a funkció a következő gombbal együtt nyer értelmet: elnevezünk egy réteget olyan néven, amit majd jelképezni fog a tartalma (pl. a szintvonal rétegen csak a szintvonalak fognak látszani). Később rájövünk, hogy nekünk nem kell ez a réteg, ezért rákattintunk a "Normál szín" gombra, ami visszatölti az eredeti, még manipulálatlan képet a rétegbe. Ez után átnevezzük a réteget szintvonalról valami másra, amit az adott réteg tartalmazni fog majd.



5.6. ábra. Réteg átnevezése

A következő gombok a réteg képét változtatják meg (5.7. ábra). Az elsővel visszakapjuk az eredeti képet, a mellette levő "Szürkeárnyalatos kép" gombra kattintva a kép szürkeárnyalatos lesz. Az "Invertált szín" után a kép színei a *kiegészítő színekre* cserélődnek.



5.7. ábra. Normál, szürkeárnyalatos és invertált kép

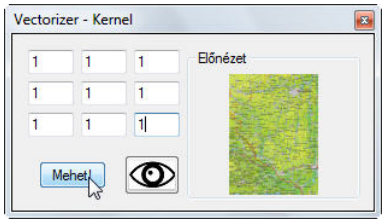
5.3. Összetettebb funkciók

A "Szűrők..." gomb mögött összetettebb manipuláló funkciók találhatóak (5.8. ábra). Az első a "Simító", célja a képen található kisebb foltok, "piszkok" eltüntetése. Alatta a "Laplace-szűrő" egy *éldetektor*, a képen található hírtelen színváltozásokat keresi, és emeli ki. A "Saját" gomb alatt a felhasználó szabadon választhat kernelt, ami majd végigfut a képen. Mérete szigorúan csak 3x3-as mátrix lehet (5.9. ábra).

5.8. ábra. Normál, simított és éldetektált kép

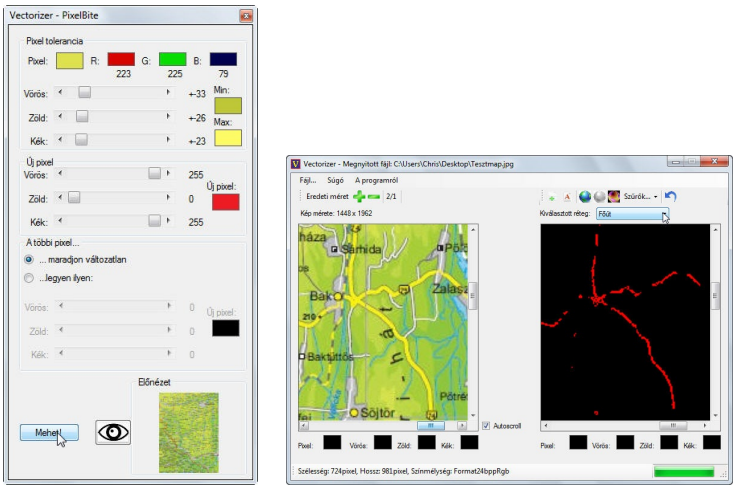
Ehhez a funkcióhoz már tartozik "Előnézet" is, melynek célja a beállított kernel hatékonyságának bemutatása: ennek a folyamatnak a lefutásához már több idő szükséges, és ha a felhasználó rosszul állította be a kernelt, nem a várt eredményt kapta, akkor várhat a következő lefutás végig. Az előnézettel azonban a beállított kernel helyessége gyorsan leellenőrizhető, hiszen a kiválasztott réteg képének egy tömörített változatán fut végig a kernel, amihez kevesebb idő szükséges.

A "Pixel kiemelése" funkció (5.10. ábra) már igen csak interaktív. Lényege, hogy a kiválaszott réteg képén bal egérgombbal kijelölünk egy színt, majd (ismerve az esetleges képi tömörítéseket, nem biztos, hogy a képen kiválasztott kék máshol is ugyan olyan intenzitású) megadunk egy tolerancia értéket, ami még elfogadható. Az így kiválasztott színnek megadhatjuk az új RGB színét, amivé átszíneződik a funkció lefutása után. Le-



5.9. ábra. Saját kernel

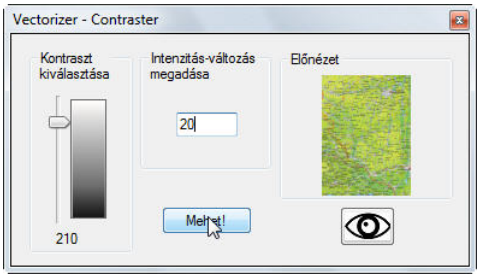
hetőségünk van beállítani, mi legyen a többi pixellel, melyek nem estek bele a megadott szín-toleranciába: vagy változatlanul maradnak, vagy átszínezhetjük őket. A bal oldali



5.10. ábra. Pixel-kiemelés eredménye

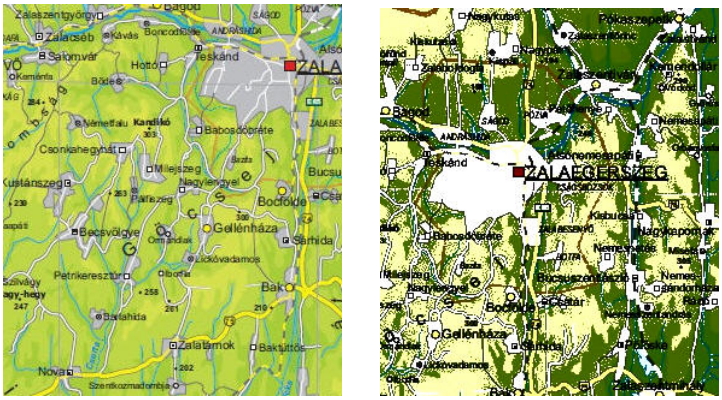
keret (eredeti) képén látható főút sárga színét választottuk ki, majd színeztük át pirosra, a többi pixel színét pedig feketére változtattuk.

Egy újabb összetettebb funkció a "Kontraszt" (5.11. ábra) tartalma. Itt kiválaszthatunk egy szürkeárnyalatos skálán egy adott intenzitás-értéket, majd a program végigfut a kép pixelein, és attól függően, mekkora értéket állítottunk be változásnak, a program a kiválasztott intenzitás feletti értékű pixelekhez hozzáadja, illetve az az alatti értékű pixelekből kivonja a változás értékét (5.12. ábra).



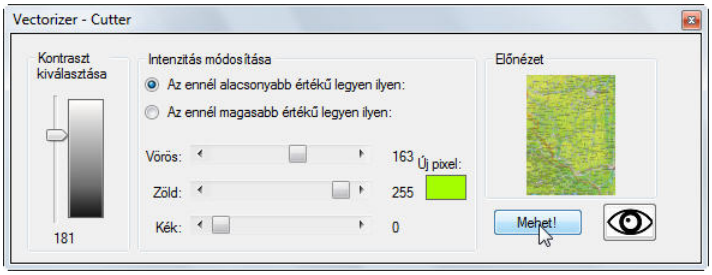
5.11. ábra. A kontrasztálás...

A "Vágó" funkció (5.13. ábra) hasonló elven működik. A felhasználó itt is kiválasztja a neki megfelelő intenzitást, majd - attól függően, mit választott - a program az ez alatti, illetve feletti értékű pixeleket átszínezi a beállított színre (5.14. ábra). Ha a funkciók

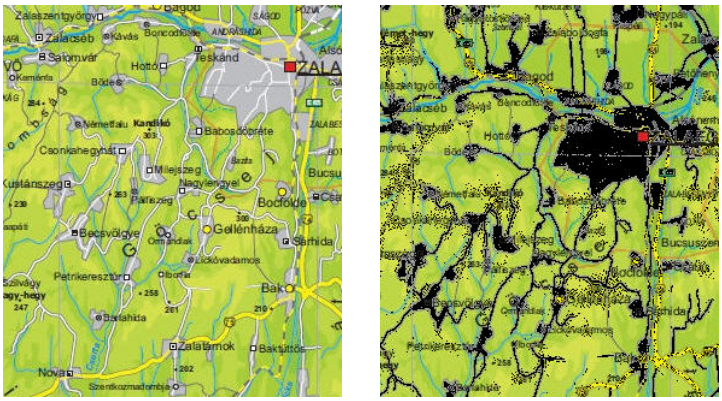


5.12. ábra. ...és eredménye: normál, és kontrasztált kép

alkalmazása nem jó eredményhez vezetne, akkor a menü jobb szélső gombjával ("Vissza") egy lépést visszaléphetünk, azaz az utolsó változtatást törölhetjük.



5.13. ábra. A vágó beállítása...

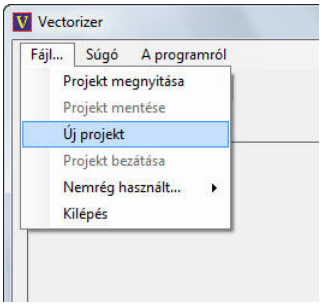


5.14. ábra. ...és alkalmazása: normál, és felül vágott kép

Megjegyzés. A kiválasztott réteg képére jobb gombbal kattintva kimenthetjük az aktuális képet különböző formátumokban.

5.4. Projekt mentése/betöltése

Hogy ne kelljen egy képen újra és újra elvégeznünk ugyanazokat a folyamatokat, valahányszor elindítjuk a programot, lehetőségünk van menteni a projektet. Ezt a "Fájl..." menüben tehetjük meg (5.15. ábra). A program a megadott néven elmenti a fájlt, és létrehoz egy ilyen nevű mappát is, ahova egyesével kimenti a meglévő rétegeket, azok tulajdonságaival együtt. Ezek után a "Projekt megnyitása" gombbal tölthetjük be a programba elmentett projektünket. A gyors elérés érdekében helyet kapott a programban egy "Nemrég használt" nevű menüpont is. Ennek segítségével az utolsó két használt projektet keresés nélkül, egy kattintással elérhetjük.



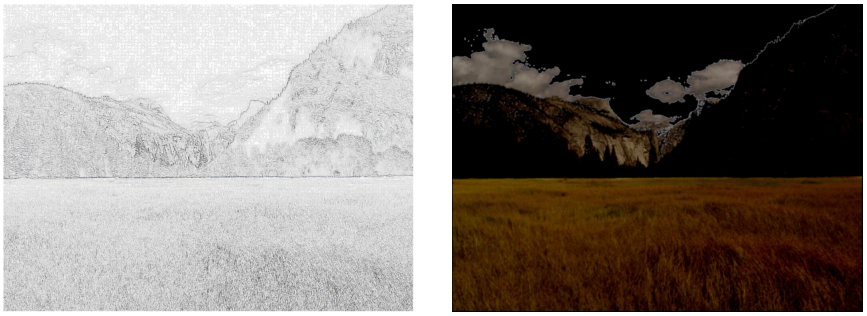
5.15. ábra. A Fájl-almenü tartalma

5.5. Egyéb felhasználási módok

Természetesen a program nem csak térképeket képes manipulálni. Lehetőségünk van bármilyen fénykép, sőt, űrfotó átalakítására is. A fentebb említett műveletek egymás után elvégezhetők, így például egy képet először invertálhatunk, majd ezt követően szürkeárnyalatossá tehetjük. A következő, 5.16. ábrából[2] kiindulva láthatjuk a soron követező képeken, mennyi lehetőség rejlik a programban.

5.16. ábra. Az eredeti kép

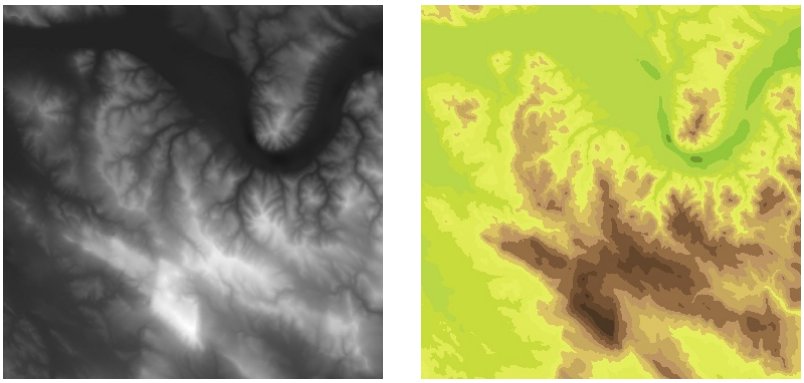
[2]Forrás: `http://raydesigned.com/wp-content/uploads/2011/02/Yosemite-Valley-1024x768.jpg`



5.17. ábra. Lehetőségek I.

5.18. ábra. Lehetőségek II.

Űrfotók esetében a különbségek kiemelésére van lehetőségünk, de – ahogy az alábbi, 5.19. ábra is mutatja – egy szürkeárnyalatos képet kiszínezhetünk hipszometrikusan is.



5.19. ábra. Az eredeti és a hipszometrikus űrfotó

5.6. További lehetőségek a program fejlesztésében

Ahogy egy térkép, úgy egy program sincs kész, csak be van fejezve a szerkesztése/készítése. Amint az ember megold egy problémát, vagy megvalósít egy ötletet, úgy jönnek elő az újabbnál újabb gondok és elképzelések. Bár a program vektorizáló alkalmazásnak indult – ahogy a neve is mutatja –, időigényessége és összetettsége miatt ez a funkció végül kimaradt. De természetesen ez nem azt jelenti, hogy lehetetlen lenne megvalósítani.

Egy másik fontos lehetőség a mentés funkció fejlesztése: a program jelen helyzetében minden egyes réteget úgy ment ki, hogy a réteg képét egy képfájlba tölti, és a réteg nevén

elmenti a megadott mappába, majd betöltéskor ezen képfájlokat teszi be a megfelelő rétegnév alá, és amikor kiválasztjuk az adott réteget, megjeleníti a jobb oldali ablakban. Ha nagy képfájlokkal dolgozunk, és sok rétegünk van, akkor nagyon hely és időigényessé tud válni a program, hiszen mentéskor minden réteg annyi helyet foglal el a merevlemezen, amekkora az eredeti kép mérete volt. A nagy fájlok betöltése pedig tovább tart. A mentés hely- és időigényét úgy lehetne lerövidíteni, hogy a program nem magát a rétegek képét menti ki, hanem csak a rajta elvégzett folyamatokat, azok sorrendjét és az eredeti képet. Hiszen minek minden egyes réteg képét kimenteni, ha mindegyik ugyanabból az alapból indult: az eredeti képből. Betöltéskor minden réteg megkapná az eredeti képet, majd lefutnának rajtuk a megfelelő funkciók a megfelelő sorrendben.

Ha már vektorizálásról van szó, lehetőség lenne olyan vektoros formátumokba kimenteni a rétegeket, amiket a legtöbb szoftver ismer (1.2. fejezet).

A szűrők futásidejét is lehetne csökkenteni. Ugyan a Visual Studio ajánlott módszeréhez képest jelentősen felgyorsult a szűrők lefutása, további lehetőségek is számba vehetők: a lefutás során a legnagyobb időveszteség a feltételes ciklusok miatt történik. Minél több feltétel szerepel egy cikluson belül, annál lassabb lesz a végrehajtódás. A kernel nagysága is befolyásolja a futási időt: egyenes arányban nő a kernel méretével. Ezen is lehet javítani, ha a kernel mátrixát felbontjuk sor- és oszlopvektorokra, majd először az egyiken, majd a másikon hajtódik végre a cilus.

Egy másik – még megoldatlan – hibája a programnak, hogy csak 24 bites képeket tud kezelni. Ezt orvosolhatjuk úgy is, hogy az ennél alacsonyabb bitsűrűségű képeket átkonvertáljuk 24 bitessé, majd a végrehajtott módosítások után visszakonvertáljuk őket az eredeti bitsűrűségűvé. Ez a módszer csak 24, vagy annál alcsonyabb bitsűrűségű képeknél működik, hiszen 32 bit visszakonvertálása során már adatvesztést szenved a kép.

5.7. Felhasznált oldalak

Kép színmélységének megváltoztatása
`http://msdn.microsoft.com/en-us/library/ms141944.aspx`
A radio-button működése
`http://www.vbtutor.net/vb2008/vb2008_lesson18.html`
Kattintás egérrel
`http://www.vbforums.com/showthread.php?t=504462`
Egérgomb nyomvatartása
`http://www.homeandlearn.co.uk/net/nets10p2.html`
Kilépés ciklusból
`http://forums.asp.net/t/271367.aspx/1`
Fájl vagy mappa törlése

`http://bytes.com/topic/visual-basic-net/answers/336752-delete-all-files-folder`

Mappa létrehozása

`http://stackoverflow.com/questions/85996/how-do-i-create-a-folder-in-vb-if-it-doesnt-exist`

Kép kimentése fájlba

`http://stackoverflow.com/questions/5813633/a-generic-error-occurs-at-gdi-at-bitmap-save-after-using-savefiledialog`

Kép átalakítása bájttömbbé

`http://www.vbforums.com/showthread.php?t=358917`

Messagebox működése

`http://www.thevbprogrammer.com/VBNET_08/08-00A-Msgbox.htm`

Bájttömb elrendezése

`http://www.bobpowell.net/lockingbits.htm`

Megjegyzés. A fentebb felsorolt honlapok a diplomamunka elkészültéig elérhetőek voltak.

6. fejezet

Fogalomtár

- **Attribútum**: Tulajdonságot meghatározó jelölés. Egy vonal attribútuma például a vastagsága.
- **Bézier-görbe**: Széles körben alkalmazzák a számítógépes grafikában görbe vonalak modellezésére. A görbék interaktív módszerrel is könnyen igazíthatók kontroll pontjaik mozgatásával. Általában másod- és harmadfokú Bézier-görbéket használnak, a magasabb fokszámú görbék előállítása túlságosan sok számítást igényel.
- **Bmp, jpg, tif, png**: Képformátumok, részletes leírásuk az 1.1. fejezetben található.
- **CCD érzékelő**: Charged-coupled device - töltéscsatolt elem. Olyan eszköz, amely az egyes, sorba vagy négyzetes mátrixba rendezett elemeken elektromos töltéscsomagok elvezetésével képi információ feldolgozására képes.
- **Dpi**: Dot/inch, a digitális felbontóképesség mértékegysége, 60 dpi azt jelenti, hogy egy inch-en (2,54 cm) 60 darab képpont (pötty) fér el. Metrikus egysége a pont/cm vagy a vonal/cm.
- **Éldetektor**: Az élek a képnek azon helyei, ahol az intenzitás megváltozása a legnagyobb. Az éldetektáló szűrők a kép ilyen részeit keresik.
- **Felbontás**: Lásd dpi.
- **Generalizálás**: A térképtartalom kiválogatása, egyszerűsítése, összefogása és fogalmi átalakítása az újonnan létrehozandó térkép méretarányának vagy céljának megfelelően.
- **Hipszometria**: A domborzat magasságának csoportosítása színekkel. Az azonos magassági övbe eső területek azonos színűek. Ha az övek száma és ezzel a színárnyalatok száma is igen nagy, akkor az övhatárok nem is látszanak, egy folyamatos átmenetet lehet biztosítani. Ha az övek és ezzel a színfokozatok száma is korlátozott,

akkor öves, vagy hipszometrikus ábrázolásmódról beszélünk. A színfokozatok kialakításánál két gyakorlat alakult ki, melyeket együttesen alkalmaznak. Az egyik szerint a magasabb területek sötétebb színűek, mint az alacsonyabban fekvő területek, a tengerek, vizek esetében pedig fordítva: minél mélyebbek, annál sötétebbek. A másik a színezésre vonatkozik: a sík vidékeken a zöld szín, a hegyek ábrázolására a barna szín, a vizek ábrázolására a kék szín árnyalatait használják a leggyakrabban.

- **Kernel**: Esetünkben egy $n \times n$-es mártix, mely a raszteres képek manipulálásakor végigszalad a képen, és – amennyiben az elemei értékekkel vannak felruházva, úgy azt is figyelembe véve – módosítja a pixelek értékét.

- **Kiegészítő színek**: Egymást fehérre kiegészítő, a színkörben egymással szemben álló színek: kék-sárga, zöld-bíbor, kékeszöld-vörös.

- **Méretarány**: Megmutatja, hogy a térképen egységnyi hosszúság (rendszerint 1 cm) a valóságban hány centiméternek felel meg. Pontosabban: a térkép hossztartó vonalain mért távolságnak és a valóságban vízszintesre redukált hossznak az aránya. Például az 1 : 10 000 méretarányú térkép 1 centimétere a valóságban 10 000 centiméternek, azaz 100 méternek felel meg.

- **Pixel**: A digitális képfeldolgozásban a képpont (angolul pixel) egy pont egy raszteres (vagy rasztergrafikus) képen. Általános esetben ezek egy kétdimenziós négyzetrács mentén helyezkednek el, és mint négyzetlapok vagy pontok jelennek meg. A képpont a legkisebb szerkeszthető alkotóeleme a képnek.

- **Rajzi struktúra**: Réteg- vagy layerszerkezet. A modern rajzprogramok lehetőséget adnak különböző rétegek (szintek) használatára. Általában térképek szerkesztésénél külön rétegen találjuk a vízrajzi elemeket, a domborzatot reprezentáló szintvonalakat vagy rétegszínezést, a névrajzi elemeket stb. Ez megkönnyíti a szerkesztést.

- **RGB**: Az angol Red (vörös), Green (zöld), Blue (kék) kezdőbetűk rövidítése.

- **Színárnyalat**[1, 5. dia]: Teleszínből raszterráccsal bontott szín, ugyanannak a teleszínnek az árnyalata (pl. sárga–halványsárga). Egy színárnyalatot nem csak egy színből, hanem több szín árnyalatainak egymásra nyomásával is elő lehet állítani.

- **Színmélység**: Minél magasabb a színmélységet jelölő szám , annál több színárnyalatot (pl. 1 bit – 2 szín, 4 bit – 16 szín, 8 bit – 256 szín, 16 bit – 65 536 szín, 24 bit – 16 777 216 szín) képes megjeleníteni az adott készülék (pl. monitor).

- **Topológiailag helyes tárolás**: Vektoros adatok helyes tárolásakor elmentjük az objektumok (pontok, vonalak, felületek) egymással való kapcsolatát is. Például szomszédsági viszonyok, kapcsolatok, csomópontok, bennfoglalás, felépítés stb.

- **Valós színek**: Angolul true color. Olyan színmélység, amely több különböző szín ábrázolására ad lehetőséget, mint amennyit az emberi szem meg tud különböztetni. True color a számítógépes szakmában azon VGA-grafikus kártyák megnevezése, amelyek a 24 bites színmélység segítségével 16,7 millió különböző, a képernyőn egyidőben megjeleníthető színt képesek előállítani.

Irodalomjegyzék

[1] Faragó Imre (2007): A térképi generalizálás, a domborzatábrázolás

http://mercator.elte.hu/~farago/TSZT1-04%20gyak%20Generalizalas-Domborzatabrazolas%202007.pps

[2] Faragó Imre (2007): A térkép, tömegtérképek, a térképkészítés folyamata

http://mercator.elte.hu/~farago/TSZT1-01%20gyak%20T%C3%A9rk%C3%A9p-T%C3%B6megt%C3%A9rk%C3%A9p-Folyamat%202007.pps Utolsó elérés: 2012. június 3.

[3] Faragó Imre (2007): Síkrajz III.: A közlekedés és a fedettség ábrázolása

http://mercator.elte.hu/~farago/TSZT1-07%20gyak%20Kozlekedes%202007.pps Utolsó elérés: 2012. június 3.

[4] Zentai László (2003): Output orientált digitális kartográfia, doktori értekezés, Budapest.

http://lazarus.elte.hu/hun/dolgozo/zentail/dsc/zl-nagydoktori3.pdf Utolsó elérés: 2012. június 3.

[5] Elek István – Dezső Balázs – Máriás Zsigmond (2007): IRIS, Automatikus raszter-vektor konverziós rendszer fejlesztése (IKKK 5. kutatási főirány, beszámoló jelentés), Budapest, ELTE Informatikai Kar, IKKK.

http://mapw.elte.hu/elek/pdf/iris.pdf Utolsó elérés: 2012. június 3.

[6] Lapolvasók tesztje 3/1, a szkennerek működése: http://prohardver.hu/teszt/lapolvasok_tesztje_3_1/a_szkennerek_mukodese.html. Utolsó elérés: 2012. június 3.

Köszönetnyilvánítás

Köszönettel tartozom Szarvas Andrásnak, aki felkeltette érdeklődésemet a téma iránt; Elek Istvánnak, mint témavezetőmnek a sok segítségért, bíztatásért, pozitív kritikákért; testvéremnek, Nemes Gergőnek a dolgozat elkészítésében nyújtott segítségéért, és köszönöm Bérces Ádámnak a tőle kapott térképet, melyet felhasználtam a dolgozatomban.

Melléklet

FŐKÉPERNYŐ

```
Imports System.String
Public Class Form1
    Dim fil As New Filters
    Public undopic, trick(10), trickalap, casualpic, alap, bmp(10) As Bitmap
    Public foldern, lname, fname, rawname, layername(10) As String
    Dim volt, selected, pic2loaded, show1, show2, alapvan As Boolean
    Public sd, numb As Integer
    Dim zooml, zoomr, zoom As Double


    'KÉP IMPORTÁLÁSA
    Private Sub Import_Click(sender As System.Object, e As System.EventArgs) Handles Import.Click
        OpenFileDialog1.Filter = "Képfájlok|*.bmp;*.jpg;*.bmp;*.tif;*.png"
        If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            fname = OpenFileDialog1.FileName
            zoomr = 1
            zooml = 1
            ToolStripZoom.Text = zooml & "/" & zoomr
            Normal.Enabled = True
            ZoomIn.Enabled = True
            ZoomOut.Enabled = True
            Import.Enabled = False
            Closes.Enabled = True
            Recent.Enabled = False
            Save.Enabled = True
            alap = Bitmap.FromFile(OpenFileDialog1.FileName)
            PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
            PictureBox1.Image = alap
            PictureBox1.Width = alap.Width
            PictureBox1.Height = alap.Height
            PictureBox1.Visible = True
            PictureBox1.Refresh()
            ToolStripStatusLabel1.Text = "Szélesség: " & alap.Width & "pixel, Hossz: " & alap.Height
            & "pixel, Színmélység: " & alap.PixelFormat.ToString
            Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
            Me.Text = "Vectorizer - Megnyitott fájl: " + OpenFileDialog1.FileName
        End If

    End Sub

    'BETÖLTÉS
    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        Dim Recenter As New Ini(Application.StartupPath & "\Recent.rec")
        'új recent előre
        ToolStripRecentN.Text = Recenter.GetString("Recent", "Newer", "")
```

```
    If ToolStripRecentN.Text = "" Then
        ToolStripRecentN.Visible = False
    End If
    'régi recent hátra
    ToolStripRecentO.Text = Recenter.GetString("Recent", "Older", "")
    If ToolStripRecentO.Text = "" Then
        ToolStripRecentO.Visible = False
    End If
    pic2loaded = False
    If ComboBox1.Items.Count > 0 Then
        ComboBox1.SelectedIndex = 0
    Else
        ComboBox1.SelectedIndex = -1
    End If
    numb = 0
End Sub

'EGÉR MOZGATÁSA
Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.
MouseEventArgs) Handles PictureBox1.MouseMove
    If PictureBox1.Visible = True Then
        ToolStripMouse.Text = "Koordináták: " & e.X & ", " & e.Y
        show1 = True
    Else
        show1 = False
    End If
    'Színkiszedés
    Dim col As Color
    If e.X < PictureBox1.Width And e.Y < PictureBox1.Height Then
        Try
            If zoomr = 1 Then
                col = alap.GetPixel(e.X / zooml, e.Y / zooml)
            Else
                col = alap.GetPixel(e.X * zoomr, e.Y * zoomr)
            End If
        Catch
        End Try
    End If
    If show1 = True Then
        Pixel1.BackColor = col
        Dim colR As Byte = col.R
        Red.Text = col.R.ToString
        Dim colG As Byte = col.G
        Green.Text = col.G.ToString
        Dim colB As Byte = col.B
        Blue.Text = col.B.ToString
        PictureBoxRed.BackColor = Color.FromArgb(colR, 0, 0)
        PictureBoxGreen.BackColor = Color.FromArgb(0, colG, 0)
        PictureBoxBlue.BackColor = Color.FromArgb(0, 0, colB)
    Else
        PictureBoxRed.BackColor = Color.FromArgb(0, 0, 0)
        PictureBoxGreen.BackColor = Color.FromArgb(0, 0, 0)
        PictureBoxBlue.BackColor = Color.FromArgb(0, 0, 0)
        Blue.Text = Green.Text = Red.Text = ""
        Pixel1.BackColor = Color.FromArgb(0, 0, 0)
    End If
End Sub

'EGÉR MOZGATÁSA 2
Private Sub PictureBox2_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.
```

```
MouseEventArgs) Handles PictureBox2.MouseMove
    If PictureBox2.Visible = True Then
        ToolStripMouse.Text = "Koordináták: " & e.X & ", " & e.Y
        show2 = True
    Else
        show2 = False
    End If
    'Színkiszedés
    Dim col As Color
    If e.X < PictureBox2.Width And e.Y < PictureBox2.Height Then
        Try
            If zoomr = 1 Then
                col = bmp(sd).GetPixel(e.X / zooml, e.Y / zooml)
            Else
                col = bmp(sd).GetPixel(e.X * zoomr, e.Y * zoomr)
            End If
        Catch
        End Try
    End If
    If show2 = True Then
        Pixel2.BackColor = col
        Dim colR As Byte = col.R
        Red2.Text = col.R.ToString
        Dim colG As Byte = col.G
        Green2.Text = col.G.ToString
        Dim colB As Byte = col.B
        Blue2.Text = col.B.ToString
        PictureBoxRed2.BackColor = Color.FromArgb(colR, 0, 0)
        PictureBoxGreen2.BackColor = Color.FromArgb(0, colG, 0)
        PictureBoxBlue2.BackColor = Color.FromArgb(0, 0, colB)
    Else
        PictureBoxRed2.BackColor = Color.FromArgb(0, 0, 0)
        PictureBoxGreen2.BackColor = Color.FromArgb(0, 0, 0)
        PictureBoxBlue2.BackColor = Color.FromArgb(0, 0, 0)
        Blue2.Text = Green2.Text = Red2.Text = ""
        Pixel2.BackColor = Color.FromArgb(0, 0, 0)
    End If
End Sub

'EGÉR MOZGATÁSA KÉPEN KÍVÜL
Private Sub Form1_MouseMove(sender As Object, e As System.Windows.Forms.MouseEventArgs)
 Handles Me.MouseMove
    ToolStripMouse.Text = ""
    PictureBoxRed.BackColor = Color.FromArgb(0, 0, 0)
    PictureBoxGreen.BackColor = Color.FromArgb(0, 0, 0)
    PictureBoxBlue.BackColor = Color.FromArgb(0, 0, 0)
    Blue.Text = ""
    Green.Text = ""
    Red.Text = ""
    Pixel1.BackColor = Color.FromArgb(0, 0, 0)
    PictureBoxRed2.BackColor = Color.FromArgb(0, 0, 0)
    PictureBoxGreen2.BackColor = Color.FromArgb(0, 0, 0)
    PictureBoxBlue2.BackColor = Color.FromArgb(0, 0, 0)
    Blue2.Text = ""
    Green2.Text = ""
    Red2.Text = ""
    Pixel2.BackColor = Color.FromArgb(0, 0, 0)
End Sub

'NAGYÍTÁS
```

```
Private Sub ZoomIn_Click_1(sender As System.Object, e As System.EventArgs) Handles
 ZoomIn.Click
    ZoomOut.Enabled = True
    If zooml >= 2 Then
        ZoomIn.Enabled = False
    End If
    If zooml < 4 Then
        If zoomr = 1 Then
            zooml = zooml * 2
        Else
            zoomr = zoomr / 2
        End If
    End If
    If zooml <= 4 Then
        zoom = zooml / zoomr
        PictureBox1.Size = New Size(alap.Width * zoom, alap.Height * zoom)
        PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
        Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
        ToolStripZoom.Text = zooml & "/" & zoomr
        If pic2loaded = True Then
            PictureBox2.Size = PictureBox1.Size
            PictureBox2.SizeMode = PictureBoxSizeMode.Zoom
            Label2.Text = alap.Width & " x " & alap.Height
        End If
    End If
End Sub

'KICSINYÍTÉS
Private Sub ZoomOut_Click_1(sender As System.Object, e As System.EventArgs) Handles
ZoomOut.Click
    ZoomIn.Enabled = True
    If zoomr >= 4 Then
        ZoomOut.Enabled = False
    End If
    If zoomr <= 8 Then
        If zooml = 1 Then
            zoomr = zoomr * 2
        Else
            zooml = zooml / 2
        End If
    End If
    If zoomr <= 8 Then
        zoom = zooml / zoomr
        PictureBox1.Size = New Size(alap.Width * zoom, alap.Height * zoom)
        PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
        Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
        ToolStripZoom.Text = zooml & "/" & zoomr
        If pic2loaded = True Then
            PictureBox2.Size = PictureBox1.Size
            PictureBox2.SizeMode = PictureBoxSizeMode.Zoom
            Label2.Text = alap.Width & " x " & alap.Height
        End If
    End If

End Sub

'RÉTEG-ELLENŐRZŐ
Public Sub CheckLayer()
    volt = False
    If Layer.TextBox1.Text = "" Then
```

```
        Layer.Label2.Text = "Nem adtál meg nevet!"
        Layer.Label2.Visible = True
        volt = True
    End If
    If Layer.TextBox1.Text <> "" Then
        If numb > 0 Then
            Dim mm As Integer
            For mm = 1 To numb
                If Layer.TextBox1.Text.ToString = layername(mm) Then
                    LoadBar.Maximum = numb
                    LoadBar.Value = mm
                    volt = True
                    'Van ilyen réteg
                    Layer.Label2.Text = "A megadott réteg neve már szerepel!"
                    Layer.Label2.Visible = True
                End If
            Next
        End If
    End If
    'réteg mentése
    If volt = False Then
        numb += 1
        Label2.Text = "Numb = " & numb
        bmp(numb) = PictureBox1.Image
        ComboBox1.Enabled = True
        ComboBox1.Items.Add(Layer.TextBox1.Text)
        layername(numb) = Layer.TextBox1.Text.ToString
        ComboBox1.SelectedIndex = ComboBox1.Items.Count - 1
        If numb = 10 Then
            ToolStripNewLayer.Enabled = False
        End If
        Layer.Close()
        volt = False
    End If
End Sub

'RÉTEGVÁLASZTÁS
Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As System.EventArgs) Handles
 ComboBox1.SelectedIndexChanged
    selected = False
    If ComboBox1.Items.Count > 0 Then
        For ds = 1 To numb
            LoadBar.Maximum = numb
            LoadBar.Value = ds
            If selected = False Then
                If layername(ds) = ComboBox1.SelectedItem.ToString Then
                    selected = True
                    'Van ilyen réteg
                    pic2loaded = True
                    PictureBox2.SizeMode = PictureBoxSizeMode.Zoom
                    PictureBox2.Image = bmp(ds)
                    ToolStripUndo.Enabled = False
                    sd = ds
                    ToolStripNormal.Enabled = True
                    ToolStripRenLayer.Enabled = True
                    ToolStripInvert.Enabled = True
                    ToolStripGrey.Enabled = True
                    ButtonFilter.Enabled = True
                    CheckBox1.Enabled = True
                    PictureBox2.Width = PictureBox1.Width
```

```
                    PictureBox2.Height = PictureBox1.Height
                    PictureBox2.Visible = True
                    PictureBox2.Refresh()
                    Label2.Text = "Numb: " & sd & "név: " & layername(sd)
                End If
            End If
        Next
    End If
End Sub

'ÚJ RÉTEG GOMB MUTATÁSA
Private Sub PictureBox1_SizeChanged(sender As Object, e As System.EventArgs)
Handles PictureBox1.SizeChanged
    If PictureBox1.Width > 0 Then
        If numb < 10 Then
            ToolStripNewLayer.Enabled = True
        End If
    Else
        ToolStripNewLayer.Enabled = False
    End If
End Sub

'EGÉRMOZGÁS KINT
Private Sub Panel1_MouseMove(sender As Object, e As System.Windows.Forms.MouseEventArgs)
    ToolStripMouse.Text = ""
End Sub
Private Sub Panel2_MouseMove(sender As Object, e As System.Windows.Forms.MouseEventArgs)
    ToolStripMouse.Text = ""
End Sub

'SCROLLOZÁS ÁTADÁSA
Private Sub Panel1_Scroll(sender As Object, e As System.Windows.Forms.ScrollEventArgs)
Handles Panel1.Scroll
    If CheckBox1.Checked = True Then
        Panel2.HorizontalScroll.Value = Panel1.HorizontalScroll.Value
        Panel2.VerticalScroll.Value = Panel1.VerticalScroll.Value
    End If
End Sub
Private Sub Panel2_Scroll(sender As Object, e As System.Windows.Forms.ScrollEventArgs)
Handles Panel2.Scroll
    If CheckBox1.Checked = True Then
        Panel1.HorizontalScroll.Value = Panel2.HorizontalScroll.Value
        Panel1.VerticalScroll.Value = Panel2.VerticalScroll.Value
    End If
End Sub

'EREDETI MÉRET
Private Sub Normal_Click(sender As System.Object, e As System.EventArgs) Handles
Normal.Click
    PictureBox1.Height = alap.Height
    PictureBox1.Width = alap.Width
    Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
    zooml = 1
    zoomr = 1
    ToolStripZoom.Text = zooml & "/" & zoomr
    If pic2loaded = True Then
        PictureBox2.Height = PictureBox1.Height
        PictureBox2.Width = PictureBox1.Width
    End If
End Sub
```

```
'SIMITÁS
Private Sub Smooth_Click(sender As System.Object, e As System.EventArgs)
Handles Smooth.Click
    undopic = bmp(sd)
    ToolStripUndo.Enabled = True
    casualpic = fil.Smoothing(bmp(sd))
    Dim cloneRect As New Rectangle(0, 0, casualpic.Width, casualpic.Height)
    Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
    Dim bit As New Bitmap(casualpic)
    bmp(sd) = bit.Clone(cloneRect, format)
    PictureBox2.Image = bmp(sd)
End Sub

'LAPLACE
Private Sub laplacer_Click(sender As System.Object, e As System.EventArgs)
Handles Laplacer.Click
    undopic = bmp(sd)
    ToolStripUndo.Enabled = True
    casualpic = fil.Laplace(bmp(sd))
    Dim cloneRect As New Rectangle(0, 0, casualpic.Width, casualpic.Height)
    Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
    Dim bit As New Bitmap(casualpic)
    bmp(sd) = bit.Clone(cloneRect, format)
    PictureBox2.Image = bmp(sd)
End Sub

'INVERTÁLÁS
Private Sub ToolStripInvert_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripInvert.Click
    undopic = bmp(sd)
    ToolStripUndo.Enabled = True
    casualpic = fil.InvertColor(bmp(sd))
    Dim cloneRect As New Rectangle(0, 0, casualpic.Width, casualpic.Height)
    Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
    Dim bit As New Bitmap(casualpic)
    bmp(sd) = bit.Clone(cloneRect, format)
    PictureBox2.Image = bmp(sd)
End Sub

'SZÜRKEÁRNYALATTÁ
Private Sub ToolStripGrey_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripGrey.Click
    undopic = bmp(sd)
    ToolStripUndo.Enabled = True
    casualpic = fil.GrayScale(bmp(sd))
    Dim cloneRect As New Rectangle(0, 0, casualpic.Width, casualpic.Height)
    Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
    Dim bit As New Bitmap(casualpic)
    bmp(sd) = bit.Clone(cloneRect, format)
    PictureBox2.Image = bmp(sd)
End Sub

'NORMÁL KÉP
Private Sub ToolStripNormal_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripNormal.Click
    undopic = bmp(sd)
    ToolStripUndo.Enabled = True
    bmp(sd) = alap
    PictureBox2.Image = bmp(sd)
```

```
End Sub

'KILÉPÉS
Private Sub Quit_Click_1(sender As System.Object, e As System.EventArgs) Handles Quit.Click
    Layer.Close()
    OKernel.Close()
    PixelBite.Close()
    RenLayer.Close()
    Contrast.Close()
    Cut.Close()
    Dim answer As DialogResult
    answer = MessageBox.Show("Biztos ki akarsz lépni?", "Vectorizer - Kilépés",
    MessageBoxButtons.YesNo, MessageBoxIcon.Question)
    'igen
    If answer = DialogResult.Yes Then
        For vv = 1 To numb
            LoadBar.Maximum = numb
            LoadBar.Value = vv
            bmp(vv).Dispose()
        Next
        If PictureBox1.Width > 0 Then
            alap.Dispose()
        End If
        End
    End If
End Sub

'PROJEKT BEZÁRÁSA
Private Sub Closes_Click(sender As System.Object, e As System.EventArgs)
Handles Closes.Click
    Layer.Close()
    OKernel.Close()
    PixelBite.Close()
    RenLayer.Close()
    Contrast.Close()
    Cut.Close()
    Dim answer As DialogResult
    answer = MessageBox.Show("El akarod menteni a projektet bezárás előtt?", _
                                "Projekt bezárása", _
                                MessageBoxButtons.YesNo, _
                                MessageBoxIcon.Question)
    'igen
    If answer = DialogResult.Yes Then
        'Mentés
        Saving()
        Disposing()
    End If
    'nem
    If answer = DialogResult.No Then
        Disposing()
    End If
End Sub

'MENTÉS
Private Sub Save_Click_1(sender As System.Object, e As System.EventArgs) Handles Save.Click
    Saving()
End Sub

'MENTÉSI FOLYAMAT
Public Function Saving()
```

```
Dim sname, foldername, destine, snev As String
SaveFileDialog1.Filter = "Vectorizer|*.vec"
If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
    'MEGHATÁROZÁSOK
    sname = SaveFileDialog1.FileName 'a .vec elérése
    'MAPPA KISZEDÉSE
    destine = sname.Substring(0, sname.LastIndexOf("\") + 1) ' a.vec mappája
    'FÁJL NEVÉNEK KISZEDÉSE
    snev = sname.Substring(sname.LastIndexOf("\") + 1, sname.Length - sname.
    LastIndexOf("\") - 5)
    'a .vec neve
    foldername = destine.ToString & snev.ToString & "\" 'a többi fájl neve
    rawname = foldername & "raw.vpi" 'alap képfájl
    'mappa létrehozása
    If System.IO.Directory.Exists(foldername) = False Then
        System.IO.Directory.CreateDirectory(foldername)
    End If
    '.vec fájl megtisztítása
    If System.IO.File.Exists(sname) = True Then
        System.IO.File.Delete(sname)
    End If
    'Újraírás
    Dim Saver As New Ini(sname) 'figyelo fájl azonosítása
    'képek biztonsági "mentése"
    Dim cloneRect As New Rectangle(0, 0, alap.Width, alap.Height)
    Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
    Dim bit As New Bitmap(alap)
    trickalap = bit.Clone(cloneRect, format)
    trickalap = New Bitmap(alap)
    alap.Dispose()
    If numb > 0 Then
        For dz = 1 To numb
            Dim clonRect As New Rectangle(0, 0, bmp(dz).Width, bmp(dz).Height)
            Dim bitt As New Bitmap(bmp(dz))
            trick(dz) = bitt.Clone(clonRect, format)
            bmp(dz).Dispose()
        Next
    End If
    'mappa megtisztítása régi fájloktól
    If foldername <> foldern Then
        Try
            Kill(foldername & "*.*")
        Catch
        End Try
    End If
    'alapkép mentése
    Saver.WriteString("Layers", "Raw", rawname)
    trickalap.Save(rawname, System.Drawing.Imaging.ImageFormat.Bmp)
    'rétegek száma
    Saver.WriteString("Layers", "Count", numb)
    'rétegek neve
    For bb = 1 To numb
        Saver.WriteString("Layers", bb, layername(bb))
        LoadBar.Maximum = numb
        LoadBar.Value = bb
    Next
    'rétegek képei
    For yy = 1 To numb
        Saver.WriteString("Folder", layername(yy), foldername & layername(yy) & ".vpi")
        'képek kimentése
```

```
            trick(yy).Save(foldername & layername(yy) & ".vpi", System.Drawing.
            Imaging.ImageFormat.Bmp)
            LoadBar.Maximum = numb
            LoadBar.Value = yy
        Next
        'képek visszaállítása
        alap = trickalap
        If numb > 0 Then
            For nn = 1 To numb
                bmp(nn) = trick(nn)
            Next
        End If
        PictureBox1.Image = alap
        PictureBox2.Image = bmp(sd)
        Label2.Text = alap.Width & " x " & alap.Height
        'sikeres üzenet
        MsgBox("A mentés sikeres!", vbInformation, "Vectorizer - mentés")
    End If
End Function

'BEZÁRÁSI FOLYAMAT
Function Disposing()
    LoadBar.Maximum = numb
    PictureBox1.Visible = False
    PictureBox1.SizeMode = PictureBoxSizeMode.Normal
    PictureBox1.Height = 0
    PictureBox1.Width = 0
    ToolStripStatusLabel1.Text = ""
    Normal.Enabled = False
    ToolStripUndo.Enabled = False
    zooml = 1
    zoomr = 1
    ToolStripZoom.Text = zooml & "/" & zoomr
    Me.Text = "Vectorizer"
    If pic2loaded = True Then
        PictureBox2.Visible = False
        PictureBox1.SizeMode = PictureBoxSizeMode.Normal
        PictureBox1.Height = 0
        PictureBox1.Width = 0
        pic2loaded = False
    End If
    For hh = 1 To numb
        LoadBar.Maximum = numb
        LoadBar.Value = hh
        If hh > 0 Then
            bmp(hh).Dispose()
        End If
    Next
    For gg = 1 To numb
        LoadBar.Maximum = numb
        LoadBar.Value = gg
        If gg > 0 Then
            layername(gg) = ""
        End If
    Next
    Import.Enabled = True
    If alapvan = True Then
        alap.Dispose()
    End If
    OKernel.Enabled = False
```

```
    ComboBox1.Items.Clear()
    ComboBox1.Enabled = False
    ToolStripNormal.Enabled = False
    ToolStripRenLayer.Enabled = False
    ToolStripInvert.Enabled = False
    ToolStripGrey.Enabled = False
    ButtonFilter.Enabled = False
    ToolStripNewLayer.Enabled = False
    Recent.Enabled = True
    ZoomIn.Enabled = False
    ZoomOut.Enabled = False
    CheckBox1.Enabled = False
    LoadBar.Value = 0
    Label8.Text = ""
    Closes.Enabled = False
    Save.Enabled = False
    numb = 0
End Function

'PROJEKT MEGNYITÁSA
Private Sub Open_Click(sender As System.Object, e As System.EventArgs) Handles Open.Click
    Dim casname1, oldrecent, oldrecentdest, casname2, rawdest, picdest As String
    Dim van As Boolean
    Disposing()
    OpenFileDialog2.Filter = "Vectorizer|*.vec"
    van = True
    If OpenFileDialog2.ShowDialog = Windows.Forms.DialogResult.OK Then
        lname = OpenFileDialog2.FileName
        'MAPPA KISZEDÉSE
        casname1 = lname.Substring(0, lname.LastIndexOf("\") + 1)
        casname2 = lname.Substring(lname.LastIndexOf("\") + 1, lname.Length -
        lname.LastIndexOf("\") - 5)
        foldern = casname1 & casname2 & "\"
        '.vec fájl megnyitása
        Dim Opener As New Ini(lname)
        'recent fájlok
        Dim Recenter As New Ini(Application.StartupPath & "\Recent.rec")
        'rétegek száma
        numb = Opener.GetString("Layers", "Count", "0")
        'alap helyének betöltése
        rawdest = Opener.GetString("Layers", "Raw", "")
        oldrecentdest = Recenter.GetString("Recent", "NewerDest", "")
        oldrecent = Recenter.GetString("Recent", "Newer", "")
        If oldrecentdest <> lname Then
            If oldrecentdest <> "" Then
                'régi recent hátra
                Recenter.WriteString("Recent", "OlderDest", oldrecentdest)
                Recenter.WriteString("Recent", "Older", oldrecent)
            End If
            'új recent előre
            Recenter.WriteString("Recent", "Newer", casname2 & ".vec")
            Recenter.WriteString("Recent", "NewerDest", lname)
            ToolStripRecentN.Text = casname2 & ".vec"
            ToolStripRecentN.Visible = True
        End If
        If oldrecent <> "" Then
            ToolStripRecent0.Visible = True
            ToolStripRecent0.Text = oldrecent
        Else
            ToolStripRecent0.Visible = False
```

```
End If
If System.IO.File.Exists(rawdest) = True Then
    alap = Image.FromFile(rawdest)
    alapvan = True
Else
    alapvan = False
    MsgBox("A fájl nem található: " & rawdest, vbExclamation, "Vectorizer
     - betöltés")
    numb = 0
    Disposing()
    Exit Sub
End If
'rétegnevek betöltése
For bb = 1 To numb
    layername(bb) = Opener.GetString("Layers", bb, "")
    LoadBar.Maximum = numb
    LoadBar.Value = bb
Next
'combobox feltöltése
For zz = 1 To numb
    ComboBox1.Items.Add(layername(zz))
    LoadBar.Maximum = numb
    LoadBar.Value = zz
Next
'képek betöltése
If van = True Then
    For hh = 1 To numb
        LoadBar.Maximum = numb
        LoadBar.Value = hh
        picdest = Opener.GetString("Folder", layername(hh), "")
        If System.IO.File.Exists(picdest) = True Then
            bmp(hh) = Image.FromFile(picdest)
        Else
            MsgBox("A fájl nem található: " & picdest, vbExclamation,
            "Vectorizer - betöltés")
            van = False
            numb = 0
            hh = 0
            Disposing()
            Exit For
        End If
        If van = True Then
            LoadBar.Maximum = numb
            LoadBar.Value = hh
        End If
    Next
End If
If van = True Then
    'képek megjelentése
    PictureBox1.Image = alap
    If numb > 0 Then
        pic2loaded = True
        ComboBox1.Enabled = True
        ComboBox1.SelectedIndex = 0
        PictureBox2.Visible = True
        PictureBox2.Width = alap.Width
        PictureBox2.Height = alap.Height
        ComboBox1.SelectedIndex = 0
    End If
    'eygéb beállítások
```

```
            zoomr = 1
            zooml = 1
            ToolStripZoom.Text = zooml & "/" & zoomr
            Normal.Enabled = True
            ZoomIn.Enabled = True
            ZoomOut.Enabled = True
            Import.Enabled = False
            Closes.Enabled = True
            Recent.Enabled = False
            Save.Enabled = True
            PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
            PictureBox1.Image = alap
            PictureBox1.Width = alap.Width
            PictureBox1.Height = alap.Height
            PictureBox1.Visible = True
            PictureBox1.Refresh()
            ToolStripStatusLabel1.Text = "Szélesség: " & alap.Width & "pixel,
            Hossz: " & alap.Height & "pixel,
             Színmélység: " & alap.PixelFormat.ToString
            Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
            Me.Text = "Vectorizer - Megnyitott fájl: " + rawdest
            'sikeres üzenet
            MsgBox("A betöltés sikeres!", vbInformation, "Vectorizer - betöltés")
        End If
    End If
End Sub

'SAJÁT KERNEL
Private Sub OwnFilter_Click(sender As System.Object, e As System.EventArgs) Handles
OwnFilter.Click
    OKernel.Show()
    OKernel.Enabled = True
End Sub

'LEGFRISSEBB RECENT FÁJL
Private Sub ToolStripRecentN_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripRecentN.Click
    If ToolStripRecentN.Text <> "" Then
        'recent adatok
        Dim lname, rawdest, picdest As String
        Dim van As Boolean
        Dim Recenter As New Ini(Application.StartupPath & "\Recent.rec")
        lname = Recenter.GetString("Recent", "NewerDest", "")
        van = True
        'fájlok megnyitása
        Dim Opener As New Ini(lname)
        'rétegek száma
        numb = Opener.GetString("Layers", "Count", "0")
        'alap helyének betöltése
        rawdest = Opener.GetString("Layers", "Raw", "")
        If System.IO.File.Exists(rawdest) = True Then
            alap = Image.FromFile(rawdest)
            alapvan = True
        Else
            alapvan = False
            MsgBox("A fájl nem található: " & rawdest, vbExclamation, "Vectorizer
            - betöltés")
            numb = 0
            Disposing()
            Exit Sub
```

```
End If
'rétegnevek betöltése
For ff = 1 To numb
    layername(ff) = Opener.GetString("Layers", ff, "")
    LoadBar.Maximum = numb
    LoadBar.Value = ff
Next
'combobox feltöltése
For jj = 1 To numb
    ComboBox1.Items.Add(layername(jj))
    LoadBar.Maximum = numb
    LoadBar.Value = jj
Next
'képek betöltése
If van = True Then
    For mm = 1 To numb
        LoadBar.Maximum = numb
        LoadBar.Value = mm
        picdest = Opener.GetString("Folder", layername(mm), "")
        If System.IO.File.Exists(picdest) = True Then
            bmp(mm) = Image.FromFile(picdest)
        Else
            MsgBox("A fájl nem található: " & picdest, vbExclamation,
            "Vectorizer - betöltés")
            van = False
            numb = 0
            mm = 0
            Disposing()
            Exit For
        End If
        If van = True Then
            LoadBar.Maximum = numb
            LoadBar.Value = mm
        End If
    Next
End If
If van = True Then
    'képek megjelentése
    PictureBox1.Image = alap
    If numb > 0 Then
        pic2loaded = True
        ComboBox1.Enabled = True
        ComboBox1.SelectedIndex = 0
        PictureBox2.Visible = True
        PictureBox2.Width = alap.Width
        PictureBox2.Height = alap.Height
        ComboBox1.SelectedIndex = 0
    End If
    'egyéb beállítások
    zoomr = 1
    zooml = 1
    ToolStripZoom.Text = zooml & "/" & zoomr
    Normal.Enabled = True
    ZoomIn.Enabled = True
    ZoomOut.Enabled = True
    Import.Enabled = False
    Closes.Enabled = True
    Recent.Enabled = False
    Save.Enabled = True
    PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
```

```
            PictureBox1.Image = alap
            PictureBox1.Width = alap.Width
            PictureBox1.Height = alap.Height
            PictureBox1.Visible = True
            PictureBox1.Refresh()
            ToolStripStatusLabel1.Text = "Szélesség: " & alap.Width & "pixel,
            Hossz: " & alap.Height & "pixel,
             Színmélység: " & alap.PixelFormat.ToString
            Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
            Me.Text = "Vectorizer - Megnyitott fájl: " + rawdest
            'sikeres üzenet
            MsgBox("A betöltés sikeres!", vbInformation, "Vectorizer - betöltés")
        End If
    End If
End Sub

'RÉGEBBI RECENT FÁJL
Private Sub ToolStripRecentO_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripRecentO.Click
    If ToolStripRecentO.Text <> "" Then
        'recent adatok
        Dim lname, rawdest, picdest As String
        Dim van As Boolean
        Dim Recenter As New Ini(Application.StartupPath & "\Recent.rec")
        lname = Recenter.GetString("Recent", "OlderDest", "")
        van = True
        'fájlok megnyitása
        Dim Opener As New Ini(lname)
        'rétegek száma
        numb = Opener.GetString("Layers", "Count", "0")
        'alap helyének betöltése
        rawdest = Opener.GetString("Layers", "Raw", "")
        If System.IO.File.Exists(rawdest) = True Then
            alap = Image.FromFile(rawdest)
            alapvan = True
        Else
            alapvan = False
            MsgBox("A fájl nem található: " & rawdest, vbExclamation, "Vectorizer -
            betöltés")
            numb = 0
            Disposing()
            Exit Sub
        End If
        'rétegnevek betöltése
        For bb = 1 To numb
            layername(bb) = Opener.GetString("Layers", bb, "")
            LoadBar.Maximum = numb
            LoadBar.Value = bb
        Next
        'combobox feltöltése
        For vv = 1 To numb
            ComboBox1.Items.Add(layername(vv))
            LoadBar.Maximum = numb
            LoadBar.Value = vv
        Next
        'képek betöltése
        If van = True Then
            For cc = 1 To numb
                LoadBar.Maximum = numb
                LoadBar.Value = cc
```

```
                picdest = Opener.GetString("Folder", layername(cc), "")
                If System.IO.File.Exists(picdest) = True Then
                    bmp(cc) = Image.FromFile(picdest)
                Else
                    MsgBox("A fájl nem található: " & picdest, vbExclamation,
                    "Vectorizer - betöltés")
                    van = False
                    numb = 0
                    cc = 0
                    Disposing()
                    Exit For
                End If
                If van = True Then
                    LoadBar.Maximum = numb
                    LoadBar.Value = cc
                End If
            Next
        End If
        If van = True Then
            'képek megjelentése
            PictureBox1.Image = alap
            If numb > 0 Then
                pic2loaded = True
                ComboBox1.Enabled = True
                ComboBox1.SelectedIndex = 0
                PictureBox2.Visible = True
                PictureBox2.Width = alap.Width
                PictureBox2.Height = alap.Height
                ComboBox1.SelectedIndex = 0
            End If
            'egyéb beállítások
            zoomr = 1
            zooml = 1
            ToolStripZoom.Text = zooml & "/" & zoomr
            Normal.Enabled = True
            ZoomIn.Enabled = True
            ZoomOut.Enabled = True
            Import.Enabled = False
            Closes.Enabled = True
            Recent.Enabled = False
            Save.Enabled = True
            PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
            PictureBox1.Image = alap
            PictureBox1.Width = alap.Width
            PictureBox1.Height = alap.Height
            PictureBox1.Visible = True
            PictureBox1.Refresh()
            ToolStripStatusLabel1.Text = "Szélesség: " & alap.Width & "pixel,
            Hossz: " & alap.Height & "pixel,
             Színmélység: " & alap.PixelFormat.ToString
            Label1.Text = "Kép mérete: " & PictureBox1.Width & " x " & PictureBox1.Height
            Me.Text = "Vectorizer - Megnyitott fájl: " + rawdest
            'sikeres üzenet
            MsgBox("A betöltés sikeres!", vbInformation, "Vectorizer - betöltés")
        End If
    End If
End Sub

'Kép visszaállítása
Private Sub ToolStripUndo_Click(sender As System.Object, e As System.EventArgs)
```

```
Handles ToolStripUndo.Click
    bmp(sd) = undopic
    PictureBox2.Image = bmp(sd)
    ToolStripUndo.Enabled = False
End Sub

'PICTUREBOX2 KATTINTÁS
Private Sub PictureBox2_MouseDown(sender As Object, e As System.Windows.Forms.
MouseEventArgs) Handles PictureBox2.MouseDown
    'Context menu
    If e.Button = MouseButtons.Right Then
        ContextMenuStrip1.Show(New Point(MousePosition.X, MousePosition.Y))
    End If
    'Pixel kiszedése
    If e.Button = MouseButtons.Left Then
        Try
            If zoomr = 1 Then
                PixelBite.RGB.BackColor = bmp(sd).GetPixel(e.X / zooml, e.Y / zooml)
            Else
                PixelBite.RGB.BackColor = bmp(sd).GetPixel(e.X * zoomr, e.Y * zoomr)
            End If
        Catch
        End Try
    End If
End Sub

'KÉPKIMENTÉS
Private Sub OutSave_Click(sender As System.Object, e As System.EventArgs) Handles OutSave.Click
    Dim fsave As New SaveFileDialog
    fsave.Filter = "*.bmp|*.bmp|*.jpg|*.jpg|*.tif|*.tif|*.gif|*.gif|*.png|*.png"
    If fsave.ShowDialog = Windows.Forms.DialogResult.OK Then
        Select Case System.IO.Path.GetExtension(fsave.FileName)
            Case ".tif"
                PictureBox2.Image.Save(fsave.FileName, System.Drawing.Imaging.ImageFormat.Tiff)
            Case ".bmp"
                PictureBox2.Image.Save(fsave.FileName, System.Drawing.Imaging.ImageFormat.Bmp)
            Case ".jpg"
                PictureBox2.Image.Save(fsave.FileName, System.Drawing.Imaging.ImageFormat.Jpeg)
            Case ".gif"
                PictureBox2.Image.Save(fsave.FileName, System.Drawing.Imaging.ImageFormat.Gif)
            Case ".png"
                PictureBox2.Image.Save(fsave.FileName, System.Drawing.Imaging.ImageFormat.Png)
        End Select
        MsgBox("A kép kimentése sikeres!", vbInformation, "Vectorizer - kép kimentése")
    End If
End Sub

'ÚJ RÉTEG
Private Sub ToolStripNewLayer_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripNewLayer.Click
    Layer.Show()
End Sub

'PIXEL MÓDOSÍTÁSA
Private Sub ToolStripPixel_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripPixel.Click
    PixelBite.Show()
End Sub

'RÉTEG ÁTNEVEZÉSE
```

```
Private Sub ToolStripRenLayer_Click(sender As System.Object, e As System.EventArgs)
Handles ToolStripRenLayer.Click
    RenLayer.Show()
End Sub

'RÉTEG ÁTNEVEZÉS
Public Sub RenameLayer()
    volt = False
    If RenLayer.TextBox1.Text = "" Then
        RenLayer.Label2.Text = "Nem adtál meg nevet!"
        RenLayer.Label2.Visible = True
        volt = True
    End If
    If RenLayer.TextBox1.Text <> "" Then
        If numb > 0 Then

            For dd = 1 To numb
                If RenLayer.TextBox1.Text.ToString = layername(dd) Then
                    LoadBar.Maximum = numb
                    LoadBar.Value = dd
                    volt = True
                    'Van ilyen réteg
                    RenLayer.Label2.Text = "Ez a rétegnév már szerepel!"
                    RenLayer.Label2.Visible = True
                End If
            Next
        End If
    End If
    'réteg mentése
    If volt = False Then
        Label2.Text = "Numb = " & numb
        ComboBox1.Items.RemoveAt(ComboBox1.SelectedIndex)
        ComboBox1.Items.Add(RenLayer.TextBox1.Text)
        layername(sd) = RenLayer.TextBox1.Text.ToString
        ComboBox1.SelectedIndex = ComboBox1.Items.Count - 1
        RenLayer.Close()
        volt = False
    End If
End Sub

'KONTRASZT
Private Sub KontrasztToolStripMenuItem_Click(sender As System.Object, e As System.EventArgs)
 Handles KontrasztToolStripMenuItem.Click
    Contrast.Show()
End Sub

'VÁGÓ
Private Sub VágóToolStripMenuItem_Click(sender As System.Object, e As System.EventArgs)
Handles VágóToolStripMenuItem.Click
    Cut.Show()
End Sub

'ABOUTBOX
Private Sub AProgramrólToolStripMenuItem_Click(sender As System.Object, e As System.EventArgs)
 Handles AProgramrólToolStripMenuItem.Click
    AboutBox1.Show()
End Sub

'SÚGÓ
Private Sub SúgóToolStripMenuItem_Click(sender As System.Object, e As System.EventArgs)
```

```
    Handles SúgóToolStripMenuItem.Click
        Help.Show()
    End Sub
End Class
```

CONTRAST

```
Public Class Contrast
    Dim fil As New Filters
    Dim prewpic, miniprepic, caspic As Image
    'BILLENTYŰK
    Private Sub TextBox1_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
    Handles TextBox1.KeyPress
        If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
            If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL") Then
                e.Handled = True
            End If
        End If
    End Sub

    'Contrasztolás
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Form1.undopic = Form1.bmp(Form1.sd)
        Form1.ToolStripUndo.Enabled = True
        Form1.casualpic = fil.Contraster(Form1.bmp(Form1.sd))
        Form1.bmp(Form1.sd) = Form1.casualpic
        Form1.PictureBox2.Image = Form1.bmp(Form1.sd)
        Me.Close()
    End Sub

    'SCROLLOZÁS
    Private Sub TrackBar1_Scroll_1(sender As System.Object, e As System.EventArgs) Handles
    TrackBar1.Scroll
        Label1.Text = TrackBar1.Value
    End Sub

    'MEHET-E?
    Private Sub TextBox1_TextChanged(sender As Object, e As System.EventArgs) Handles TextBox1.
    TextChanged
        If TextBox1.Text.Length > 0 Then
            Button1.Enabled = True
            Button2.Enabled = True
        Else
            Button1.Enabled = False
            Button2.Enabled = True
        End If
    End Sub

    'BETÖLTÉS
    Private Sub Contrast_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        prewpic = Form1.bmp(Form1.sd)
        miniprepic = prewpic.GetThumbnailImage(100 * prewpic.Width / prewpic.Height, 100, Nothing,
        New IntPtr)
        ' Clone a portion of the Bitmap object.
        Dim cloneRect As New Rectangle(0, 0, miniprepic.Width, miniprepic.Height)
        Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
        Dim bit As New Bitmap(miniprepic)
        caspic = bit.Clone(cloneRect, format)
        PictureBox2.Image = caspic
        TextBox1.Focus()
```

```
        End Sub

    'FÓKUSZ ÁLLÍTÁS
    Private Sub Button1_GotFocus(sender As Object, e As System.EventArgs) Handles Button1.GotFocus
        TextBox1.Focus()
    End Sub

    'FUNKCIÓ KICSINYÍTETT KÉPRE
    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
        PictureBox2.Image = fil.Contraster(caspic)
    End Sub
End Class
```

CUT

```
Public Class Cut
    Dim fil As New Filters
    Public scroller As Integer
    Dim prewpic, miniprepic, caspic As Image

    'ÉRTÉKADÁS
    Private Sub TrackBar1_Scroll(sender As System.Object, e As System.EventArgs)
    Handles TrackBar1.Scroll
        Label1.Text = TrackBar1.Value
    End Sub

    'TÖBBI PIXEL SZÍNE
    Private Sub VantR_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
    Handles VantR.Scroll
        RGBVant.BackColor = Color.FromArgb(VantR.Value, VantG.Value, VantB.Value)
        Rlike.Text = VantR.Value
    End Sub
    Private Sub VantG_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
    Handles VantG.Scroll
        RGBVant.BackColor = Color.FromArgb(VantR.Value, VantG.Value, VantB.Value)
        Glike.Text = VantG.Value
    End Sub
    Private Sub VantB_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
    Handles VantB.Scroll
        RGBVant.BackColor = Color.FromArgb(VantR.Value, VantG.Value, VantB.Value)
        Blike.Text = VantB.Value
    End Sub

    'BETÖLTÉS
    Private Sub Cut_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        prewpic = Form1.bmp(Form1.sd)
        miniprepic = prewpic.GetThumbnailImage(100 * prewpic.Width / prewpic.Height, 100, Nothing,
        New IntPtr)
        ' Clone a portion of the Bitmap object.
        Dim cloneRect As New Rectangle(0, 0, miniprepic.Width, miniprepic.Height)
        Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
        Dim bit As New Bitmap(miniprepic)
        caspic = bit.Clone(cloneRect, format)
        PictureBox2.Image = caspic
        RGBVant.BackColor = Color.FromArgb(0, 0, 0)
    End Sub

    'MEHET!
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Form1.undopic = Form1.bmp(Form1.sd)
```

```
        Form1.ToolStripUndo.Enabled = True
        Form1.casualpic = fil.Cutter(Form1.bmp(Form1.sd))
        Form1.bmp(Form1.sd) = Form1.casualpic
        Form1.PictureBox2.Image = Form1.bmp(Form1.sd)
        Me.Close()
    End Sub

    'INTENZITÁS-VÁLASZTÁS
    Private Sub RadioButton1_CheckedChanged(sender As System.Object, e As System.EventArgs)
    Handles RadioButton1.CheckedChanged
        scroller = 1
        Button1.Enabled = True
        Button2.Enabled = True
    End Sub
    Private Sub RadioButton2_CheckedChanged(sender As System.Object, e As System.EventArgs)
    Handles RadioButton2.CheckedChanged
        scroller = 2
        Button1.Enabled = True
        Button2.Enabled = True
    End Sub

    'KICSINYÍTETT KÉP
    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
        PictureBox2.Image = fil.Cutter(caspic)
    End Sub
End Class
```

LAYER

```
Public Class Layer

    'RÉTEG ELKÜLDÉSE
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Form1.CheckLayer()
    End Sub

    'RÉTEG ELKÜLDÉSE ENTERREL
    Private Sub TextBox1_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
    Handles TextBox1.KeyPress
        Label2.Text = ""
        If Asc(e.KeyChar) = 13 Then
            Form1.CheckLayer()
        End If
    End Sub

    'FÓKUSZÁLÁS
    Private Sub Layer_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        TextBox1.Focus()
    End Sub
    Private Sub Button1_GotFocus(sender As Object, e As System.EventArgs) Handles Button1.GotFocus
        TextBox1.Focus()
    End Sub
End Class
```

OKERNEL

```
Public Class OKernel
    Dim fil As New Filters
    Dim PicByte As New ImageByteArray
```

```
Dim prewpic, miniprepic, caspic As Image

'SAJÁT SZŰRŐ
Private Sub OkKernel_Click(sender As System.Object, e As System.EventArgs) Handles OkKernel.Click
    Form1.undopic = Form1.bmp(Form1.sd)
    Form1.ToolStripUndo.Enabled = True
    Form1.casualpic = fil.Own(Form1.bmp(Form1.sd))
    Form1.bmp(Form1.sd) = Form1.casualpic
    Form1.PictureBox2.Image = Form1.bmp(Form1.sd)
    Me.Close()
End Sub

'SZÁM BEVITELE
Private Sub TextBox1_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox1.KeyPress
    If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
        If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
        e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
            e.Handled = True
        End If
    End If
End Sub
Private Sub TextBox2_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox2.KeyPress
    If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
        If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
        e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
            e.Handled = True
        End If
    End If
End Sub
Private Sub TextBox3_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox3.KeyPress
    If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
        If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
        e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
            e.Handled = True
        End If
    End If
End Sub
Private Sub TextBox4_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox4.KeyPress
    If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
        If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
        e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
            e.Handled = True
        End If
    End If
End Sub
Private Sub TextBox5_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox5.KeyPress
    If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
        If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
        e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
            e.Handled = True
        End If
    End If
End Sub
Private Sub TextBox6_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
Handles TextBox6.KeyPress
```

```
        If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
            If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
            e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
                e.Handled = True
            End If
        End If
    End Sub
    Private Sub TextBox7_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
    Handles TextBox7.KeyPress
        If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
            If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
            e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
                e.Handled = True
            End If
        End If
    End Sub
    Private Sub TextBox8_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
    Handles TextBox8.KeyPress
        If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
            If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
            e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
                e.Handled = True
            End If
        End If
    End Sub
    Private Sub TextBox9_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
    Handles TextBox9.KeyPress
        If (e.KeyChar < "0" OrElse e.KeyChar > "9") Then
            If (e.KeyChar <> Microsoft.VisualBasic.ChrW(8) AndAlso e.KeyChar <> "DEL" AndAlso
            e.KeyChar <> "-" AndAlso e.KeyChar <> ".") Then
                e.Handled = True
            End If
        End If
    End Sub

    'KICSINYÍTETT KÉP
    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
        PictureBox2.Image = fil.Own(caspic)
    End Sub

    'BETÖLTÉS
    Private Sub OKernel_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        prewpic = Form1.bmp(Form1.sd)
        miniprepic = prewpic.GetThumbnailImage(100 * prewpic.Width / prewpic.Height, 100, Nothing,
        New IntPtr)
        ' Clone a portion of the Bitmap object.
        Dim cloneRect As New Rectangle(0, 0, miniprepic.Width, miniprepic.Height)
        Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
        Dim bit As New Bitmap(miniprepic)
        caspic = bit.Clone(cloneRect, format)
        PictureBox2.Image = caspic
    End Sub
End Class
```

PIXELBITE

```
Public Class PixelBite
    Public minred, mingreen, minblue As Integer
    Public maxred, maxgreen, maxblue As Integer
    Public othercolor As Boolean
```

```
Dim prewpic, miniprepic, caspic As Image
Dim fil As New Filters

'TOLERANCIA CSÚSZKÁK
Private Sub RScroll_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles RScroll.Scroll
    Rvalue.Text = "+-" & RScroll.Value
    minred = Val(Label14.Text) - RScroll.Value
    If minred < 0 Then minred = 0
    mingreen = Val(Label15.Text) - GScroll.Value
    If mingreen < 0 Then mingreen = 0
    minblue = Val(Label16.Text) - BScroll.Value
    If minblue < 0 Then minblue = 0

    maxred = Val(Label14.Text) + RScroll.Value
    If maxred > 255 Then maxred = 255
    maxgreen = Val(Label15.Text) + GScroll.Value
    If maxgreen > 255 Then maxgreen = 255
    maxblue = Val(Label16.Text) + BScroll.Value
    If maxblue > 255 Then maxblue = 255

    MinPixel.BackColor = Color.FromArgb(minred, mingreen, minblue)
    MaxPixel.BackColor = Color.FromArgb(maxred, maxgreen, maxblue)
End Sub
Private Sub GScroll_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles GScroll.Scroll
    Gvalue.Text = "+-" & GScroll.Value
    minred = Val(Label14.Text) - RScroll.Value
    If minred < 0 Then minred = 0
    mingreen = Val(Label15.Text) - GScroll.Value
    If mingreen < 0 Then mingreen = 0
    minblue = Val(Label16.Text) - BScroll.Value
    If minblue < 0 Then minblue = 0

    maxred = Val(Label14.Text) + RScroll.Value
    If maxred > 255 Then maxred = 255
    maxgreen = Val(Label15.Text) + GScroll.Value
    If maxgreen > 255 Then maxgreen = 255
    maxblue = Val(Label16.Text) + BScroll.Value
    If maxblue > 255 Then maxblue = 255

    MinPixel.BackColor = Color.FromArgb(minred, mingreen, minblue)
    MaxPixel.BackColor = Color.FromArgb(maxred, maxgreen, maxblue)
End Sub
Private Sub BScroll_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles BScroll.Scroll
    Bvalue.Text = "+-" & BScroll.Value
    minred = Val(Label14.Text) - RScroll.Value
    If minred < 0 Then minred = 0
    mingreen = Val(Label15.Text) - GScroll.Value
    If mingreen < 0 Then mingreen = 0
    minblue = Val(Label16.Text) - BScroll.Value
    If minblue < 0 Then minblue = 0

    maxred = Val(Label14.Text) + RScroll.Value
    If maxred > 255 Then maxred = 255
    maxgreen = Val(Label15.Text) + GScroll.Value
    If maxgreen > 255 Then maxgreen = 255
    maxblue = Val(Label16.Text) + BScroll.Value
    If maxblue > 255 Then maxblue = 255
```

```
    MinPixel.BackColor = Color.FromArgb(minred, mingreen, minblue)
    MaxPixel.BackColor = Color.FromArgb(maxred, maxgreen, maxblue)
End Sub

'TÖBBI PIXEL SZÍNE
Private Sub VantR_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles VantR.Scroll
    RGBVant.BackColor = Color.FromArgb(VantR.Value, VantG.Value, VantB.Value)
    Rlike.Text = VantR.Value
End Sub
Private Sub VantG_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles VantG.Scroll
    RGBVant.BackColor = Color.FromArgb(VantR.Value, VantG.Value, VantB.Value)
    Glike.Text = VantG.Value
End Sub
Private Sub VantB_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles VantB.Scroll
    RGBVant.BackColor = Color.FromArgb(VantR.Value, VantG.Value, VantB.Value)
    Blike.Text = VantB.Value
End Sub

'BETÖLTÉS
Private Sub PixelBite_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
    prewpic = Form1.bmp(Form1.sd)
    miniprepic = prewpic.GetThumbnailImage(100 * prewpic.Width / prewpic.Height, 100, Nothing,
    New IntPtr)
    ' Clone a portion of the Bitmap object.
    Dim cloneRect As New Rectangle(0, 0, miniprepic.Width, miniprepic.Height)
    Dim format As Imaging.PixelFormat = Imaging.PixelFormat.Format24bppRgb
    Dim bit As New Bitmap(miniprepic)
    caspic = bit.Clone(cloneRect, format)
    PictureBox4.Image = caspic

    Button1.Enabled = False
    Button2.Enabled = False
    VantR.Value = VantG.Value = VantB.Value = 0
    GScroll.Value = RScroll.Value = BScroll.Value = 0
    Glike.Text = 0
    Rlike.Text = 0
    Blike.Text = 0
    Gvalue.Text = "+-0"
    Rvalue.Text = "+-0"
    Bvalue.Text = "+-0"
    RGBVant.BackColor = Color.FromArgb(0, 0, 0)
End Sub

'MI LEGYEN A TÖBBI PIXELLEL?
Private Sub RadioButton2_CheckedChanged(sender As System.Object, e As System.EventArgs)
Handles RadioButton2.CheckedChanged
    If RadioButton2.Checked = True Then
        othercolor = True
        Panel1.Enabled = True
    Else
        othercolor = False
        Panel1.Enabled = False
    End If
End Sub

'ÚJ PIXEL-SZÍN
```

```
Private Sub NewRScroll_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles NewRScroll.Scroll
    NewR.Text = NewRScroll.Value
    NewPictureBox.BackColor = Color.FromArgb(NewRScroll.Value, NewGScroll.Value, NewBScroll.Value)
End Sub
Private Sub NewGScroll_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles NewGScroll.Scroll
    NewG.Text = NewGScroll.Value
    NewPictureBox.BackColor = Color.FromArgb(NewRScroll.Value, NewGScroll.Value, NewBScroll.Value)
End Sub
Private Sub NewBScroll_Scroll(sender As System.Object, e As System.Windows.Forms.ScrollEventArgs)
Handles NewBScroll.Scroll
    NewB.Text = NewBScroll.Value
    NewPictureBox.BackColor = Color.FromArgb(NewRScroll.Value, NewGScroll.Value, NewBScroll.Value)
End Sub

'ÚJ KATTINTOTT SZÍN
Private Sub RGB_BackColorChanged(sender As Object, e As System.EventArgs) Handles
RGB.BackColorChanged
    Button1.Enabled = True
    Button2.Enabled = True
    Dim colR As Byte = RGB.BackColor.R
    Label14.Text = RGB.BackColor.R.ToString
    PictureBox1.BackColor = Color.FromArgb(colR, 0, 0)
    Dim colG As Byte = RGB.BackColor.G
    Label15.Text = RGB.BackColor.G.ToString
    PictureBox2.BackColor = Color.FromArgb(0, colG, 0)
    Dim colB As Byte = RGB.BackColor.B
    Label16.Text = RGB.BackColor.B.ToString
    PictureBox3.BackColor = Color.FromArgb(0, 0, colB)

    minred = Val(Label14.Text) - RScroll.Value
    If minred < 0 Then minred = 0
    mingreen = Val(Label15.Text) - GScroll.Value
    If mingreen < 0 Then mingreen = 0
    minblue = Val(Label16.Text) - BScroll.Value
    If minblue < 0 Then minblue = 0

    maxred = Val(Label14.Text) + RScroll.Value
    If maxred > 255 Then maxred = 255
    maxgreen = Val(Label15.Text) + GScroll.Value
    If maxgreen > 255 Then maxgreen = 255
    maxblue = Val(Label16.Text) + BScroll.Value
    If maxblue > 255 Then maxblue = 255

    MinPixel.BackColor = Color.FromArgb(minred, mingreen, minblue)
    MaxPixel.BackColor = Color.FromArgb(maxred, maxgreen, maxblue)
End Sub

'MEHET!
Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
    Form1.undopic = Form1.bmp(Form1.sd)
    Form1.ToolStripUndo.Enabled = True
    Form1.casualpic = fil.OwnPixel(Form1.bmp(Form1.sd))
    Form1.bmp(Form1.sd) = Form1.casualpic
    Form1.PictureBox2.Image = Form1.bmp(Form1.sd)
    Me.Close()
End Sub

'KICSINYÍTETT KÉP
```

```
    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
        PictureBox4.Image = fil.OwnPixel(caspic)
    End Sub
End Class
```

RENLAYER

```
Public Class RenLayer

    'MEHET!
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Form1.RenameLayer()
    End Sub
    'RÉTEGNÉV ELKÜLDÉSE ENTERREL
    Private Sub TextBox1_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs)
    Handles TextBox1.KeyPress
        Label2.Text = ""
        If Asc(e.KeyChar) = 13 Then
            Form1.RenameLayer()
        End If
    End Sub

    'FÓKUSZÁLÁS
    Private Sub Layer_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        TextBox1.Focus()
    End Sub
    Private Sub Button1_GotFocus(sender As Object, e As System.EventArgs) Handles Button1.GotFocus
        TextBox1.Focus()
    End Sub
End Class
```

IMAGEBYTETOARRAY

```
Imports System.Drawing.Imaging

Public Class ImageByteArray

    Function BitmapToByteArray(ByVal img As Bitmap) As Byte()
        Dim bmpData As BitmapData = img.LockBits(New Rectangle(0, 0, img.Width, img.Height),
        ImageLockMode.ReadOnly, img.PixelFormat)
        Dim pixelBytes As Integer = System.Drawing.Image.GetPixelFormatSize(img.PixelFormat) / 8
        Dim ptr As IntPtr = bmpData.Scan0
        Dim size As Int32 = bmpData.Stride * bmpData.Height
        Dim byteOut(size - 1) As Byte
        System.Runtime.InteropServices.Marshal.Copy(ptr, byteOut, 0, size)
        img.UnlockBits(bmpData)
        _ImHeight = img.Height
        _ImWidth = img.Width
        _bytesPerPixel = pixelBytes
        _stride = bmpData.Stride
        _colDepth = img.PixelFormat
        Return byteOut
    End Function

    Function ByteArrayToBitmap(ByVal byteIn() As Byte) As Bitmap
        Dim picOut As Bitmap
        picOut = New Bitmap(_ImWidth, _ImHeight, _colDepth)
        Dim bmpdata As BitmapData = picOut.LockBits(New Rectangle(0, 0, picOut.Width, picOut.Height),
        ImageLockMode.WriteOnly, _colDepth)
```

```
        Dim ptr As IntPtr = bmpdata.Scan0
        Dim size As Int32 = bmpdata.Stride * bmpdata.Height
        System.Runtime.InteropServices.Marshal.Copy(byteIn, 0, ptr, size)
        picOut.UnlockBits(bmpdata)
        Return picOut
    End Function

    Function ByteArrayToBitmap(ByVal byteIn() As Byte, ByVal ImWidth As Integer, ByVal ImHeight
    As Integer, ByVal colDepth As PixelFormat) As Bitmap
        Dim picOut As Bitmap
        picOut = New Bitmap(ImWidth, ImHeight, colDepth)
        Dim bmpdata As BitmapData = picOut.LockBits(New Rectangle(0, 0, picOut.Width, picOut.Height),
         ImageLockMode.WriteOnly, colDepth)
        Dim pixelBytes As Integer = System.Drawing.Image.GetPixelFormatSize(colDepth) / 8
        Dim ptr As IntPtr = bmpdata.Scan0
        Dim size As Int32 = bmpdata.Stride * bmpdata.Height
        _bytesPerPixel = pixelBytes
        _ImHeight = ImHeight
        _ImWidth = ImWidth
        _stride = bmpdata.Stride
        _colDepth = colDepth
        System.Runtime.InteropServices.Marshal.Copy(byteIn, 0, ptr, size)
        picOut.UnlockBits(bmpdata)
        Return picOut
    End Function
    Private _ImWidth As Int32
    Public ReadOnly Property ImageWidth() As Int32
        Get
            Return _ImWidth
        End Get
    End Property
    Private _ImHeight As Int32
    Public ReadOnly Property ImageHeight() As Int32
        Get
            Return _ImHeight
        End Get
    End Property
    Private _stride As Int32
    Public ReadOnly Property Stride() As Int32
        Get
            Return _stride
        End Get
    End Property
    Private _bytesPerPixel As Integer
    Public ReadOnly Property bytesPerPixel() As Integer
        Get
            Return _bytesPerPixel
        End Get
    End Property
    Private _colDepth As PixelFormat
    Public ReadOnly Property colorDepth As PixelFormat
        Get
            Return _colDepth
        End Get
    End Property

End Class
```

Nyilatkozat

Alulírott, …….. nyilatkozom, hogy jelen dolgozatom teljes egészében saját, önálló szellemi termékem. A dolgozatot sem részben, sem egészében semmilyen más felsőfokú oktatási vagy egyéb intézménybe nem nyújtottam be. A diplomamunkámban felhasznált, szerzői joggal védett anyagokra vonatkozó engedély a mellékletben megtalálható.

A témavezető által benyújtásra elfogadott diplomamunka PDF formátumban való elektronikus publikálásához a tanszéki honlapon

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2012. június 8.

…….
a hallgató aláírása