EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

Térképek georeferálása hiányos koordináták segítségével

DIPLOMAMUNKA
TÉRKÉPÉSZ MESTERSZAK

Készítette:
Zelenka Balázs

Témavezető:
dr. Kerkovits Krisztián András
adjunktus
ELTE Térképtudományi és Geoinformatikai Tanszék

Budapest, 2020.

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

TÉRKÉPTUDOMÁNYI ÉS GEOINFORMATIKAI TANSZÉK

DIPLOMAMUNKA-TÉMA BEJELENTŐ

Név: ZELENKA BALÁZS Neptun kód: BXFMAZ

Szak: **térképész MSc**

Témavezető neve: dr. Kerkovits Krisztián András

munkahelyének neve és címe: ELTE IK Térképtudományi és Geoinformatikai Tanszék

beosztása és iskolai végzettsége: adjunktus, PhD

A dolgozat címe: TÉRKÉPEK GEOREFERÁLÁSA HIÁNYOS KOORDINÁTÁK SEGÍTSÉGÉVEL

A dolgozat témája:

A régi, csak a kereten ívvonallal ellátott térképeket nem lehet hagyományos módon georeferálni. Cél a hiányos koordinátapárok ismeretlen értékeinek megközelítése legkisebb négyzetek módszerével, majd georeferálás az így kiszámolt koordinátapárokkal.

A témavezetést vállalom.

dr. Kerkovits Krisztián

Kérem a diplomamunka témájának jóváhagyását.

Budapest, 2019. december 1.

Zelenka B.

A diplomamunka-témát az Informatikai Kar jóváhagyta.

Budapest, 201 . december 1.

Zentai

Dr. Zentai László

tanszékvezető, egyetemi tanár

Tartalomjegyzék

1. Bevezetés

A térképek feladata a tárgyak és jelenségek térbeli elhelyezése úgy, hogy tájékozódásra, áttekintésre vagy leltározásra alkalmas legyen. Ebből egyenesen következik, hogy a térképlapon minden pontnak a valóságban is egy-egy pontra kell vonatkoznia. Tehát a térképlap elhelyezhető egy koordináta-rendszerben, amely a földfelszín kicsinyített, valós vagy vetületi egyenletetekkel torzított (síkba vetített) modellje. A modern térinformatikai szoftverek alapvetően ilyen koordináta-rendszerekkel működnek, minden objektum koordinátákkal van ellátva. Ahhoz, hogy a papírtérképek szkennelt, raszteres állományait betehessük ezen programokba, szükséges az egyes pixelek térbeli elhelyezése. Ezt a műveletet *georeferálásnak* nevezzük. A felhasználónak nem szükséges, hogy minden pixelt ellásson koordinátákkal, elegendő kellő mennyiségű ismert pontot elhelyezni, majd egy kiválasztott matematikai módszerrel a többi pixelt interpolációval kiszámítani. Ezeket a pontokat a koordinátahálózati vonalak metszésénél könnyű megadni. Metszéspontok hiányában ismert objektumokat adhatunk meg fixpontnak; azonban így növeljük a hibák kockázatát, ugyanis időben visszafele bizonytalanabb az objektumok helye és léte, illetve az objektumok térbeli kiterjedése sem feltétlen pontszerű. Olyan helyzet is előfordulhat, hogy csak a térképlap keretén található földrajzi vagy vetületi koordináta, csak egy tüske alakú őrvonal látszik, vagyis csupán egy tengely mentén ismert a koordináta. A térinformatikai szoftverek georeferáló ablakai elvárják, hogy koordinátapárokat adjunk meg; így jelenleg a fent említett probléma ezekkel a szoftverekkel még nem oldható meg. Célom, hogy erre egy QGIS-bővítménnyel szolgáljak megoldásként.

2. Térképek georeferálása

2.1. Georeferálás térinformatikai szoftverekben

A térinformatika a térben elhelyezkedő adatokkal foglalkozik, beleértve az adat bevitelét, tárolását, kezelését, elemzését, modellezését és megjelenítését. Mindezen adatok elhelyezhetők egy definiált koordináta-rendszerben, amelyben mérhetünk, számolhatunk, és más koordináta-rendszerbe átválthatunk. A térinformatikai szoftverek olyan fájlokkal dolgoznak, melyekben leírt térképi elemek geometriája és attribútumai összekapcsoltak, akár vektoros akár raszteres formában. Amely fájlok nem felelnek meg ennek, azt a szoftveren belül kapcsolhatjuk össze: egy táblázatot hozzáfűzhetünk attribútumok vagy koordinátapárok alapján, vagy térben elhelyezhetünk raszteres képeket, és ezáltal új állományt hozhatunk vele létre. A legismertebb bemeneti képformátumok a JPG, JPEG, PNG, ezekben a fájlokban színeket rendelünk az egyes pixelkoordinátákhoz, így tároljuk az egész képet.

A georeferálás során a felhasználó a grafikus felületen keres beazonosítható pontokat (koordinátahálózati metszéspontot vagy pontszerű objektumot) és a pontra kattintva megadja a koordinátáit egy előre meghatározott koordináta-rendszerben (Burt et al., 2020, 47. o.). A program a kép pixelkoordinátái és a megadott pont földrajzi vagy vetületi koordinátái között kapcsolatot létesít. Ezután a rendelkezésre álló pontok mennyiségéből következtetve választunk interpolációs módszert: lineáris, Helmert, polinom vagy spline transzformációt. A lineáris transzformáció a legegyszerűbb, elegendő legalább három pontot megadnunk, hogy a képünket síkban elhelyezzük. Több pont megadása esetén a transzformáció a legkisebb négyzetek módszerével közelíti a hibákat, hogy a lehető legkisebbre csökkentse azokat. A módszert később részletesen ismertetem.

A letett pontokat georeferálás előtt vagy után elmenthetjük egy *.points* kiterjesztésű GCP (ground control points) szöveges fájlba, ezt a lehetőséget kihasználom az általam készített programban. A georeferálást elindítva a program a kép minden pixeléhez koordinákat rendel, és létrehoz egy új, általában TIFF típusú fájlt.

2.2. Georeferálás őrvonalak alapján

Alapvető elvárás a térképektől, hogy tájékozhatók legyenek, távolságot, területet, irányt lehessen rajtuk mérni vagy legalább becsülni. A legtöbb modern térképen található fok- vagy kilométerhálózat, koordináták, északjel, mértéklécstb., ezért térben könnyen elhelyezhetők. Időben visszafele haladva ez nem mindig teljesül, egyes térképek matematikailag szépen levezetettek, mások egyáltalán nem. A térképi objektumok pontossága a felmérés pontosságán múlik, ez jelenti a távolság- és szögmérésből létrehozott rácsháló pontosságát, de az abszolút térbeli elhelyezéshez földrajzi koordináta-rendszerben elhelyezett pontokra van szükség, ezek meghatározása a műholdas technológia előtt csillagászati méréseket követelt. A 17-18. század során az európai nagyhatalmak topográfiai térképezésbe kezdtek, ehhez egyre pontosabban meghatározták a Föld alakját, és háromszögelési alappontokat hoztak létre. Ebből a korai térképezésből fellelhetők olyan topográfiai vagy levezetett térképek, ahol nincs fokháló, ahol földrajzi koordinátákat csak a kereten találhatunk, egy-egy rovátka vagy tüske jelzi a hosszúsági vagy szélességi kör értékét (Burt et al. 2020, 51. o.). Ezeket *őrvonalaknak*, hiányos koordinátáknak nevezzük. A térinformatikai szoftverekben szükséges koordinátapárt megadnunk, viszont őrvonalas térképek esetén előfordulhat, hogy a térképen egyetlen beazonosítható koordinátapárt sem találunk.

3. QGIS-bővítmény készítésének alapjai

3.1. A QGIS szoftver

A QGIS világszerte elterjedt, a legnépszerűbb nyílt forráskódú térinformatikai szoftver. 2002 óta folyamatosan fejlesztés alatt áll (2013-ig Quantum GIS néven), fut a legtöbb UNIX felületen, Windowson és MacOS-en egyaránt, sőt már Androidra is folynak a tesztelések. A programot C++ nyelven írták és a Qt függvénykönyvtárat használják, ezáltal biztosítva könnyen kezelhető grafikus felhasználói felületet (QGIS dokumentáció, 2020, 3. o.). A legtöbb használt raszteres és vektoros fájlformátumot ismeri és kezeli, és amit nem, azt a szorgos felhasználók adják hozzá bővítmény formájában az alapprogramhoz.

3.2. Plugin

A bővítmény (közismert angol nevén *plugin*) olyan programkód, amely nem fut önállóan, célja egy már meglévő program bővítése, új funkciók hozzáadása. Nagy és sokoldalú szoftverek fejlesztői engedik meg a felhasználóiknak hogy bővítményeket állítsanak elő, ugyanis egyes eszközökre nem merül fel akkora igény, hogy a fejlesztői csapat sok időt fordítson rá, másfelől az alapprogramot igyekeznek kis méretben összeállítani, hogy a széleskörűen nem használt eszköz ne foglaljon minden felhasználónál felesleges tárhelyet.

Meglepően hangzik, de a QGIS georeferáló eszköze (GDAL Georeferencer), maga is bővítmény, amit a fejlesztői csapat készített C++ nyelven (*core plugin*); viszont így is alapvető része a telepített programnak, csupán csak aktiválnia kell a felhasználónak, hogy használhassa (QGIS Dokumentáció, 583. o.). Érthető tehát, hogy a hiányos koordinátájú térképekhez a fejlesztői csapat nem is készített bővítményt, ugyanis ezek csak kis hányadát képezik a georeferálandó térképeknek. A külsős felhasználók python nyelven írhatnak bővítményt (External Plugin), amit a QGIS-profil *python/plugins* mappájába elhelyezve használhatnak. Amennyiben a plugin hasznos, stabil és jól dokumentált, a készítője feltöltheti a QGIS hivatalos tárhelyére, amely a moderátori ellenőrzésen átesve bárki által elérhető és használható lesz. (QGIS Dokumentáció, 563. o.)

Két QGIS-bővítményt említenék meg, amelyek az új pluginok készítéséhez hasznosak. A *Plugin Builder 3* az új plugin létrehozásában segít, automatikusan létrehozza a szükséges fájlokat és kitölti a fő függvényeket. A *Plugin Reloader* pedig QGIS-en belül frissíti a kiválasztott bővítményt, így nem szükséges a QGIS újraindítása, ha módosítottunk valamit a bővítményünk forráskódjában.

3.3. Python programozási nyelv

A Python egy általános célú és magas szintű, interpreteres (a futtatás közvetlenül a forráskódból történik) programozási nyelv. Guido van Rossum holland programozó kezdte el fejleszteni az 1990-es években, sok ötletet merítve az ABC nyelvből, amelyen ő maga is dolgozott (Python dokumentáció; History and Licence, 2020). A *Zen of Python* című angol nyelvű összefoglalás röviden vázolja a Python célját (amely az *import this* parancssal elérhető bármelyik Python konzolon). E szerint a forráskód egyszerűsége és olvashatósága előbbrevaló, mint a program futásának gyorsasága. A nyelv könnyen olvasható és értelmezhető, egyszerű angol kulcsszavakat használ. Hiányoznak belőle a C nyelvekből ismert blokkok lehatárolására használt *{* és *;* karakterek, helyette a tabulátorokkal tördel, ami egyszerűen megköveteli a program írójától hogy átláthatóan dolgozzon. A Pythonban osztályokat és függvényeket hozhatunk létre, az alárendelt osztályokat ponttal kapcsoljuk össze a fölérendelttel, a függvényeket is ponttal kapcsoljuk ezekhez, a változókat pedig zárójel közé zárva soroljuk fel a függvény neve után. A logikai kapcsolatokat kulcsszavakkal írjuk, a *for* ciklus pedig alapvetőnek veszi, hogy egy lista elemein egyesével végigmegyünk, hacsak máshogy nem rendelkezünk: ez egy sok pontot tartalmazó lista (több dimenziós mátrix) esetében rendkívül előnyös. A nyelv ismeri a *string* (szöveg) típust, és könnyű az átalakíthatóság a szöveg és a számok típusai között, ami megkönnyíti az adatbevitelt és kiírást. A Python meglehetősen nagy alapfüggvénytárral rendelkezik, ehhez társul még sok publikus könyvtár, amelyek elemeit az *import* paranccsal hívhatjuk meg. A Python nyelven írt QGIS-bővítmények írásának első eleme a *qgis* könyvtár meghívása (Garrand, 15-34. o., 2016).

Nem áll módomban a nyelvet bővebben ismertetni, mivel csak egy bizonyos részére támaszkodtam, mindenekelőtt a QGIS könyvtárát felhasználva. Az egyes Python adta megközelítéseket és esetleges előnyöket az általam írt program egyes lépéseinél ismertetem.

3.4. Qt könyvtár

A Qt egy C++ nyelven írt grafikus eszköztár (widget toolkit), amellyel grafikus felhasználói felületet hozhatunk létre előre megírt vezérlőelemek (widgetek) segítségével (Qt Dokumentáció). Az elkészítendő bővítménynek az illesztőpontok lerakása miatt feltétlenül szükséges grafikai felülettel rendelkeznie, és mivel a QGIS grafikus motorja a Qt, így a plugint is abban készítem. Feltétlenül előnyös, ha a bővítmény a lehető legjobban hasonlít a QGIS georeferáló ablakhoz, mert szorosan összekapcsolódik vele. A Qt Python és QML nyelven is használható, a programomban a legfrissebb PyQt5 könyvtárat használom.

4. Az implementálás folyamata

A diplomamunka célkitűzése tehát egy olyan program (tulajdonképpen egy QGIS-bővítmény), amely beolvas egy képfájlt, megjeleníti azt, továbbá a felhasználó fixpontokat tehet le és láthat el koordinátákkal, illetve megadhat egy fájlnevet és elérési útvonalat a mentendő GCP-fájlnak. A bővítmény maga nem georeferál, de mindent előkészít ahhoz, hogy a felhasználó kényelmesen és rövid időn belül a megfelelő fájlok és interpolációs módszer kiválasztásával a QGIS Georeferencer pluginban georeferálhasson.

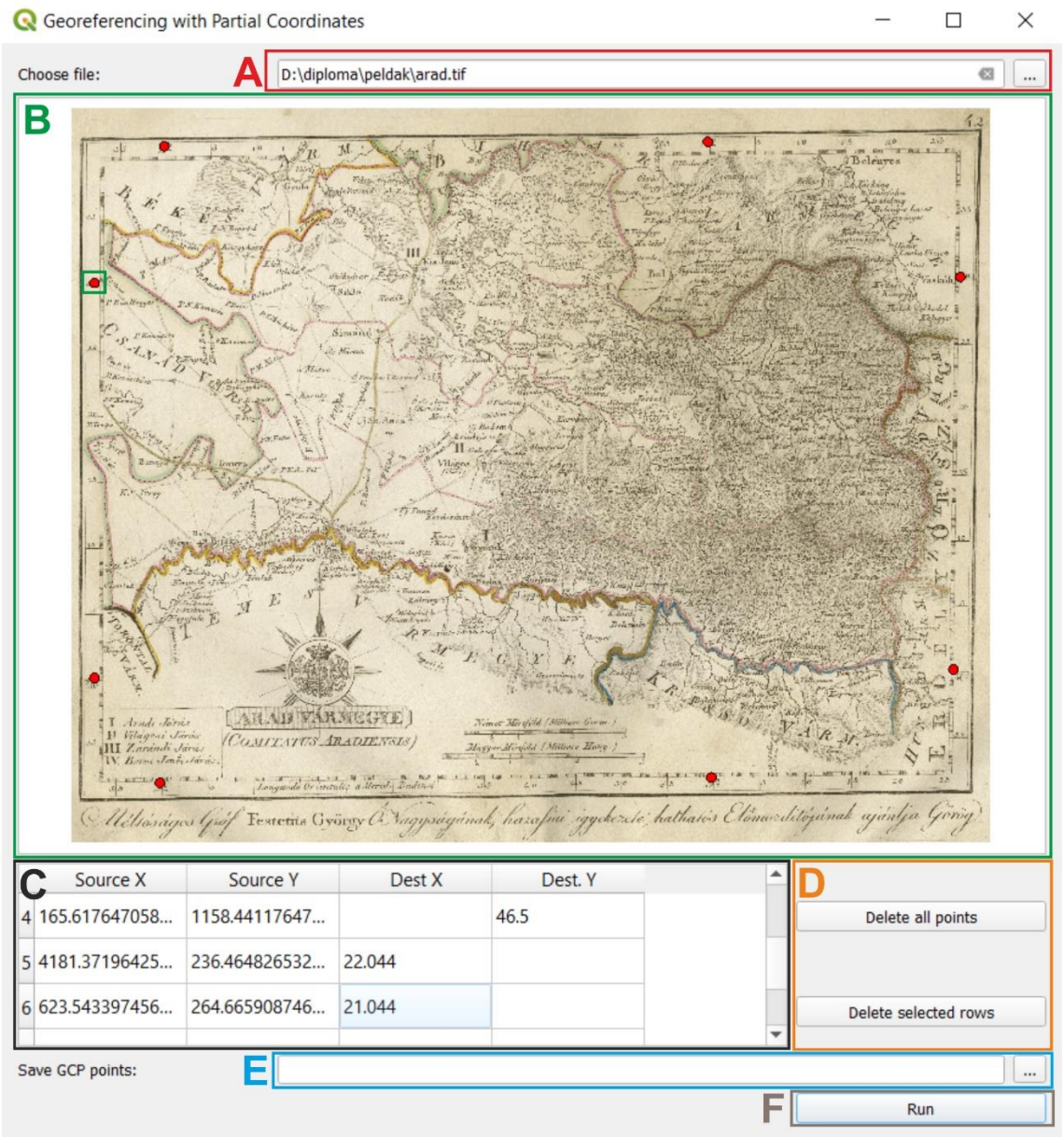
További célkitűzés, hogy ez a program mások által felhasználható és széles körben terjeszthető legyen, így elsősorban angol nyelven fejlesztettem. Ez nem csak a névválasztásra, a különböző feliratok, gombok és üzenetek angol nyelvű használatára vonatkozik, de még a forráskód változóit és függvényeit is angolul neveztem el, hogy bárki más továbbfejleszthesse.

Első lépésként létrehoztam a bővítményt *Georeferencing with partial coordinates* néven a Plugin Builder segítségével, második lépésként pedig megterveztem a felhasználói felületet.

4.1. Felhasználói felület

A program grafikus felhasználói felülettel (GUI) rendelkezik, amelyet a QGIS-szel együtt települt *Qt Designer* alkalmazással készítettem (pontosabban a *Qt Designer with QGIS 3.8.2 custom widgets* verzióval). Az elemek rugalmasan illeszkednek egymáshoz, az ablak magassága és szélessége tetszőlegesen változtatható, valamint megjeleníthető teljes ablakos nézetben, illetve letehető tálcára. A bővítménybe betöltött információk nem vesznek el az ablak bezárásával, csupán a QGIS bezárásával.

A. Fájlkiválasztó elem (*QGSFileWidget*), amellyel a betöltendő képfájlt kiválasztható.
B. Képmegjelenítő (*QGraphicsView*), amely alaphelyzetben üres, de fájlt kiválasztva a kép kitölti a rendelkezésre álló teret. Ha a megjelenítő tartalmaz képet, lehetőség van illesztőpont elhelyezésére (piros pontként jelenik meg).
C. Táblázat, amely a lerakott pontok koordinátáit jeleníti meg (*QTableWidget*)
D. Két nyomógomb a pontok törlésére (összesre vagy a táblázatban kijelöltekre).
E. Fájlkiválasztó elem (*QGSFileWidget*), amellyel a mentendő GCP-fájl nevét és elérési útvonalát adhatjuk meg.
F. A program futtatását elindító nyomógomb.



1. ábra: a program felhasználói felülete

4.1.1. Fájlbeolvasás

A program első eleme a fájlbeolvasó modul, enélkül a képmegjelenítő üres, és nem lehet pontot elhelyezni. A vezérlőelemre kattintva az operációs rendszernek megfelelő fájlkiválasztó ablak ugrik fel, ahol csak egy fájlt enged kiválasztani. A program csak JPG, JPEG, PNG és TIFF típusú fájlt enged betölteni.

```
self.chooseFile.setFilter('*.jpg;*.jpeg;*.png;*.tif;*.tiff')
self.chooseFile.fileChanged.connect(self.openImage)
```

A megnyitást választva a program betölti a képet a megjelenítőbe.

```
scene = QtWidgets.QGraphicsScene()
self.scene=scene
self.gv.setScene(self.scene)
p=QtGui.QPixmap()
self.pixmapItem=scene.addPixmap(p)
```

4.1.2. Illesztőpont-elhelyezés

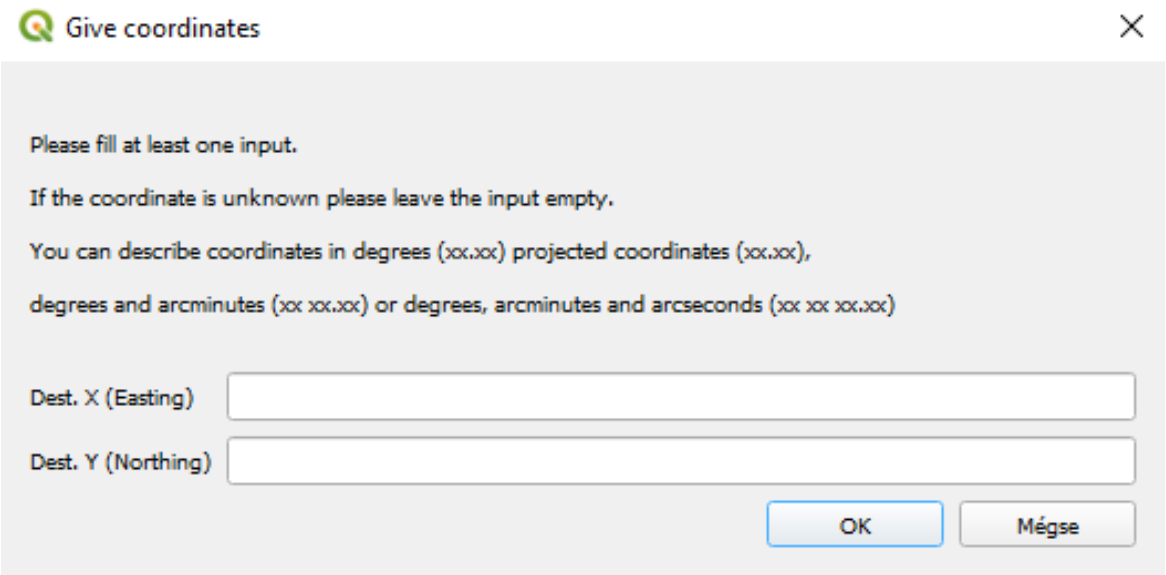
A kép betöltését követően a felhasználó az egérgörgővel szabadon nagyíthat/kicsinyíthet a képre (zoom), illetve az egér bal gombját lenyomva tartva mozgathatja azt (pan). A képmegjelenítőben való bal egérgomb lenyomását, mozgatását, felengedését valamint a görgőt külön-külön függvények figyelik összehangolva, tárolva az események pozícióit, és lehetővé téve a fent említett műveletek végrehajtását. Ha a kattintás során a felhasználó az egeret nem mozgatta el, úgy pontot rak le, vagyis rögtön felugrik egy új, kisebb méretű ablak, amit a forráskódban *SubDialog* néven hoztam létre. A kisebbik ablakban két szövegdobozban a két koordináta tölthető ki (Easting és Northing, vagyis a keleti és az északi tengely). Ha az egyik tengely koordinátája ismeretlen, üresen kell hagyni.

A kivételkezelést, vagyis a különböző hibák és az ebből fakadó nemkívánt programösszeomlás kivédését már ezeknél a szövegdobozoknál elkezdtem. Több programnyelvhez hasonlóan a Pythonhoz csatolt Qt könyvtár is rendelkezik RegExp (*regular expression*) validátorral, amellyel szabályokhoz köthetjük a szövegdobozok kitöltését. A koordináták esetében ez a negatív előjel, 0–9 számjegyek, és a tizedesvessző; ez lehet pont, vagy a közép-európai kultúrkörben elterjedt vessző karakter, amelyet

a program kitöltés után pontra cserél (Goyvaerts, 2007). Ugyancsak írható maximum két esetben szóköz karakter a számjegyek közé, amellyel a felhasználó fok-perc, illetve fok-perc-másodperc formátumban viheti be a koordinátát, amelyet a program rögtön átszámol fokra és úgy jeleníti meg a táblázatban. A különböző formátumok szerinti kitöltési lehetőségek olvashatók a felugró dialógusablak magyarázatában.

```
FORM_CLASS, _ = uic.loadUiType(os.path.join(
    os.path.dirname(__file__),
'partial_coordinates_dialog_coord.ui'))

class SubDialog(QtWidgets.QDialog, FORM_CLASS):
    def __init__(self, parent=None):
        super(SubDialog, self).__init__(parent)
        self.setupUi(self)
        regex = QRegExp("(-?(0|[1-9][0-9]*)([\\.,]
[0-9]+|\\s[0-9][0-9]?([\\.,][0-9]+|\\s[0-9][0-9]?
([\\.,][0-9]+)?)?)?)?")
        validator = QtGui.QRegExpValidator(regex)
        self.destX.setValidator(validator)
        self.destY.setValidator(validator)
```



2. ábra: felugró ablak a koordináták kitöltéséhez

A szövegdobozok kitöltése után az OK gombot lenyomva a felugró ablak bezárul, és a képmegjelenítőben a kattintás helyére egy *QGraphicsItem* objektum kerül, amely piros pontként jelenik meg (ebben is törekedtem a Georeferencerhez való hasonlatosságra), valamint a képi és megadott koordinátákkal egy új sorral bővíti a táblázatot. Emellett az objektum listázódik az *itemArray* nevű tömbben, ami a későbbi ponttörlési műveletekhez szükséges. Az illesztőpontok kezeléséhez külön osztályt (class) hoztam létre, amely kezeli a pontok grafikai megjelenését; így a pont mindig helyben marad, és nagyítási szinttől függetlenül megtartja méretét.

4.1.3. Illesztőpontok táblázata

Az illesztőpontok táblázata nagyban hasonlít a Georeferencer táblázatához, az oszlopok elnevezései megegyeznek: a képpontok koordinátái *SourceX* és *SourceY*, a megadott koordinátáké pedig *DestX* és *DestY*. A táblázat nem csupán dokumentációként szolgál a felhasználó számára, hanem lehetőséget ad arra, hogy utólag szerkeszthesse a megadott koordinátákat. Ha a felhasználó elgépelte, rossz tengelyre írta be, vagy felcserélt koordinátákat, a pontok törlése nélkül szerkesztheti őket, és mindezt egy RegExp validátor védi (itt már csak pont karaktert enged tizedesvesszőnek, valamint nem lehetséges szögpercet és szögmásodpercet megadni). A felhasználónak nincs lehetősége arra, hogy az illesztőpontok képkoordinátáit szerkessze, viszont lehetősége van a nem kívánt pontok törlésére.

```
self.tableWidget.insertRow(self.tableWidget.rowCount())
item = QtWidgets.QTableWidgetItem(eX)
```

A táblázat melletti *Delete all points* gommbal a felhasználó kitörli az eddig letett pontokat, törli a táblázat sorait, és képen lehelyezett piros pontokat, viszont magát a betöltött képet bent hagyja a megjelenítőben. Új kép betöltése esetén a táblázat és a pontok szintén törlésre kerülnek.

```
def deleteAll(self):
   self.tableWidget.setRowCount(0)
   for i in self.scene.items():
      if i != self.pixmapItem:
```

```
            self.scene.removeItem(i)
    self.itemArray = []
```

Ha az alatta levő *Delete selected rows*-ra kattintunk, akkor azok a sorok (és a hozzájuk tartozó itemek) törlődnek, amelyek a táblázatban ki vannak jelölve. Nem szükséges az egész sort, elegendő bármelyik elemét kijelölni. A törlések nem visszavonható műveletek.

```
def deleteRows(self):
    rows = set()
    for index in self.tableWidget.selectedIndexes():
        rows.add(index.row())
    for row in sorted(rows, reverse=True):
        self.tableWidget.removeRow(row)
        self.scene.removeItem(self.itemArray[row])
        del self.itemArray[row]
```

4.1.4. Kivételkezelések

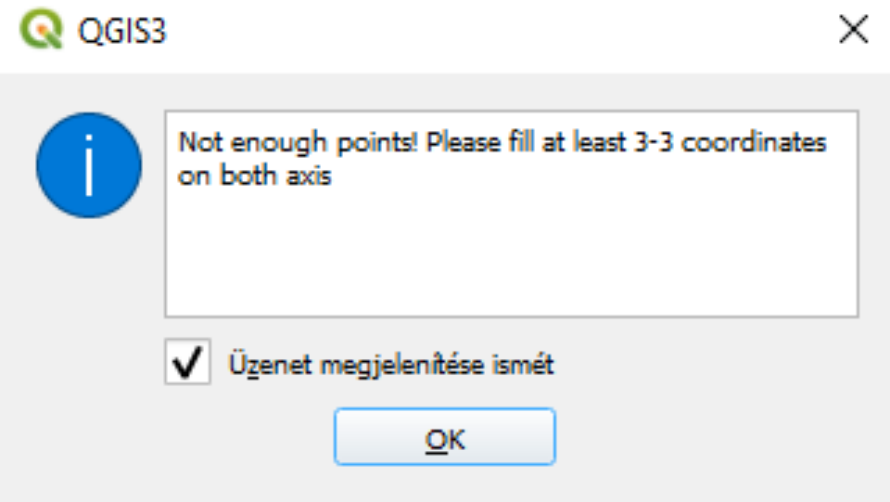
A programfutás során fellépő hibák kivédése az egyik legfontosabb feladat. Problémát okozhatnak a matematikai hibák vagy a felhasználó okozta hibák, például egyes szükséges lépések kihagyása. Ezek kivédése két módon oldható meg: vagy megakadályozzuk a hibás művelet megtételét, vagy felugró ablakkal figyelmeztetjük a felhasználót, és a program visszaugrik a hiba előtti állapotra. Az első esetre példa a RegExp-pel korlátolt szövegbevitel, vagy a pontlerakás akadályozása addig, amíg nincs képfájl betöltve. Másodikra jó példa, hogy a program elvár legalább 3-3 x' és y' megadott félkoordinátát (ennek a matematikai okait később részletezem), de ez már a felhasználó számára nem magától értetődő, nem célszerű egyszerűen akadályoztatni, hogy a futtatást elindítsa. Ehelyett a futtatás gomb megnyomása után egy felugró ablak figyelmezteti a matematikai követelményekre. Ehhez a kis ablakhoz nem szükséges egy teljesen új párbeszédablakot létrehozni, a *QtWidgets* könyvtárban ehhez adott egy *QErrorMessage* osztály. Ezt az osztályt elég egyszer az *_init_()* függvényben definiálni, és későbbiekben többször felhasználható.

```
class PartialCoordinatesDialog(QtWidgets.QDialog,
FORM_CLASS):
    def __init__(self, parent=None):
```

```
        self.error = QtWidgets.QErrorMessage()
...
        if len(dest_x) < 3 or len(dest_y) < 3:
            self.error.showMessage('Not enough points! Please
fill at least 3-3 coordinates on both axis!')
```

Ugyancsak probléma, ha a felhasználó nem ad meg nevet és elérési útvonalat a mentendő GCP-fájlnak, és nyom rá a futtatásra, ezért erre szintén egy felugró ablak figyelmezteti.

```
self.error.showMessage('No filename to save!')
```



3. ábra: felugró hibaüzenet-ablak

4.2. Futtatás

Mikor a felhasználó úgy véli, hogy elegendő, jól elhelyezett pontot rakott le, akkor rákattint a *Run* feliratú gombra, és elindítja a program fő metódusát. A *run()* függvény először kiszedi az összes értéket a *QTableWidget* típusú táblázatból, és bemásolja egy kétdimenziós tömbbe, az üres stringeket *NULL* értékre cserélve. Mindez egy sorral megoldható a forráskódban a *lambdafüggvény* segítségével. A lambdafüggvény egysoros, egyszer használatos függvény, vagyis használat után már törlődik is.

```
self.gcps = [[(lambda s: None if s=='' else float(s))
(self.tableWidget.item(y,x).text()) for x in range(t_col)]
for y in range(t_row)]
```

A tömb létrehozása után a sorok szétválogatásra kerülnek. Azon pontok képkoordinátái, amelyek rendelkeznek megadott x' vetületi koordinátával, tárolásra kerülnek a *src_x* nevű kétdimenziós mátrixban, a hozzájuk tartozó x'koordináták pedig egy *dest_x* nevű egysoros mátrixban. Mindez megtörténik az y' koordinátának megfelelően is. Így duplikálódnak azok a pontok, amelyhez mindkét koordináta ismert, és kiesnek azok a pontok, amelyeket üresen adott meg a felhasználó. Mihelyst készen állnak a mátrixok, elindul a lineáris transzformáció.

4.2.1. Legkisebb négyzetek módszere

A lineáris transzformáció alapja egy közelítés, a cél hogy a rendelkezésre álló adatok alapján a hiányzó adatokat megbecsüljük, úgy hogy a maradék ellentmondások négyzetösszege minimális legyen. Ezt a legkisebb négyzetek módszerének nevezzük (Sárközy, 1987). Ennek az eredménye két függvény, a két tengelynek megfelelően. Jelölje x, y a képkoordinátákat, x', y' a vetületi koordinátákat, és $\hat{x}, \hat{y}$ a becsült vetületi koordinátákat. Továbbiakban csak *x*-re írom fel az összefüggéseket, mivel ugyanez értendő *y*-ra is. A vetületi koordinátákat a képkoordináták sima függvényének tekintjük, így az tetszőleges pontossággal közelíthető a következő hatványsorral:

$$\hat{x} = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + \cdots = \sum_{j=0}^{n}\sum_{k=0}^{n-j} a_{jk}x^j y^k$$

Az a_{jk} értékek ismeretlen együtthatók. Adott *i* darab ismert pontunk, mindegyikhez tartozik egy valós vetületi koordintáta, és keressük a lehető legkisebb maradék ellentmondást, a hibákat normális eloszlásúnak feltételezve.

$$\sum_i (\hat{x}_i - x'_i)^2 \rightarrow \min$$

A kifejezés csak akkor minimális, ha a parciális deriváltja zérus.

$$\frac{\partial \sum_i (\hat{x}_i - x'_i)^2}{\partial a_{jk}} = 0$$

Az alábbi összefüggést felismerve elvégezhetjük a parciális deriválást.

$$\frac{\partial \hat{x}}{\partial a_{jk}} = x^j y^k$$

$$\frac{\partial \sum_i \left(\hat{x}_i{}^2 - 2\hat{x}_i x'_i + x'_i{}^2\right)}{\partial a_{jk}} = \sum_i \left(2\hat{x}_i x_i{}^j y_i{}^k - 2x'_i x_i{}^j y_i{}^k\right) = \sum_i x_i{}^j y_i{}^k (\hat{x}_i - x'_i) = 0$$

Az egyenletet átrendezve, majd az $\hat{x}_i$-t kifejezve:

$$\sum_i \hat{x}_i x_i{}^j y_i{}^k = \sum_i x'_i x_i{}^j y_i{}^k$$

$$\sum_i x_i{}^j y_i{}^k (a_{00} + a_{10} x_i + a_{01} y_i + \cdots) = \sum_i x'_i x_i{}^j y_i{}^k$$

Az ismeretleneket a szummák elé kiemelve kapjuk:

$$a_{00} \sum_i x_i{}^j y_i{}^k + a_{10} \sum_i x_i{}^{j+1} y_i{}^k + a_{01} \sum_i x_i{}^j y_i{}^{k+1} + \ldots = \sum_i x'_i x_i{}^j y_i{}^k$$

A végtelen sor összes ismeretlenének kiszámításához végtelen sok ismert pontra van szükségünk. Ez a feladat szempontjából lehetetlen megoldás, határt kell húzni. Ezt a határt több szempont határozza meg: nem szeretnénk, hogy a polinomfüggvényt a pontok hibái látványosan „rángassák”, és szeretnénk kevés illesztőponttal is használni (lineáris transzformáció esetén legalább 3, másodfokú esetén legalább 6 pont szükséges). Lineáris transzformációt használva az első három tagra van szükségünk. Mivel $n = 1$ ezért j és k értékeknél három eset lép fel, ami az alábbi három egyenletet eredményezi:

$$a_{00} \sum_i 1 + a_{10} \sum_i x_i + a_{01} \sum_i y_i = \sum_i x'_i$$

$$a_{00} \sum_i x_i + a_{10} \sum_i x_i{}^2 + a_{01} \sum_i x_i y_i = \sum_i x'_i x_i$$

$$a_{00} \sum_i y_i + a_{10} \sum_i x_i y_i + a_{01} \sum_i y_i{}^2 = \sum_i x'_i y_i$$

Ez egy három ismeretlenből és három egyenletből álló egyenletrendszer, tehát lehet egzakt megoldása, az ismeretleneket a Gauss–Jordan-elimináció segítségével számolhatjuk ki.

4.2.2. Gauss–Jordan-elimináció

A Gauss–Jordan-eliminációt *n* ismeretlenes *n* egyenletből álló egyenletrendszernél alkalmazhatjuk, ahol az ismeretlenek együtthatóit egy *n*×*n*-es mátrixba rendezzük, oszloponként sorban a megfelelő ismeretlenek együtthatóival (együtthatómátrix), majd kibővítjük az egyenletek konstans tagjaiból alkotott oszlopmátrix-szal (eredménymátrix). Ezután a sorok lineáris kombinációjával, összeadásával, kivonásával, skalárral való szorzásával és akár a sorok cseréjével addig kombinálunk, amíg az együttható-mátrixban a főátlóban 1, a többi helyen 0 érték áll. Ez esetben az eredménymátrix értékei sorrendben a keresett ismeretlenek értékei. A mi esetünkben a kezdőállapot így néz ki:

$$\begin{bmatrix} \sum_i 1 & \sum_i x_i & \sum_i y_i \\ \sum_i x_i & \sum_i {x_i}^2 & \sum_i x_i y_i \\ \sum_i y_i & \sum_i x_i y_i & \sum_i {y_i}^2 \end{bmatrix} \cdot \begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \end{bmatrix} = \begin{bmatrix} \sum_i {x'}_i \\ \sum_i {x'}_i x_i \\ \sum_i {x'}_i y_i \end{bmatrix}$$

Fontos megjegyezni, hogy az együtthatómátrix determinánsa nem lehet zérus, különben az együtthatók lineárisan nem függetlenek, így nem számolhatók ki. Ha pedig az érték a nulla szűk környezetébe esik, úgy a hibákat felnagyítja. Ez elkerülhető, ha a futtatás során a program ellenőrzi, hogy az együtthatómátrix determinánsának abszolútértéke kisebb-e, mint 10^{-4}. A determináns kiszámolását külső függvényre bíztam, amelyet a *run()* függvényben hív meg az alábbi feltétel:

```
if abs(self.determ(coeff_x))<0.0001 or
abs(self.determ(coeff_y))<0.0001:
     self.error.showMessage('The solution could not be
determined. Please try allocate your points to not fall on
one straight line')
```

A feltétel akkor teljesül, ha a determináns közel nulla, ami gyakorlatban azt jelenti, hogy a megadott illesztőpontok közel egy egyenesre esnek. Így a program futása megáll, és egy hibaüzenet arra kéri a felhasználót, hogy pontjait jobban szétszórva helyezze le. Ha a feltétel nem teljesül, a program tovább fut és végrehajtja a Gauss–Jordan-eliminációt. A művelethez az alapszakos szakdolgozatomban Java nyelven írt kódot írtam át Python nyelvre, melyet *gauss_jordan()* néven tároltam el. Ezután létrehoztam az együtthatómátrixot és az eredménymátrixot. Az eredménymátrix feltöltéséhez nem írtam meg kézzel mind a kilenc összegzést, főleg hogy csak hat különbözőre lenne szükség, hanem készítettem egy *sum_xy()* nevű függvényt ami egy *i* elemű tömbből és *j* és *k* bemeneti értékből kiszámolja az alábbi szummát:

$$\sum_i {x_i}^j {y_i}^k$$

Az előkészített mátrixok forráskódja tehát így néz ki (továbbra is csak *x*-et hozva példának):

```
coeff_x = [[self.sum_xy(src_x, 0, 0),
    self.sum_xy(src_x, 1, 0),self.sum_xy(src_x, 0, 1)],
    [self.sum_xy(src_x, 1, 0),self.sum_xy(src_x, 2, 0),
    self.sum_xy(src_x, 1, 1)],[self.sum_xy(src_x, 0, 1),
    self.sum_xy(src_x, 1, 1),self.sum_xy(src_x, 0, 2)]]
result_x = [sum(dest_x),
    sum(src[1]*n for src, n in zip(src_x, dest_x)),
    sum(src[0]*n for src, n in zip(src_x, dest_x))]
var_x = self.gauss_jordan(coeff_x,result_x)
```

A kapott eredmények az együtthatók, amellyel kiszámolhatók az ismeretlen értékek. Egy *for* ciklus végigmegy a *self.gcps* tömbön, és ahol NULL értéket talál, ott kiszámítja a pixelkoordinátákból és az együtthatókból a hiányzó koordinátapárt.

$$x' = a_{00} + a_{10}x + a_{01}y \text{ és } y' = b_{00} + b_{10}x + b_{01}y$$

Ami a forráskódban:

```
for p in self.gcps:
    if p[2] is None:
```

```
        p[2] = var_x[0] + var_x[1] * p[1] + var_x[2] * p[0]
    if p[3] is None:
        p[3] = var_y[0] + var_y[1] * p[1] + var_y[2] * p[0]
```

4.3. Fájlkiírás

Amint elkészültek a koordinátapárjaink, a térkép tulajdonképpen georeferálható. Ahhoz, hogy a Georeferencerbe ne kelljen ezeket a pontokat kézzel bevinni, ki kell mentenünk egy GCP fájlba, amit a Georeferencer értelmezni képes. A dialógusablak alján található a fájl mentéséhez szükséges *QGSFileWidget*, amire kattintva felugrik egy fájlkiválasztó ablak, ez esetben a mentéshez. Ezt a widget *StorageMode* attribútumának átállításával lehet elérni az *_init_()* függvényben.

```
self.saveFile.setFilter("GCP file | *.points")
self.saveFile.setStorageMode(qgis.gui.QgsFileWidget.SaveFile)
```

A futtatás során a *self.gcps* tömb kiszámolása után a program megkísérli létrehozni, majd megnyitni a mentendő fájlt. Egy *try-except* kivételkezelés védi a programot, vagyis ha nem sikerül létrehozni a fájlt, akkor a program az *except*-ben szereplő utasítást hajtja végre, ebben az esetben felugró ablakban hibaüzenetet küld a felhasználónak. Ha sikerül létrehozni a fájlt, a *with* utasítással megnyitja, egy ciklussal soronként leírja az adatokat, majd lezárja.

```
try:
    with open(self.saveFile.filePath(),"w") as f:
        f.write("
mapX,mapY,pixelX,pixelY,enable,dX,dY,residual\n")
        for p in self.gcps:
            f.write(str(p[2])+","+str(p[3])+","+str(p[0])+"
,"+str(p[1])+",1,0,0,0\n")
        self.close()
except:
    self.error.showMessage('No filename to save!')
```

Ha sikeres volt a futtatás, a program egy szalagüzenettel jelzi ezt a dialógusablak megjelenítőjében. Felhasználói tesztelésben résztvevők javaslatait figyelembe véve a program sikeres futtatást követően nem zárul be, így biztosítva, hogy az első próba után több pontot lehessen hozzáadni pontosítva az eredményt.

4.3.1. GCP-fájlformátum a QGIS szoftverben.

A *.points* kiterjesztésű GCP-fájl egy szöveges állomány, az első sora a mezők neveit tartalmazza vesszővel tagolva, a további sorokban a mezőkhöz tartozó értékeket tárolja, ugyancsak vesszővel tagolva (Garrard, 212. o., 2016). Ezt a fájlformátumot a QGIS értelmezi, és nem egyezik meg a *.gcp* kiterjesztésű fájlformátummal, amelyet a Global Mapper nevű közismert térinformatikai szoftver használ szintén illesztőpontok tárolására. A Georeferencer esetében a mezők a következők: megadott vetületi *x* és *y* koordináta, *x* és *y* pixelkoordináta, láthatóság, *x* és *y* tengely szerinti hiba és javítás. A programom az első négy elemet a *self.gcps* tömbből kitölti, a maradék négyet pedig 0 értéken hagyja, mivel ezeket a Georeferencer tölti ki.

4.4. Georeferálás

A felhasználó utolsó feladata megnyitni a QGIS Georeferencert, betölteni a megfelelő rasztert, betölteni a mentett GCP pontokat, kiválasztani a vetületet amelyben megadta a koordinátákat, és beállítani a transzformáció típusát. A georeferálás futtatása után a szoftverben meggyőződhetünk annak sikerességéről.

5. A program tesztelése

Minden programozó feladata a program folyamatos tesztelése, vagyis annak rendszeres vizsgálata, hogy ellátja-e rendeltetésszerű feladatait. A legfontosabb természetesen, hogy a program lehető legjobban kiszámolja a GCP-pontokat, de ugyanannyira fontos a grafikus felület kezelhetősége, egyszerűsége, teljessége. A matematikai eljárások teszteléshez eleinte egy egyszerű négyzetrácsos kép is elegendő, de az eredmények felmutatásához már olyan alapanyagra van szükség, amely a program hatásossága mellett annak szükségességét is mutatja, vagyis egy olyan térképre, amely kizárólag őrvonalakból áll.

5.1. Alapanyag

Erre jó példaként, és egyszerű tesztelhető alapanyagként szolgál Görög Demeter és Kerekes Sámuel *Magyar Átlás* c. vármegyetérkép gyűjteménye (továbbiakban Görög–Kerekes atlasz), amely 1792 és 1811 között készült, az első katonai felmérés és csillagászati mérések felhasználásával (Farkas, 2018). Egyes térképlapokat települések beazonosításával Huszánk Daniella (2019) georeferálta, én viszont ezeket a térképeket fokhálózat alapján georeferálom, így mérhető lesz a települések földrajzi pontatlansága. Az összes megyetérkép őrvonalas, földrajzi koordinátarendszert használ gellérthegyi középmeridiánnal (19° 3’ 12,8”; Timár, 2004). A térképlapokon nem található egyetlen koordinátapár sem, ugyanis a keret sarokpontjai nem esnek egész fokokra sem fokpercekre. Ugyanakkor a keret teljes terjedelmében szerepelnek őrvonalak, így nem kell aggódni, hogy a determináns túl kicsi lenne a transzformációhoz. Példaterületnek Pest-Pilis-Solt és Arad vármegyét használtam. Mindkettő arányaiban nagy kiterjedésű, egyik észak–déli, másik kelet–nyugati irányban elnyúlt. A georeferálás során figyelni kell a gellérthegyi és greenwich-i meridiánok közti átváltásra, valamint arra, hogy a kornak megfelelő ellipszoidot használjunk, lehetséges megoldás a Zách–Oriáni-ellipszoid. Ehhez a QGIS beállításoknál az *egyéni vetületek* menüpontban az alábbi paramétereket kell megadni (Timár, 2004):

```
+proj=longlat +a=6376130 +rf=310 +towgs84=1764,283,569
+pm=19.05355556
```

Ezt a vetületet a georeferált réteg forrásánál kell beállítani.

4. ábra: részlet a georeferált Görög–Kerekes-féle Pest-Pilis-Solt vármegye térképről, OpenStreetMap háttértérképpel vegyítve. Látható, hogy a kelet–nyugati irányú hibák nagyobbak, mint az észak–déli irányúak, ami a kor mérési technológiájából adódik.

5.2. Felhasználói vélemények

Amint a plugin alkalmas volt a pontos georeferálásra, felkértem térképész oktatókat és hallgatókat a program felhasználói tesztelésére, hiszen elsősorban nekik készül. A program készítője kihagyhat olyan funkciókat, amelyeket más alapvetőnek venne; könnyen megtalál olyan elemeket, mely más felhasználónak nehézséget okozna. Fontos a helyes kommunikáció az új felhasználó felé, hogy hamar megértse és elsajátítsa a bővítmény működését; és megértse a hibaüzeneteket, ha nem megfelelően használja. Továbbiakban ismertetek pár tesztelés során kapott tanácsot és ötletet, leírom hogy sikerült-e implementálni, és az okait akár igen akár nem.

Egy fontos javaslat volt, hogy a programból meg lehessen nyitni a Georeferencert, automatikusan betöltve a képfájlt és az illesztőpontokat. Ez sajnos a QGIS jelenlegi állapotában nem lehetséges, egyrészt mivel a Georeferencer is plugin, másrészt C++-ban írták meg és nem lehet a függvényeit Pythonból meghívni. A programba alapvető georeferáló funkciót írni pedig meglehetősen időigényes és véleményem szerint felesleges, mivel a Georeferencer kiválóan használható. Szintén javasolták, hogy az illesztőpontok lerakása során jelenjenek meg becsült fokhálózati vonalak, ezzel is segítve a további pontok lerakását, viszont mivel a program nem kezeli a vetületeket, és nem csak földrajzi koordináták adhatók meg, hanem méter alapú koordináták is, a segédvonalak beosztásának kiszámítása túlságosan nagy munkát igényelt volna, amit már nem tudtam volna időben megvalósítani.

Ugyanakkor sok egyszerűbben megvalósítható praktikus elem bekerült a programba, mint például a fok-perc-másodperc formátumú koordináta-megadás lehetősége vagy a lenyomott bal egérgombbal való képmozgatás, illetve a csak kijelölt sorok törlési lehetősége. Emellett sok apró hibára (közkedvelt nevén: *bug*) is felhívták a figyelmemet, mint például a kép megváltoztatásnál az illesztőpontok bennragadtak vagy hogy a fok-perc-másodperc átváltása negatív számok esetén hibás volt, de olyan banális hibát is elkövettem, hogy eleinte nem hívtam fel a felhasználó figyelmét megadott koordináták szükséges minimális mennyiségére. Összeségében véve kijelenthető, hogy a minél nagyobb és képzettebb közösség nézi át a készülő programot, annál több hibát lehet kiszűrni és közkedvelt funkciót beleépíteni a programba.

6. Eredmények összefoglalása

A legfontosabb célkitűzést sikerült elérnem: elkészíteni egy olyan programot, amely lehetővé teszi hiányos koordinátájú térképek georeferálását. Pontosabban véve a program végterméke egy olyan fájl, amely biztosítja a képfájl hagyományos georeferálását egy másik programban. Sikerült kényelmesen kezelhető, laikusok által is könnyen használható grafikus felületet létrehoznom. A programot egy olyan szoftver bővítményeként hoztam létre, amely ingyenes és világszerte ismert. A program nyelve, valamint a forráskód paramétereinek és megjegyzéseinek nyelve angol, így a felhasználók köre nem korlátozódik Magyarországra. Jelenleg, mint kísérleti programot a felhasználóknak manuálisan kell elhelyezniük a QGIS plugins mappájába, viszont amint sikerül feltölteni a QGIS online tárhelyére (QGIS Plugins Repository, lásd: https://plugins.qgis.org/) úgy bármely QGIS felhasználó egyszerűen letöltheti a modulok menüpontban. Ezt a diplomamunka leadásáig nem volt lehetőségem megvalósítani.

A programot igyekeztem jól dokumentálni, a lényeges elemeket fel is tüntettem a dolgozatban, a teljes forráskód megtekinthető a mellékletben. Részletesen kifejtettem a program matematikai alapjait, jó útmutatóként szolgál a lineáris transzformáció használatához. Végül kipróbáltam a programot olyan térképeken, amelyek jelenleg legpontosabban ezzel a programmal georeferálhatók, ezt TIFF-formátumban szintén csatolom a mellékletben, a program által létrehozott GCP-fájllal együtt.

Köszönetnyilvánítás

Hatalmas köszönettek tartozom témavezetőmnek, Kerkovits Krisztiánnak, aki rengeteg időt fordított rám, előrelátó ütemtervet szabott ki, kiválóan ismertette és oldatta meg velem a feladat matematikai és informatikai problémáit. Hálásan köszönöm a munka szövegének átfogó ellenőrzését és segítségét annak logikai felépítésében is. Köszönöm a türelmét, segítőkészségét, kedvességét és közvetlenségét.

Köszönöm Gede Mátyásnak a diplomamunka témáját adó ötletet, a prototípus forráskódot, és a tesztelői javaslatokat.

Köszönöm Dusek Bencének, Nyulas Leventének, Németh Nórának, Sipos Kristófnak, Stork Mihálynak és Szabó Enikőnek a program tesztelését, és a javaslatokat.

Köszönöm Juhász Dorinának, valamint édesanyámnak, Szedlák Krisztinának és édesapámnak Zelenka Zoltánnak a diplomamunka tartalmának ellenőrzését és a formázási tanácsaikat.

Irodalomjegyzék

Burt, James E.; White, Jeremy; Allord, Gregory; Then, Kenneth M. & Zhu A-Xing (2020): *Automated and semi-automated map georeferencing*, Cartography and Geographic Information Science, 47:1, 46-66

Farkas Réka (2018): *A Görög–Kerekes atlasz vármegyetérképeinek adatbázisa és webes megjelenítése* (Diplomamunka), Budapest, ELTE Térképtudományi Tanszék

Garrard, Chris (2016): *Geoprocessing with Python*, Shelter Island (NY), Manning Publications Co.

Goyvaerts, Jan (2007): *Regular Expressions: The complete tutorial* (72. old.)

Huszánk Daniella (2019): *A Görög–Kerekes atlasz térképeinek vizsgálata vetületi szempontból* (Diplomamunka), Budapest, ELTE Térképtudományi Tanszék

Sárközy Ferenc (1987): *Geodézia* (old.: 99-100) Budapest, Tankönyvkiadó

Python dokumentáció (2020): *Python Documentation*, Python Sofware Foundation, url.: https://docs.python.org/3/

QGIS dokumentáció (2017): *QGIS User Guide, release 3.4*, QGIS Development Team, url.: https://docs.qgis.org/3.4/pdf/en/QGIS-3.4-UserGuide-en.pdf

QT dokumentáció (2020): Qt Company Ltd., url.: https://doc.qt.io/

Timár Gábor (2004): *GIS integration of the second military survey sections – a solution valid on the territory of Slovakia and Hungary*. Kartografické listy 12: 119-126. url.: http://sas2.elte.hu/tg/kl12_timar.htm

DIPLOMAMUNKA LEADÁSI
és
EREDETISÉG NYILATKOZAT

Alulírott ZELENKA BALÁZS Neptun-kód: BXFMAZ

az Eötvös Loránd Tudományegyetem Informatikai Karának, Térképtudományi és Geoinformatikai Tanszékén
TÉRKÉPEK GEOREFERÁLÁSA HIÁNYOS KOORDINÁTÁK SEGÍTSÉGÉVEL
című diplomamunkámat a mai napon leadtam.

Témavezetőm neve: DR. KERKOVITS KRISZTIÁN ANDRÁS

CD-t / DVD-t mellékelek *(aláhúzandó)*: igen nem

Büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom, hogy jelen szakdolgozatom/diplomamunkám saját, önálló szellemi termékem; az abban hivatkozott szakirodalom felhasználása a szerzői jogok általános szabályainak megfelelően történt.
Tudomásul veszem, hogy szakdolgozat/diplomamunka esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

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

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, ~~2018.~~ 2020. május 15.

Zelenka B.
hallgató aláírása