Eötvös Loránd Tudományegyetem
Informatikai Kar
Térképtudományi és Geoinformatikai Tanszék

Vetületek automatikus felismerése a fokhálózati vonalak képe alapján

DIPLOMAMUNKA

Készítette:
Barancsuk Ádám
térképész mesterszakos hallgató

Témavezető:
Gede Mátyás
adjunktus
ELTE Térképtudományi és Geoinformatikai Tanszék

Budapest, 2015

Tartalomjegyzék

Kivonat

A térképek vetületének ismerete alapvető fontosságú azok használatakor, különös tekintettel térképszerkesztési alapként és térinformatikai rendszerekben való felhasználásukkor. Ennek ellenére (különösen régebbi térképeknél) gyakori, hogy a használni kívánt térképre vonatkozó vetületi információ hiányos, vagy teljes egészében hiányzik.

Jelen diplomamunka célja egy olyan félautomatikus módszer kidolgozása, amely az Érdi-Krausz György által publikált vetületmeghatározási rendszerben (Érdi-Krausz, 1958) képes ismeretlen vetületű, kis méretarányú térképek vetületének és vetületi paramétereinek meghatározására a fokhálózati vonalak alakja és egyéb tulajdonságai alapján. E célra egy – laikusok által is használható – webes felületű alkalmazást fejlesztettünk. A felület rajzeszközök segítségével lehetőséget biztosít a fokhálózati vonalak képeinek kézi átrajzolására. Az átrajzolt vonalakra különböző algoritmusokat alkalmazva lehetővé válik azok görbetípusának és más jellemzőinek (egyenközűség, koncentricitás, metszési szögek stb.) meghatározása. Ezeket a tulajdonságokat ismerve a térkép vetülete elhelyezhető az Érdi-Krausz által javasolt hierarchiában.

Abstract

Knowing a map's projection is of essential importance, particularly when using them as a source for creating derivative works or dealing with them in a GIS environment. However (especially on older maps), projection information is often absent or partially present.

The objective of this thesis is to develop a semi-automated approach for determining the projection and projection parameters of a small-scale map, based on the shape and secondary properties of its graticule lines – as outlined in the hierarchy published by Érdi-Krausz, 1958. To this end, a web-based tool is created, explicitly designed to be usable by a non-professional audience. Drawing tools are provided for manually tracing graticule lines on pre-uploaded raster maps. Given the approximate traces, we employ a number of algorithms to determine the shape and secondary properties (e.g. equidistancy, concentricity, angles of intersection etc.) of graticule lines. Having computed these properties, one can fit the projection into Érdi-Krausz's system.

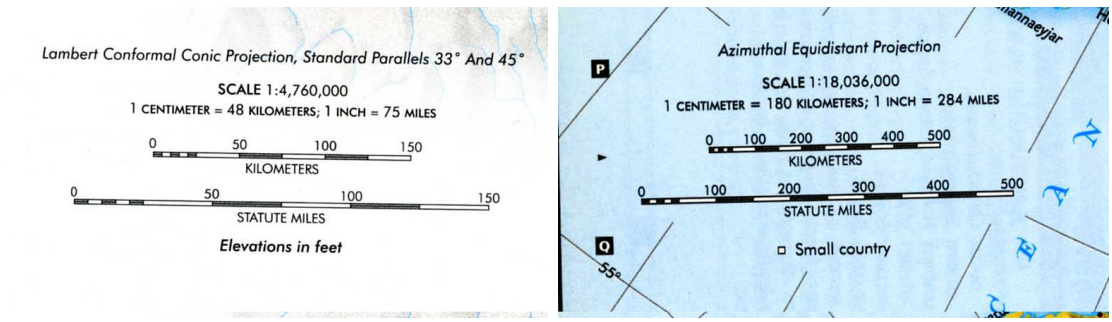
1. Bevezetés

Létező térképművek térképszerkesztési alapanyagként való felhasználásához szükség van azok vetületének ismeretére. Ez fokozottan érvényes a napjainkban egyre szélesebb teret nyerő térinformatikai háttérrel segített térképszerkesztés során: a különböző forrásokból származó térképi alapanyagok közös földrajzi referenciarendszerbe való szabatos beillesztésének alapvető feltétele a forrásmunkák vetületének és vetületi paramétereinek ismerete.

Egy térkép vetületének pontos ismeretéhez nemcsak a vetületi egyenleteknek, de a vetület egyéb paramétereinek (segédpólus helyzete, ferde helyzetben az elforgatás mértéke stb.) ismeretére is szükségünk van. (Györffy, 2012)

Sok esetben (főleg digitális térképi modellek esetén) ez az információ közvetlenül elérhető a térképhez csatolt leíró adatok (metaadatok) formájában. Más esetekben a vetületre vonatkozó információ csak közvetetten vagy részlegesen hozzáférhető (pl. atlaszoknál gyakran tapasztalható, hogy bár a használt vetület típusát feltüntetik, a pontos vetületi paraméterek közlésétől a szerkesztők eltekintenek). Emellett gyakori a térképre vonatkozó vetületi adatok teljes hiánya is.

(A részleges vagy hiányos vetületi adatok ugyanakkor nem mindig akadályozzák meg a térképi alapként való felhasználást: Nagy méretarányú, kis területet ábrázoló térképi kivágatoknál gyakran eltekinthetünk a vetületi különbségek problémájának megoldásától. A kartográfiai gyakorlatban emellett elterjedt az a módszer is, amely úgy használ fel egyidejűleg két különböző vetületű térképet forrásként, hogy azokat egy közös pontban egymáshoz illeszti, majd a pont valamilyen környezetében elhanyagolhatónak tekinti a vetületi különbségeket. Nagyobb terület esetén több ilyen lokálisan optimális illeszkedés használatára van szükség.)



1.1. ábra. *Teljes vetületi definíciók a* National Geographic Family Reference Atlas of The World *kiadványban (*National Geographic Society*, 2002).*

A vetületmeghatározás problémája azonban nemcsak a térképszerkesztésben jelentkezik: Világszerte folyamatosan zajlik a papíralapú könyvtári állományok digitalizálása, illetve a már digitális formában hozzáférhető dokumentumok beillesztése a könyvtári katalógusokba. A folyamat során magától értetődő feladat az állomány elemeinek leíró adatokkal való ellátása. A metaadatok tárolásának legelterjedtebb szabványa az 1960-as években, az Egyesült Államok Kongresszusi Könyvtárában kidolgozott MARC *(machine readable catalog)* formátum. Hozzájárult a MARC széleskörű elterjedéséhez az is, hogy ez a formátum szolgált a Nemzetközi Szabványügyi Szervezet által 1981-ben elfogadott, a bibliográfiai adatok tárolását egységesítő ISO 2709[1] szabvány alapjául.

A MARC-ban több mező is szolgál a térképművek vetületének leírására. Míg a `008`-as kódú mezőben térképművek esetén egy kétkarakteres azonosító van fenntartva a térképvetület meghatározására, addig az elem fizikai leírását *(`3XX` – Physical Description, Etc. Fields)* megadó mezők között szerepel egy részletesebb, a földrajzi referenciára *(`342` – Geospatial Reference Data)* vonatkozó mező is. Ezek segítségével részletekbe menően megadhatóak egy térképmű geodéziai alapjára és vetületére vonatkozó információk.

Kód	**Vetület**	**Kód**	**Vetület**
ac	Lambert-féle területtartó síkvetület	**bd**	Mercator-vetület
bi	Gauss–Krüger vetületi rendszer	**ca**	Albers-féle területtartó kúpvetület
cb	Bonne-vetület	**da**	Raisz Erwin-féle Armadillo-vetület
zz	Egyéb	##	*(nincs meghatározva)*

1.1. táblázat. *Részlet a MARC által definiált térképvetületekből. A teljes lista közel ötven elemet tartalmaz.*

A MARC rekordjaihoz hasonlóan az INSPIRE (Infrastructure for Spatial Information in the European Community) szabvány is lehetővé teszi részletes vetületi információk hozzárendelését téradatokhoz.

A gyakorlatban ezek a mezők azonban – a kitöltő vetülettani ismereteinek hiányában – gyakran vagy üresen maradnak, vagy részleges, a vetület pontos megállapítására alkalmatlan adatokat tartalmaznak.

A vázolt két probléma (szabatos georeferencia a GIS-rendszerekben való felhasználáshoz és könyvtári dokumentumok metaadatainak meghatározása) szolgált motivációul a dolgozat témájának megválasztásakor: lehetséges-e a fenti problémák számítógép segítségével automatizált megoldása?

Az alábbiakban bemutatjuk az Érdi-Krausz György által megalkotott vetületazonosítási rendszert, amely a probléma jelen dolgozatban javasolt megoldásának alapjaként szolgál. Ezek után áttekintjük a problémakör megoldására más szerzők által javasolt félautomatikus módszereket, majd összehasonlítjuk ezeket a dolgozatban bemutatott módszerrel, annak előnyeit és hátrányait előtérbe helyezve.

[1] `http://www.iso.org/iso/catalogue_detail.htm?csnumber=41319`

A dolgozat második felében bemutatjuk az általunk javasolt módszer alkalmazott matematikai alapjait és kísérletet teszünk az algoritmusok konkrét implementációjára is. Az így elkészült szoftvert szintetikus környezetben teszteljük, elemezve az implementáció és a tesztelés során felismert problémákat és azok lehetséges megoldásait.

1.1. Vetületanalízis

A magyar kartográfiai szakirodalomban elsőként Érdi-Krausz György foglalkozott ismeretlen térképi vetületek meghatározásával. A *Studia Cartologicában* megjelent tanulmányát elsősorban az motiválta, hogy a pontos vetület ismeretében kiszámíthatóak a vetületi torzulások, így lehetővé válik pontos mérések elvégzése a térképen (Érdi-Krausz, 1958). A tanulmány túlmutat a vetületmeghatározás jelen dolgozatban tárgyalt, szűkebb értelemben vett témakörén: Érdi-Krausz nemcsak erre a kérdésre ad választ, de részletesen tárgyalja a pontos vetületi paraméterek meghatározásához szükséges számítási módszereket, és útmutatást ad arra az esetre is, amikor csak az adott vetület torzulási tulajdonságait szeretnénk meghatározni a vetület pontos ismerete nélkül. Ezen kívül módszereket közöl a térképeken kézzel, vagy egyszerű és bonyolultabb műszerekkel (pl. tükrös vonalzó, prizmás derivátor) elvégezhető mérések pontos menetére is.

1.2. A vetületanalízis főcsoportjai a fokhálózat alakja alapján

Érdi-Krausz tanulmányában közöl egy vetületek rendszerezésére alkalmas csoportrendszert, melynek elsődleges rendezési elve a térképi fokhálózat vonalainak jellege, görbéinek típusa. A rendszer lehetővé teszi, hogy a vizsgált vetületet először a fokhálózati kép alapján valamely főcsoportba soroljuk, majd további mérések segítségével a meghatározzuk annak pontos típusát a csoporton belül. Az Érdi-Krausz által javasolt főcsoportok az alábbiak:

Főcsop.	**Meridiánok**	**Szélességi körök**
1.	Összetartó egyenesek	Koncentrikus zárt körívek
2.	Összetartó egyenesek	Koncentrikus nyílt körívek
3.	Összetartó egyenesek	Kúpszeletek
4.	Párhuzamos egyenesek	Párhuzamos egyenesek
5.	Párhuzamos egyenesek	Kúpszeletek
6.	Körívek	Körívek
7.	Görbe vonalak	Körívek
8.	Ellipszisívek	Párhuzamos egyenesek
9.	Ellipszisívek	Ellipszisívek
10.	Ellipszisívek	Görbe vonalak
11.	Szinuszgörbék	Párhuzamos egyenesek
12.	Görbe vonalak	Görbe vonalak

1.2. táblázat. *Az Érdi-Krausz által javasolt vetületcsoportosítási rendszer.*

Az Érdi-Krausz által javasolt vetületmeghatározási rendszert fejlesztette tovább Györffy János (Györffy, 2012). Ez a változat nagyobb részben megtartja a főcsoportok kijelöléséhez használt szempontokat, azonban azokat kismértékben módosítja, átalakítja a csoportok sorrendjét, többet pedig összevon:

Főcsop. (Györffy)	**Főcsop. (É.-K.)**	**Meridiánok**	**Szélességi körök**
1.	**4.**	Párhuzamos egyenesek	Párhuzamos egyenesek
2.	**8., 11.**	Egyéb vonalak	Párhuzamos egyenesek
3.	**1.**	Összetartó egyenesek	Koncentrikus zárt körívek
4.	–	Egy pontba összefutó görbék	Koncentrikus zárt körívek
5.	**2.**	Összetartó egyenesek	Koncentrikus nyílt körívek
6.	**6., 7.**	Egyéb vonalak	Körívek
7.	**5.**	Párhuzamos egyenesek	Hiperbolák
8.	**3.**	Összetartó egyenesek	Kúpszeletek
9.	**9.**	Ellipszisívek	Ellipszisívek
10.	–	Hiperbolák	Ellipszisívek
11.	**10., 12.**	Görbe vonalak	Görbe vonalak

1.3. táblázat. *A Györffy által javasolt vetületcsoportosítási rendszer.*

A rendszer egy további módosítását adja Gede és Barancsuk, 2015:

Főcsop. (Gede)	**Főcsop. (Györffy)**	**Főcsop. (É.-K.)**	**Meridiánok**	**Szélességi körök**
1.	**3.**	**1.**	Összetartó egyenesek	Koncentrikus zárt körívek
2.	**5.**	**2.**	Összetartó egyenesek	Koncentrikus nyílt körívek
3.	**1.**	**4.**	Párhuzamos egyenesek	Párhuzamos egyenesek
4.	**2.**	**8., 11.**	Egyéb vonalak	Párhuzamos egyenesek
5.	**6.**	**6., 7.**	Egyéb vonalak	Körívek
6.	**4.**	–	Egy pontba összefutó görbék	Koncentikus zárt körívek
7.	**7.**	**5.**	Párhuzamos egyenesek	Hiperbolák
8.	**8.**	**3.**	Összetartó egyenesek	Kúpszeletek
9.	**9.**	**9.**	Ellipszisívek	Ellipszisívek
10.	**10.**	–	Hiperbolák	Ellipszisívek
11.	–	–	*Két szimmetriatengellyel rendelkező vetületek*	
12.	**11.**	**10., 12.**	Görbe vonalak	Görbe vonalak

1.4. táblázat. *A Gede által javasolt vetületcsoportosítási rendszer.*

2. Numerikus módszerek a vetületanalízisben

2.1. Korábbi munkák

2.1.1. Régi térképek online georeferálása

Kapcsolódó kulcsszavakra keresve[1] az interneten látható, hogy az ismeretlen vetületű térképek georeferálása gyakori probléma, ennek ellenére az ajánlott megoldások legtöbbször nem általános célúak, csak az adott problémára vonatkoznak.

Több olyan webszolgáltatás vagy webes felületű, nyílt forrású szoftver ismert, amely lehetőséget biztosít régi, vetületi információ nélküli térképek georeferálására. Ilyen a MapWarper[2] és a rá épülő WorldMap WARP[3], a MetaCarta Labs Rectifier[4] nevű szoftvere és a Georeferencer[5] (2.1 ábra). Ezek közös tulajdonsága, hogy a georeferálás során nem törekednek a térkép eredeti vetületének felismerésére, hanem a felhasználó által megadott illesztőpontpárok földrajzi és térképi koordinátapárjainak ismeretében hajtanak végre valamilyen (általában hasonlósági, affin vagy splinealapú) transzformációt a raszteres képen. Ezek a szoftverek legtöbb esetben a nyílt forrású GDAL és OGR függvénykönyvtárakra támaszkodnak, de az eredeti vetület ismerete nélkül csak korlátozott pontosságú eredményt nyújtanak.

Az internetes változatokhoz hasonló eszközök hozzáférhetőek minden közismert asztali GIS szoftverben is, így a QGIS az ArcGIS és a Global Mapper különböző verzióiban. ArcGIS környezetben továbbá hozzáférhető olyan dokumentáció[6] is, amely segít a georeferálandó térkép vetületének meghatározásában, valamint megjelent egy, a témával foglalkozó könyv is (Kessler, 2006).

[1] http://stackoverflow.com/questions/1816996/determining-the-projection-in-a-given-map
http://gis.stackexchange.com/questions/7839/identifying-coordinate-system-of-shapefile-when-unknown
http://gis.stackexchange.com/questions/22540/how-do-i-determine-the-projection-of-a-paper-map

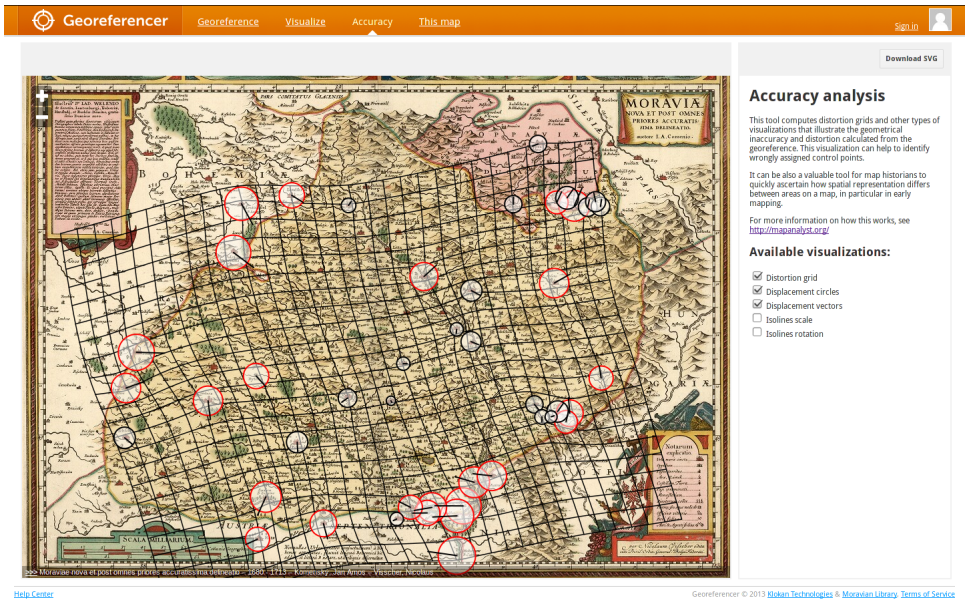
[2] https://github.com/timwaters/mapwarper

[3] http://warp.worldmap.harvard.edu

[4] https://github.com/crschmidt/labs-rectifier

[5] http://www.georeferencer.com

[6] http://support.esri.com/es/knowledgebase/techarticles/detail/24893



2.1. ábra. *Lokális torzulási viszonyok a Georeferencer webalkalmazásban.*

2.1.2. MapAnalyst

Az Oregon State University-n fejlesztett MapAnalyst[7] szoftver elsődleges célja régi, papír alapú térképek lokális és globális torzulási jellemzőinek kiszámítása, valamint ezek vizualizációja. Ahogy az előző szakaszban tárgyalt szoftverek, úgy a MapAnalyst is az eredeti térképen és egy referenciatérképen egyaránt elhelyezett illesztőpontok segítségével számítja ki a torzulási jellemzőket. Kimenetként azonban nem csak egy georeferált rasztert készít el, hanem kiszámolja a szóban forgó térkép méretarányát, elforgatási szögét is. Megjeleníthetőek vele ezen kívül a térképi pontokat azok valós helyével összekötő eltolásvektorok és a térképre jellemző torzulási háló is.

A MapAnalyst által használt technológiát Jenny és Hurni, 2011 mutatja be. Amennyiben az eredeti térkép vetülete ismeretlen, úgy a szoftver megpróbálja azt naiv próbálgatással helyreállítani. Ehhez lehetséges vetületek egy beépített listájára végrehajtja az alábbi lépéseket:

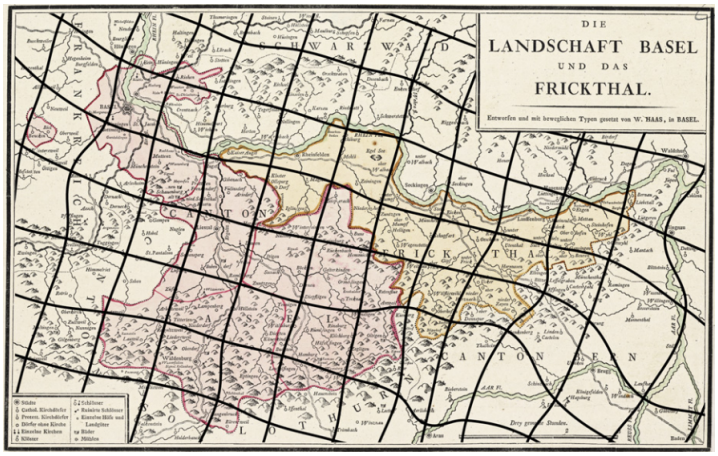
1. A referenciatérkép illesztőpontjainak földrajzi koordinátákká alakítása, majd a választott, lehetséges vetületbe transzformálása;

2. Affin transzformáció végrehajtása az illesztőpontok képein úgy, hogy azok a lehető legjobban illeszkedjenek az eredeti, georeferálandó képen található illesztőpontokhoz;

3. Az eredeti térkép illesztőpontjai és a referenciatérkép transzformált illesztőpontjai közti távolsághiba szórásának kiszámítása.

A MapAnalyst a fenti eljárást minden lehetséges vetületre többször elvégzi, miközben az adott vetülethez tartozó paramétereket a megadott intervallumokon belül változtatja. Végül feltételezi, hogy az eredeti térkép vetülete az, amelyre a fenti algoritmus által kiszámított szórás a legkisebb.

[7] `http://www.mapanalyst.org`



2.2. ábra. *Torzulási háló a MapAnalyst szoftverben*

2.1.3. Blue Marble Projection Recovery

A Global Mapper fejlesztője, a Blue Marble 2009-ben jelentette be[8] Projection Recovery nevű módszerét, amely Geographic Calculator nevű szoftverében publikált. Az algoritmus szintén illesztőpontokkal dolgozik, de a georeferálást nem direkt interpoláció segítségével végzi, hanem – egy ismeretlen, a Blue Marble által „fordított interpolációként" említett módszerrel – kísérletet tesz a vetület megállapítására is.

A vizsgált módszerek közül ez az egyetlen, amely nem nyílt forrású, így csak az azt megvásárlók számára hozzáférhető. A Blue Marble azonban a saját termékeiben való felhasználáson kívül más cégek számára is licenceli a Geographic Calculator technológiáit GeoCalc SDK néven. A GeoCalc SDK-t használja többek között az Avenza cég MAPublisher nevű, Adobe Illustratorhoz készült térinformatikai–kartográfiai kiegészítője is, amely minden jel szerint ezt a technológiát alkalmazza ismeretlen vetületű vektoros adatok vetületének közelítő megállapítására.

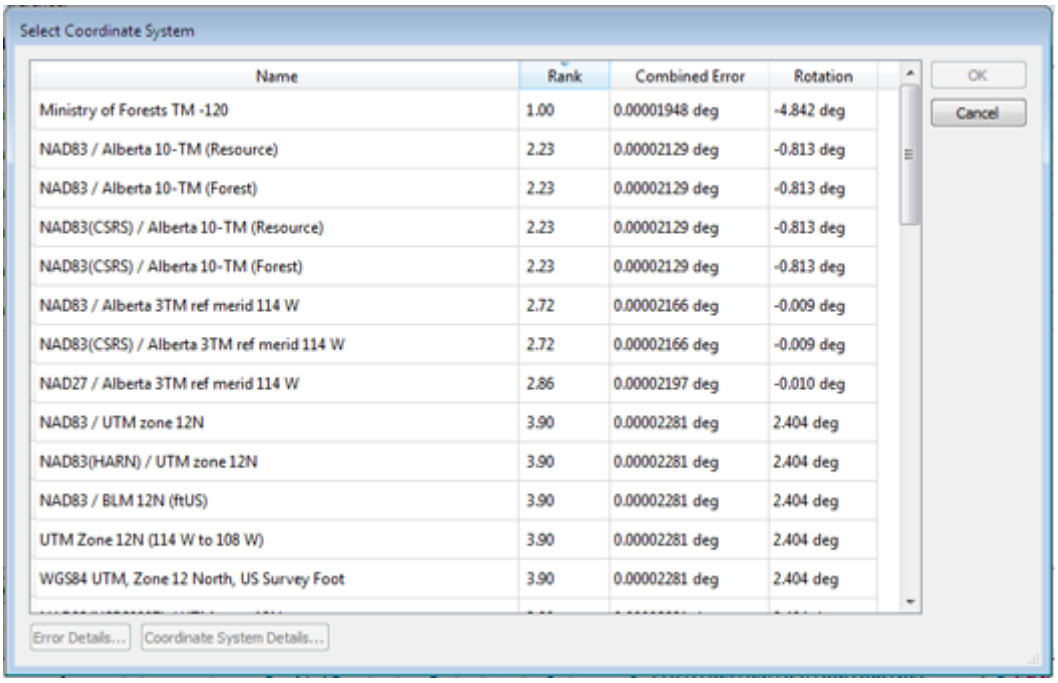
A MAPublisher vonatkozó funkciójának felhasználói felülete (2.3 ábra) alapján sejthető, hogy a Projection Recovery a következő szakaszban tárgyalt `detectproj`-hoz hasonlóan valamilyen hibamérték szerint csökkenő sorrendben rangsorolja a vetületeket.

2.1.4. `detectproj`

Tomáš Bayer tanulmányaiban (Bayer et al., 2010, Bayer, 2014) több, régi térképek félautomatikus georeferálására alkalmas módszert bemutat. A módszereket a szabad szoftverként közzétett `detectproj`[9] nevű alkalmazásban implementálja.

A Bayer által javasolt módszer több szempontot vesz figyelembe, melyeket egy optimalizációs probléma közös célfüggvényében egyesít. A célfüggvény paramétervektora a vizsgált megoldástér egy pontja, vagyis egy lehetséges vetület $\mathbf{X} = (\varphi_k, \lambda_k, \varphi_0, \lambda_0)$

[8] `http://www.bluemarblegeo.com/about-us/press.php?id=179`
[9] `http//github.com/bayertom/detectproj`



2.3. ábra. *Ismeretlen vetületű vektoros adatok vetületének meghatározása illesztőpontok alapján a MAPublisher szoftverben*

vetületi paraméterei, ahol φ_k és λ_k a segédpólus földrajzi koordinátái, φ_0 a hossztartó parallelkör szélessége és λ_k a középmeridián hosszúsága. A célfüggvény minden vizsgált vetülethez egy skaláris értéket rendel, amely jellemzi a vetület jóságát.

A célfüggvény komplexitása nem teszi lehetővé olyan optimalizációs módszer használatát, amely feltételezi a függvény gradiensének ismeretét, így a szerző a tanulmányban a függvény minimalizálására a Nelder–Mead algoritmusra épülő *downhill simplex* módszert javasolja.

Az először 1965-ben publikált Nelder–Mead módszer (Nelder és Mead, 1965) az egyik leggyakrabban használt direkt optimalizációs algoritmus többdimenziós, nemlineáris problémák megoldására. Az évek során sok variációja látott napvilágot, amelyek létrejöttét általában az eredeti algoritmus hibái motiválták: a Nelder–Mead heurisztikus volta nem garantálja a globális optimum megtalálását: eredményessége nagyban függ a paramétervektor kezdeti értékétől, és bizonyos problémák esetén hajlamos lokális minimumokhoz konvergálni, ahonnan aztán képtelen a további iterációk során elmozdulni (Lagarias et al., 1998).

A MapAnalysthoz és a Blue Marble Projection Recoveryhez hasonló módon a `detectproj` is kontrollpontok vizsgált és referenciatérképi, egymásnak megfeleltethető leképezésére alapoz. A referenciatérképi pontokat az aktuálisan vizsgált vetületbe transzformálja, majd egy hasonlósági transzformáció paramétereinek megbecslésével megállapítja azok hasonlóságát a vizsgált térkép pontjaihoz.

A `detectproj` a megbízhatóság növelésére az illesztőpontokon kívül képes magasabb dimenziószámú vektoros térképi elemek (töröttvonalak és poligonok) hasonlóságának elemzésére is. (Mivel ezek a célfüggvény bonyolultságát tovább növelik, a program képes egy egyszerűsített elemzés elvégzésére is, ami ezeket nem veszi figyelembe.)

Amennyiben nem állnak rendelkezésre töröttvonalak, a detectproj ezeket a kontrollpontok alapján automatikusan megtalált fokhálózati görbékkel, a poligonokat pedig a pontszerű elemekből képzett Voronoi-cellákkal helyettesíti. A javasolt módszer a magasabb dimenziószámú elemek hasonlóságának megállapítására több módszert is ajánl.

A `detectproj` ezen kívül több olyan heurisztikát alkalmaz a célfüggvény kiszámítása előtt, amelyek képesek a probléma számításelméleti bonyolultságának és így a futási időnek a csökkentésére. Azonban a jelenleg használt, Nelder–Mead alapú optimalizáció számításigényessége nem teszi lehetővé a módszer valós- vagy közel valósidejű felhasználását. Ennek ellenére tudomásunk szerint az ismeretlen vetületek meghatározásának problémáját a legátfogóbb módon a `detectproj` oldja meg: képes nemcsak a vetületi típus, de a konkrét vetületi paraméterek robosztus kiszámítására is.

Az alkalmazás parancssoros interfésze a `Proj.4` vetületi függvénykönyvtáréval analóg módon került megalkotásra és az optimalizáció eredményeképp előálló vetületi paramétereket is ebben a formátumban bocsátja a fehasználó rendelkezésére.

2.2. Az általunk javasolt algoritmus

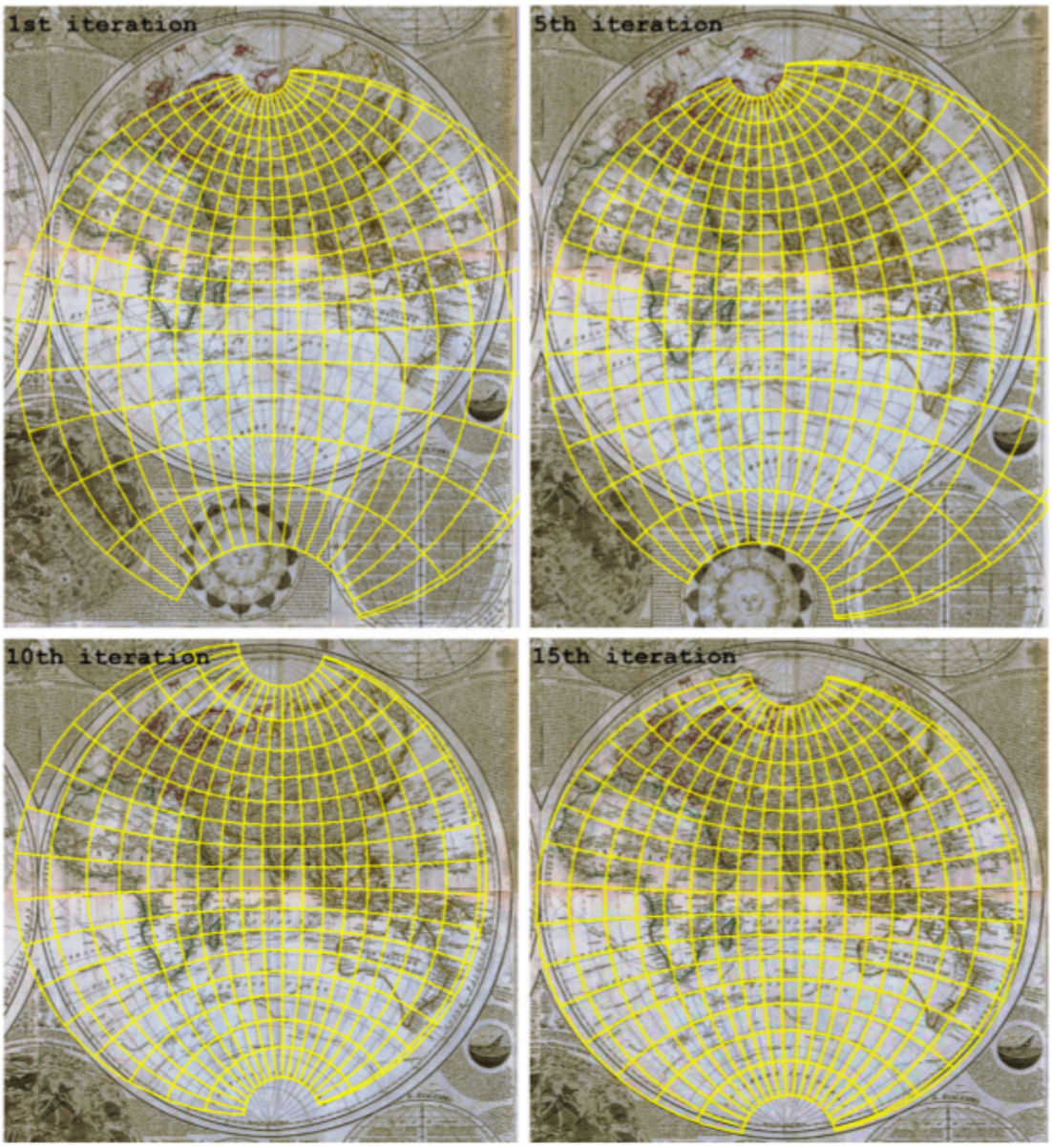
Az előző fejezetben láthattuk, hogy a vizsgált, ismeretlen térképvetületek meghatározását végző módszerek (MapAnalyst, Blue Marble Projection Recovery, `detectproj`) bár implementációs részleteikben, használt algoritmusaikban, a figyelembe vett térképi elemek dimenziószámában eltérnek ugyan, mégis egymással analóg módon tekintenek a problémára:

1. Tekintsük a lehetséges vetületek egy listáját, valamint a vizsgált és egy referenciatérképen elhelyezkedő, egymásnak megfeleltetett kontrollpontpárokat.

2. Haladjunk végig a lehetséges vetületeken, transzformáljuk a referenciatérképi kontrollpontokat az aktuálisan vizsgált vetületbe, majd valamilyen módszerrel állapítsuk meg a vizsgált térkép kontrollpontjai és a transzformált kontrollpontok közti különbséget.

3. Az így kiszámított különbségekből képezzünk egy, a vetület jóságát jellemző skaláris mennyiséget. A vetületeket eszerint sorbarendezve tekintsük a legkisebb hibájú, tehát legnagyobb jóságú vetületet a legjobb megoldásnak.

A dolgozatban leírt módszer nem követi a fenti gondolatmenetet. Az ismert vetületek listáján való végighaladás helyett az általunk javasolt algoritmus a fokhálózati képnek az Érdi-Krausz György által leírt vetületmeghatározási rendszer csoportjaiba való elhelyezésére, mint szervezési elvre épül. Mivel a csoportosítás a fokhálózati vonalak képeinek típusát tekinti elsődleges szempontnak, mi is csak a szélességi és hosszúsági körök képeit vizsgáljuk. Az alkalmazás jelen állapotában a fokhálózati vonalak átrajzolását a felhasználóra bízzuk, és az így kapott nyomvonal pontjaival, mint illesztőpontokkal dolgozunk tovább. Az illesztőpontok halmazából kiindulva



2.4. ábra. *Transzverzális helyzetű sztereografikus vetület iteratív illesztése a* `detectproj` *szoftverben (*Bayer*, 2014).*

minden átrajzolt nyomvonalról megállapítjuk annak görbetípusát (egyenes illetve kúpszeletek különböző típusai).

Ezt követően a csoportosítás alapjául szolgáló további ismérvekből (görbék metszéspontjainak egyenközűsége, metszési szögek, póluspontosság/-vonalasság stb.) egy *döntési fát* képezünk. Kiszámolva a vizsgált fokhálózati vonalakra vonatkozó fenti tulajdonságokat, a gyökérből kiindulva végighaladhatunk a döntési fán és besorolhatjuk a vetületi képet valamely konkrét csoportba. Bár a dolgozatban ezt nem tárgyaljuk, ezt követően további vizsgálatokkal lehetőségünk van a konkrét vetületi paraméterek megállapítására is.

A javasolt módszer előnyei:

- A fokhálózati kép vizsgálata gyorsabb mint a lehetséges vetületek listáján való végighaladás és az illeszkedés vetülettípusonkénti vizsgálata.

- A javasolt módszer ugyan használ iteratív optimalizációt, de a választott módszer (Levenberg–Marquardt) épít a célfüggvény Jacobi-mátrixára, így a konvergenciája jobb és futása gyorsabb mint a Nelder–Mead algoritmusé.
- A javasolt algoritmusok nem heurisztikusak.
- Az Érdi-Krausz féle vetületmeghatározási rendszer ugyan nem veszi figyelembe minden vetület ferdetengelyű és transzverzális helyzetét, azonban a benne szereplő vetületek lefedik a geokartográfiában leggyakrabban használt vetületek körét. Csak ezekre koncentrálva nem kell olyan vetületek, vagy vetületváltozatok illesztésével foglalkoznunk, amelyek csekély valós térképészeti jelentőséggel bírnak.

Ugyanakkor az előző fejezetben tárgyalt módszerekkel összehasonlítva a javasolt módszernek több hátránya is van:

- Csak olyan vetületeket tudunk felismerni, amelyek szerepelnek az Érdi-Krausz féle hierarchiában. Ennek kiterjesztése bár lehetséges, de korántsem triviális. A fenti módszerek közös tulajdonsága, hogy új vetületek figyelembe vétele csak a vetületi egyenletek programba való beépítését igényli, mivel az illesztéshez használt módszerek vetületagnosztikusak. Az általunk javasolt módszerre ez nem igaz.
- Nem tudunk egy vetületfüggetlen értéket rendelni az illesztett vetületek jóságához.
- Néhány esettől eltekintve nem foglalkozunk a vetületek transzverzális és ferdetengelyű változataival.
- Módszerünk jelen dolgozat keretei között tárgyalt változata csak világtérképek vetületeinek felismerésére alkalmas és továbbfejlesztésével is csak kisméretarányú térképek vetületének felismerése válik lehetségessé.

2.3. Görbék illesztése fokhálózati vonalakra

Az Érdi-Krausz által felállított vetületmeghatározási rendszer elsődleges szempontként a fokhálózati vonalak jellegét vizsgálja, főcsoportjait a görbék típusai szerint különíti el. Ehhez mérten az általunk javasolt módszer is elsősorban a felhasználó által megrajzolt fokhálózati vonalak görbéinek felismerésére koncentrál. Az Érdi-Krausz-féle rendszer az alábbi alapvető görbetípusokat különíti el:

1. Egyenesek
2. Körívek
3. Ellipszisívek
4. Parabolaívek
5. Hiperbolaívek
6. Szinuszívek

A dolgozatban vázolt módszer ezek közül az első öt típus felismerését teszi lehetővé (lásd még: 4.3.1 szakasz). Figyelembe véve, hogy a 2-5. számú görbék mind kúpszeletek alesetei, egy olyan algoritmuspárt javaslunk, amely a görbefelismerést két lépésben végzi: először egy egyenest próbál meg a nyomvonalra illeszteni, majd, ha az illesztés sikertelen, egy kúpszeletillesztéssel próbálkozik.

2.3.1. Legkisebb négyzetek módszere

Ahhoz, hogy megragadhassuk, mit értünk a görbeillesztő algoritmusaink „sikeressége" alatt, fontos megértenünk, hogy a bemeneti adatként használt, a felhasználó által megrajzolt nyomvonalak – legyenek bármilyen pontosak –, jellegükből adódóan matematikai-statisztikai értelemben hibákkal terheltek. Ez azt jelenti, hogy a felvett pontokra a legtöbb esetben sem egyenes, sem kúpszelet nem illeszthető pontosan. Az ilyen esetekben a feladat egy optimalizációs problémává alakul át.

Az általunk vizsgált görbeillesztési problémák esetén a cél egy olyan paraméteres görbeegyenlet megkeresése, amelyre a felhasználó által megrajzolt nyomvonal pontjainak mindegyike a lehető legjobban illeszkedik. A probléma egy N (az illeszteni kívánt pontok számával megegyező) elemű egyenletrendszer megoldására vezethető vissza, ahol az egyenletek $f(x_i, \mathbf{p}) = 0$ alakúak, $i = 1..N$ a mérési pontok koordinátái és $\mathbf{p} \in \mathbf{R}^M$ a görbe ismeretlen paramétervektora.

A mérési pontok azonban egyrészt a fent tárgyalt hibával terheltek, másrészt esetünkben a probléma jellemzően *túlhatározott*, vagyis a felhasználó által megrajzolt nyomvonal több pontból áll, mint ahány pont a keresett görbét pontosan meghatározza.[10] Így $N > M$, a megoldandó probléma tehát több egyenletből áll, mint ahány ismeretlen van, vagyis az egyenletrendszernek a legtöbb esetben nincs egyértelmű megoldása. Olyan módszert kell keresnünk, amely az egyenletrendszer egy közelítő megoldását adja, úgy, hogy annak hibája valamilyen szempontból minimális legyen.

A hiba egy lehetséges definíciója egy olyan $\mathbf{r} \in \mathbb{R}^N$ eltérésvektor, amellyel a nyomvonal pontjainak $\mathbf{x}$ pozícióit javítva az $f(\mathbf{x} + \mathbf{r}, \mathbf{p}) = 0$ egyenletrendszer egyértelműen megoldhatóvá válik. A cél tehát annak a $\mathbf{p}$ paramétervektornak a megkeresése, amire az $\mathbf{r}$ hibavektor nagyságai minimálisak. A statisztikában megszokott módon a hiba helyett a hibanégyzetek minimalizálására törekedve az alábbi optimalizációs problémához jutunk:

$$min \sum_{i=1}^{N} \|r_i\|^2 \tag{2.1}$$

Ez az ún. *legkisebb négyzetek módszere*. Az alábbiakban bemutatjuk, hogyan oldhatóak meg ezzel a módszerrel a konkrét görbeillesztési feladatok.

[10] Az euklideszi síkon egy egyenest két, egy kúpszeletet öt pont határoz meg. Utóbbi esetben – a kúpszelet elfajultságát megakadályozandó – kikötjük továbbá, hogy semelyik három pont nem eshet egy egyenesbe.

2.3.2. Koordinátarendszer-választás

A felhasználó által megrajzolt nyomvonal pontjai egy földrajzi és térképi koordinátarendszerektől független euklideszi síkkoordinátarendszerben értelmezettek. A további feldolgozáshoz ezeket a koordinátapárokat homogén koordinátákká alakítjuk.

A homogén (más néven projektív) koordináták egy n dimenziós tér pontjainak helyzetét $n+1$ koordináta segítségével írják le. Ehhez az eredeti koordinátavektor $(x_1, x_2, \dots x_n)$ koordinátáit egy c skalárral szorozzuk, és ezt tekintjük az $n+1$-edik koordinátának $((cx_1, cx_2, \dots cx_n, c))$.

A homogén koordinátákat széles körben alkalmazzák a számítógépes képfeldolgozásban és grafikában, mert lehetővé teszik affin és projektív transzformációk, valamint a tér elemei (pl. pontok, egyenesek és kúpszeletek) közti műveletek mátrixműveletekkel való leírását.

2.3.3. Egyenesillesztés

A kétdimenziós térben egy egyenes leírható a meredekségét és az egyik tengelyen való metszéspontját tartalmazó $y = ax+b$ egyenlettel. Ez az alak azonban nem teszi lehetővé függőleges egyenesek leírását, amelyek a vizsgált problémakörben viszont rendszeresen előfordulnak. Emiatt az egyeneseinket homogén koordinátahármasokkal határozzuk meg. Egy általános alakban adott $ax+by+c=0$ egyenest homogén koordinátákkal az $\mathbf{l} = (a, b, c)$ vektor, míg a $P(x, y)$ pontot (például) az $\mathbf{p} = (x, y, 1)$ vektor írja le. Egy pont akkor esik egy egyenesre, ha koordinátái kielégítik az egyenes egyenletét. Homogén koordinátarendszerben ennek az összefüggésnek a bal oldala az egyenes és a pont vektorainak skaláris szorzatává alakul át: $\mathbf{p} \cdot \mathbf{l} = 0$.

Célunk egy olyan egyenes megkeresése, amely a megrajzolt nyomvonal pontjainak mindegyikére (a lehető legjobban) illeszkedik. N pont esetén ez egy N elemű lineáris egyenletrendszer (közelítő) megoldásaként adódik. Tekintsük ehhez a nyomvonal pontjainak homogén kooridnátáiból képzett

$$\mathbf{A} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & 1 \end{pmatrix} \tag{2.2}$$

mátrixot. Ekkor a probléma leírható az $\mathbf{Ax} = 0$ mátrixegyenletként, ahol $\mathbf{x} \in \mathbb{R}^3$ a keresett egyenes paramétervektora. (Minden ilyen ún. *homogén lineáris egyenletrendszernek* triviális megoldása az $\mathbf{x} = 0$ vektor. Nemtriviális megoldásuk akkor van, ha $rank(A) < N$.) A megoldáshoz keressük az $\mathbf{A}$ mátrix szinguláris felbontását *(singular value decomposition, SVD)*. Egy $A \in \mathbb{R}^{n \times m}$ mátrix szinguláris felbontása alatt azt a bizonyíthatóan létező

$$A = U\Sigma V^T \tag{2.3}$$

mátrixszorzatot értjük, ahol $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{m\times m}$ szinguláris mátrixok, $\Sigma \in \mathbb{R}^{n\times m}$ elemei pedig a főátlóban elhelyezkedő $\sigma_1 > \sigma_2 > \ldots > \sigma_N > 0$ értékektől eltekintve nullával egyenlőek. Ekkor U-t *A bal oldali*, V-t *A jobb oldali szinguláris vektorainak*, $\sigma_1, \sigma_2, \ldots \sigma_N$-t pedig *A szinguláris értékeinek* hívjuk. Bizonyítható, hogy

$$\mathbf{x} = \{V^T_{m,1}, V^T_{m,2}, \ldots V^T_{m,N}\} \tag{2.4}$$

vagyis a homogén lineáris egyenletrendszer legkisebb négyzetek módszerével kapott közelítő megoldása (ami esetünkben a keresett egyenes homogén paramétervektora) megegyezik a legnagyobb szinguláris értékhez tartozó jobb oldali szinguláris vektorral (Fogaras, 2002).

2.3.4. Kúpszeletillesztés

A kúpszeletillesztés problémája leírható a következő módon. Adott:

- Egy kétdimenziós térben értelmezett, homogén koordinátákkal adott pontok egy $P = \{\mathbf{x}_i\}(\mathbf{x}_i = (x_i, y_i, 1))$ halmaza,
- kúpszeletek egy $C(\mathbf{a})$ csoportja, ahol $\mathbf{a}$ a kúpszelet paramétervektora,
- és egy $\delta(C(\mathbf{a}), \mathbf{x})$ távolságfüggvény, amely jellemzi $\mathbf{x}$ pont távolságát $C(\mathbf{a})$-tól.

Keressük azt az $\mathbf{a}_{min}$ paramétervektort, amire a pont-kúpszelet távolságokból képzett

$$\epsilon^2(\mathbf{a}) = \sum_{i=1}^{n} \delta(C(\mathbf{a}), \mathbf{x}_i) \tag{2.5}$$

hibafüggvény értéke minimális.

A fenti a probléma nagyon gyakran felmerül a számítógépes látás és grafika szakterületén belül, így jól kutatott. Megoldására általában a legkisebb négyzetek módszerének valamilyen variánsát használják, de más módszerek is ismertek (Zhang, 1997). A téma szakirodalma széles: különböző algoritmusok ismertek kúpszeletek egy konkrét variánsának (általában köröknek vagy ellipsziseknek) az illesztésére (Fitzgibbon, Pilu, et al., 1999) és az általános esetre is (Bookstein, 1979, Kanatani, 1994, Sturm és Gargallo, 2007). Ezekről jó áttekintést és összehasonlítást ad Fitzgibbon és Fisher, 1995.

Az algoritmusok közti legfontosabb különbség a minimalizálandó távolságfüggvény jellegéből adódik. Ez alapján algoritmusok két fő csoportját különböztethetjük meg:

Algebrai távolság minimalizációja

A kúpszeletek általánosan a

$$C(\mathbf{a}) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 = 0 \tag{2.6}$$

másodfokú paraméteres alakban adhatóak meg. Ekkor a kúpszeletre illeszkedő pontok halmaza:

$$\delta(C(\mathbf{a}), \mathbf{x}) = \mathbf{x}^T A_Q \mathbf{x} = 0, \tag{2.7}$$

ahol $\mathbf{x}$ a pontok homogén koordinátahármasa és

$$C_Q = \begin{pmatrix} a_1 & \frac{a_2}{2} & \frac{a_4}{2} \\ \frac{a_2}{2} & a_3 & \frac{a_5}{2} \\ \frac{a_4}{2} & \frac{a_5}{2} & \frac{a_6}{2} \end{pmatrix} \tag{2.8}$$

a kúpszelet $\mathbf{a} \in \mathbb{R}^6$ paramétervektorának elemeiből képzett mátrix.

A 2.7 kifejzés értéke nemcsak a kúpszeleten lévő pontokat jelöli ki; a kúpszeleten kívül eső pontokra definiál egyfajta, a kúpszelet és a vizsgált pont között értelmezett távolságot is. Ezt a távolságot nevezzük *algebrai távolságnak.* Ezt a távolságfogalmat gyakran használják a legkisebb négyzetek módszerével való görbeillesztéskor, mert kiszámítása nem erőforrásigényes. A módszer hátránya a görbeillesztés pontatlansága, ami az illesztőpontok kúpszelet menti elhelyezkedésétől függ („high curvature bias") (Fitzgibbon és Fisher, 1995, Zhang, 1997, Ahn, 2004). Ezeknek a problémáknak a megoldására a szakirodalomban felmerült az algebrai távolság egy módosított, normalizált definíciója is, ebben az esetben a

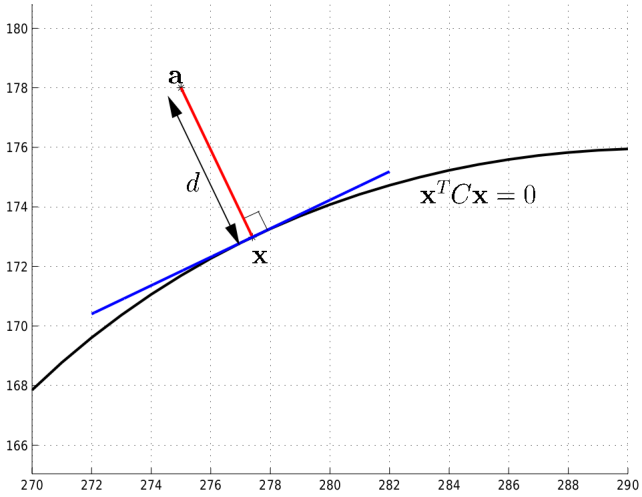
$$d_n = \frac{d_a}{\|\nabla d_a\|} \tag{2.9}$$

kifejezést minimalizáljuk (Wijewickrema et al., 2010). Ez a változat nem rendelkezik az algebrai távolság problémáival azonban meghatározása jóval számításigényesebb mint az eredeti változaté.

Geometriai távolság minimalizációja

A fentiek miatt az algebrai távolság helyett gyakran az annak hibáit kiküszöbölő geometriai (az angolszász szakirodalomban *ortogonálisnak* hívott) távolság minimalizációját választjuk. A szakirodalom ezt tekinti a legkisebb négyzetek módszeréhez használható legjobb, legtermészetesebb távolságfogalomnak (Ahn, 2004).

A kúpszeletek és pontok közti geometriai távolság minimalizációjakor azonban a megoldás numerikusan instabillá válik, emiatt a távolság kiszámítására gyakran olyan kúpszelet-specifikus geometriai tulajdonságokat használnak, mint a középpont helyzete, a féltengelyek hossza és elfordultsága (Wijewickrema et al., 2010).



2.5. ábra. *Kúpszelet és pont geometriai távolsága (*Wijewickrema *et al., 2010).*

Kúpszeletillesztő algoritmusok

A kutatás során több kúpszeletillesztő algoritmussal dolgoztunk. Elsőként (Chernov, 2010) algoritmusát vizsgáltuk meg. Tapasztalataink alapján az algoritmusra jellemző a 2.3.4 szakaszban vázolt numerikus instabilitás: amennyiben a bemeneti pontfelhő zajos, vagy a kúpszelet görbéjének csak egy része látszik (ami az általunk kutatott probléma során rendszeres), akkor az optimalizáció eredményeképp kapott kúpszelet gyakran nem illeszkedik megfelelően. Jellemző hiba volt például, hogy egy középmeridiánra szimmetrikus, ellipszisíves képzetes hengervetület (pl. a Mollweide-vetület) két, $\lambda = \pm n^\circ$ hosszúsági körére való kúpszeletillesztés során a kapott ellipszisek megközelítően sem fedték egymást – nemcsak azok középpontja tért el, de nagytengelyeik mindkét esetben a várt eredménynél jóval kisebb voltak.

A fenti algoritmus emellett az iterációk során a hibavektort (a nyomvonal pontjainak kúpszelettől számított távolságait) a kúpszelet típusától függően különböző módon számítja ki, ami további numerikus hibákat okozhat.

A fentiek miatt a kúpszeletillesztésre végül egy másik algoritmust (Wijewickrema et al., 2006 és Wijewickrema et al., 2010) használtunk. A két tanulmány egy olyan kúpszeletillesztési módszert javasol, amely a legkisebb négyzetek módszerének geometriai távolságot minimalizáló változatára épül, azonban egy új, kúpszelettípus-független módszert javasol a geometriai távolság kiszámítására, amely mentes az előző szakaszban vázolt problémáktól.

Az alábbiakban áttekintjük az algoritmus működését.

Geometriai távolságok kúpszelet-független meghatározása

Egy kúpszelet és egy pont közti geometriai távolságot a 2.5 ábrán látható módon definiáljuk, vagyis az kúpszeletre nem illeszkedő $\mathbf{x}$ pont és a kúpszeleten található $\mathbf{a}$ pont $d = \|\mathbf{x} - \mathbf{a}\|$ távolsága. Az $\mathbf{x}$ pontot úgy választjuk meg, hogy a $\mathbf{x} - \mathbf{a}$ szakasz merőleges legyen a kúpszelet $\mathbf{x}$-beli érintőjére. Bizonyítható, hogy bármely,

a kúpszelet görbéjén belül vagy kívül található **a**-ra egyértelműen meghatározható **x**.

Az **x** pont meghatározásához tekintsük a kúpszelet C mátrixának első két sorából és az **a** pont homogén koordinátahármasából képzett

$$B = \mathbf{c}_1\mathbf{a}_1^T - \mathbf{c}_2\mathbf{a}_1^T \tag{2.10}$$

kifejezést. (Az összefüggés a két pont közti vektor tulajdonságaiból (C normálvektora és egyben a legrövidebb távolság **a** és **x** között) adódik, a levezetés megtalálható az idézett tanulmányokban.) Vegyük észre, hogy $B \in \mathbb{R}^{3\times3}$, tehát B egy kúpszelet mátrixa, azonban nem szimmetrikus. Hogy azzá tegyük, képezzük a

$$D = \frac{B + B^T}{2} \tag{2.11}$$

mátrixot. Ekkor egy olyan kúpszelet áll elő, amelyen a keresett **x** pont szintén rajta van, tehát:

$$\begin{aligned} \mathbf{x}^T C\mathbf{x} &= 0, \\ \mathbf{x}^T D\mathbf{x} &= 0. \end{aligned} \tag{2.12}$$

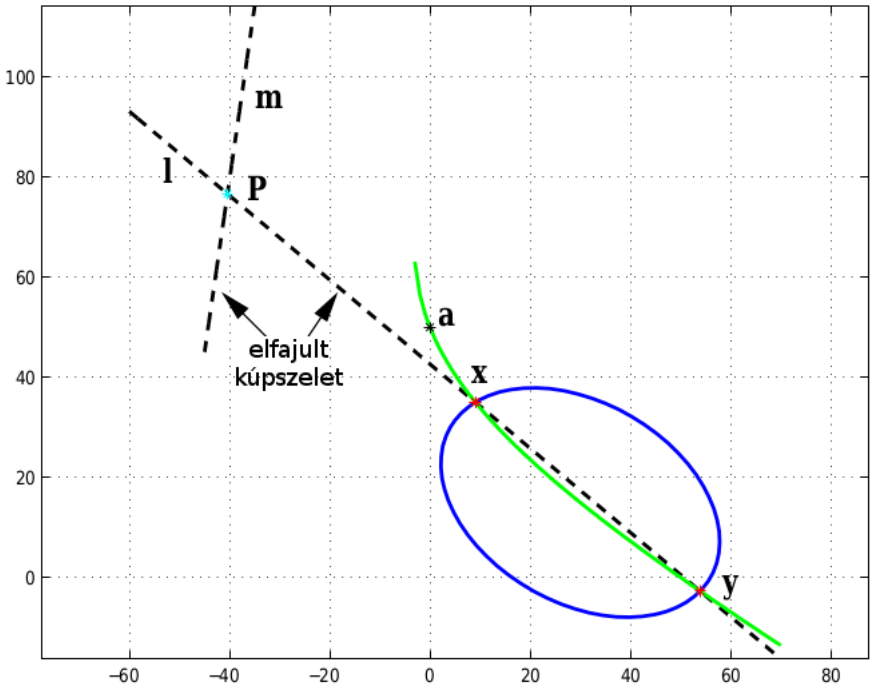
Az így kapott egyenletrendszer két kúpszelet metszéspontjainak felel meg. Általános esetben négy ilyen metszéspont van, melyek közül az **a**-hoz legközelebb esőt tekintjük a keresett pontnak. Az egyenletrendszer egy negyedfokú kifejezésre vezet, amelynek megoldása vagy iteratív úton (nemlineáris optimalizációval) vagy valamely, a kúpszelet típusától függő információ segítségével (ld. 2.3.4 szakasz) oldható meg. Ezek helyett Wijewickrema et al., 2010 egy más megoldást javasol:

Bizonyítható, hogy végtelen számú olyan kúpszelet van, amely két másik kúpszelet metszéspontjaira is illeszkedik. Ezeket a kúpszeleteket esetünkben a

$$C_f = C + \lambda D \tag{2.13}$$

kifejezés írja le, ahol λ egy skaláris paraméter. Bizonyítható, hogy a kúpszeletek fenti halmazának léteznek olyan elemei, amelyek elfajultak, azaz a kúpszelet görbéje ponttá, vagy két egybeeső vagy egymást metsző egyenessé válik. (Semple és Kneebone, 1998, idézi Wijewickrema et al., 2010). Mivel ezek leírásához nem szükséges a kúpszeletek leírására használt (2.6) másodfokú kifejezés, megkeresésükkel kikerülhető a kúpszeletek metszéspontját általános esetben leíró (2.12) negyedfokú kifejezés megoldása.

Tudjuk továbbá, hogy a fenti kúpszelethalmaz elfajult elemei átmennek C és D közös pólusain (2.6 ábra). Ezt a közös pólust megkaphatjuk a $(C-\lambda D)\mathbf{x_p} = 0$ rendszer egy



2.6. ábra. *Az eredeti kúpszelet (C), a C és* $\mathbf{a}$ *kapcsolatát reprezentáló D kúpszelet, és az ezek metszéspontjának megkereséséhez használt elfajult kúpszelet (az* $\mathbf{l}, \mathbf{m}$ *egyenespár) (*Wijewickrema *et al., 2010).*

valós sajátvektorának megkeresésével. A közös pólusnak megfelelő $\mathbf{x_p}$ sajátvektor ismeretében kiszámítható a kúpszelethalmaz egy elfajult tagjához tartozó λ_d skaláris paraméter:

$$\lambda_d = -\frac{\mathbf{x_p}^T C \mathbf{x_p}}{\mathbf{x_p}^T D \mathbf{x_p}}. \tag{2.14}$$

Az így ismertté vált $C_d = C + \lambda_d D$ elfajult kúpszelet felbontható két egyenesre (például a 2.3.3 szakaszban tárgyalt szinguláris felbontással):

$$C_d = \mathbf{l}\mathbf{m}^T + \mathbf{l}^T\mathbf{m}. \tag{2.15}$$

A két egyenes ($\mathbf{l}$ és $\mathbf{m}$) ismeretében meghatározhatóak C és D metszéspontjai, amelyek valamelyik kúpszelet és az egyenesek metszéspontjaiként adódnak. Ezek közül a – már tárgyalt módon – az $\mathbf{a}$-hoz legközelebb esőt tekintjük $\mathbf{x}$-nek. Ezek egymástól vett távolsága a keresett geometriai távolság.

Optimalizáció

Az előző szakaszban ismeretetett módon meghatározható bármely kúpszelet és pont közti geometriai távolság. Ennek a távolságdefiníciónak a felhasználásával bármely kúpszelet illeszkedése kiszámítható egy adott fokhálózati vonalhoz tartozó illesztőpontokra. A célunk ennek a hibának a minimalizálása. Az alkalmazásban a Wijewickrema et al., 2010 által javasolt *Levenberg–Marquardt* módszert (Marquardt,

1963) alkalmaztuk. (Az általunk használt konkrét implementáció megtalálható a Julia programozási nyelvhez készült `Optim.jl`[11] függvénykönyvtárban.)

A Levenberg–Marquardt módszer, bár valamivel lassabb mint más módszerek (pl. a Gauss–Newton-módszer), robosztusabb azoknál, kevésbé érzékeny a célfüggvény paramétervektorának kezdeti értékeire. A módszer feltételezi a vizsgált célfüggvény Jacobi-mátrixának ismeretét is, amelynek kiszámítását Wijewickrema et al., 2010 részletesen tárgyalja.

A paramétervektor kezdeti értékét a Fitzgibbon, Pilu, et al., 1999-ben leírt módszerrel határozzuk meg, amely az illesztőpontokra egy ellipszist illeszt.

Tapasztalataink a fenti módszerrel működő iteratív illesztés robosztusságát igazolják. A 4.1 szakaszban vázolt eredmények bemutatják, hogy a kúpszeletillesztés sok esetben magas álvéletlen zaj esetén is sikeres marad.

2.4. Egyéb algoritmusok

2.4.1. Metszéspontok vizsgálata

Bizonyos vetületek (pl. valódi síkvetületek) pontos típusának felismeréséhez szükséges a fokhálózati vonalak metszéspontjainak elhelyezkedését, az ott jelentkező metszési szögeket valamint a metszéspontok egymáshoz képesti elhelyezkedését vizsgálnunk.

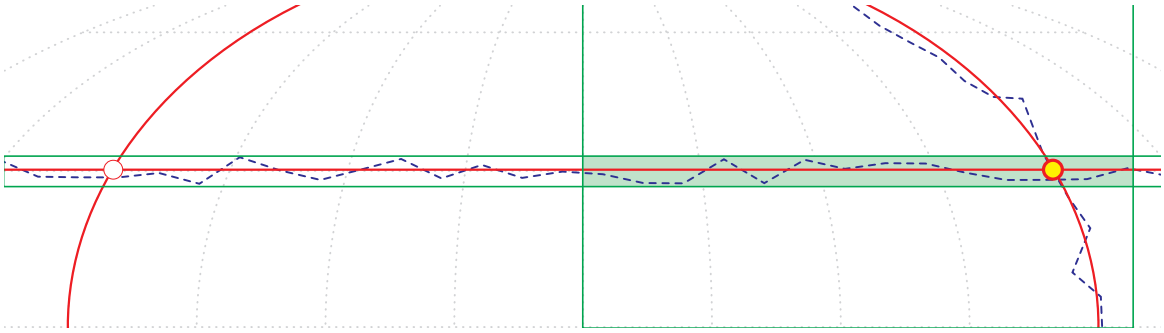
Metszéspontkeresés

A metszéspontok keresése jelenleg csak hosszúsági és szélességi körök egymással való metszéseire terjed ki, de megjegyezzük, hogy a későbbiekben – például póluspontos vetületek póluspontjainak megkereséséhez – hasznosnak bizonyulhat a hosszúsági körök közös metszéspontjainak felismerése is.

Külön vizsgáljuk a felismert két fő görbetípus (egyenes és kúpszelet) azonos illetve különböző típusú görbékkel vett metszéspontjait. Ezek alapján három eset (egyenes–egyenes, kúpszelet–egyenes, kúpszelet–kúpszelet) különíthető el. Csekély gyakorlati haszna miatt a dolgozatban a kúpszelet–kúpszelet illesztést nem tárgyaljuk. A másik két esetet külön oldjuk meg:

1. **Egyenes–egyenes metszéspontok**
   Két homogén koordinátahármassal adott egyenes metszéspontját azok vektorainak keresztszorzata $((a_1, b_1, c_1)^T \times (a_2, b_2, c_2)^T)$ adja. Az egyenesillesztés algoritmusa biztosítja, hogy a metsző egyenesek valósak. Így – amennyiben a két egyenes nem esik egybe –, a metszéspont koordinátahármasa nem az ideális pont, tehát az euklideszi sík pontosan egy pontjának feleltethető meg.

[11] `https://github.com/JuliaOpt/Optim.jl`



2.7. ábra. *Nyomvonalra illesztett görbék metszéspontjának egyértelművé tétele befoglaló téglalapok segítségével. (Kék: a fokhálózati vonal felhasználó által megrajzolt nyomvonala; zöld: a nyomvonalak által kijelölt befoglaló téglalapok – a kitöltött rész a közös metszetet jelöli; piros: a nyomvonalra illesztett görbék és ezek metszéspontjai – a sárgával jelölt metszéspont a megfelelő.)*

2. **Egyenes–kúpszelet metszéspontok**
   Képezzük az l homogén koordinátahármassal adott egyenes 3×3-as antiszimmetrikus mátrixát:

$$L = \begin{pmatrix} 0 & -l_3 & l_2 \\ l_3 & 0 & -l_1 \\ -l_2 & l_1 & 0 \end{pmatrix}. \tag{2.16}$$

   Ekkor $D = L^T \cdot C \cdot L$ egy elfajult kúpszelet (két egybeeső egyenes), amely a projektív geometria dualitása miatt két pontnak (l és C metszéspontjainak) felel meg. A két pont például az elfajult kúpszelet 2.3.3 szakaszban tárgyalt szinguláris dekompozíciójával meghatározható.

Általános esetben egy nem-elfajult kúpszeletnek és egy egyenesnek két közös pontja van – két fokhálózati vonal metszéspontja tehát a fentiek alapján nem határozható meg egyértelműen. Ezt feloldandó képezzük a két fokhálózati vonal felhasználó által megrajzolt nyomvonalainak befoglalótéglalap-metszetét és a továbbiakban az ebben elhelyezkedő pontot tekintjük a metszéspontnak (2.7. ábra).

Metszési szögek vizsgálata

Egyenesek metszési szögének vizsgálata egyértelmű, egyenes és kúpszelet metszésében azonban ehhez képeznünk kell a kúpszelet metszéspontban vett $\mathbf{l} = C\mathbf{p}^T$ érintőjét. Két, homogén koordinátákkal adott $\mathbf{l}$, $\mathbf{m}$ egyenespár metszési szögét ekkor az alábbi módon határozhatjuk meg:

$$\Theta = cos^{-1}(l_1 m_1 + l_2 m_2) \tag{2.17}$$

Az így kapott metszési szög 0 és 2π közé esik.

Metszéspontok egyenközűségének vizsgálata

Hogy megkülönböztethessük például a valódi és képzetes hengervetületek altípusait, meg kell állapítanunk, hogy bizonyos fokhálózati vonalak mentén a metszéspontok egyenközűen helyezkednek-e el. Amennyiben nem, további kérdésként merül föl, hogy az azok közti távolság a vetület kontúrja felé nő vagy csökken-e. Utóbbi problémával a dolgozat keretei között nem foglalkozunk.

Jelenleg csak az egyenes mentén elhelyezkedő metszéspontok egyenközűségét vizsgáljuk. A metszéspontok egyenesek mentén való helyzetét az egyenes vektorparaméteres egyenletével jellemezhetjük:

$$\mathbf{P} = \mathbf{P_0} + t\mathbf{r}, \tag{2.18}$$

ahol $\mathbf{P}$ jelöli az egyenes pontjait, $\mathbf{P_0}$ az egyenes egy tetszőleges pontjának helyvektora, $\mathbf{r}$ pedig az egyenes irányvektora, $\mathbf{t}$ pedig egy tetszőleges valós szám, amely $\mathbf{P}$ elhelyezkedését mutatja $\mathbf{P_0}$-hoz képest az egyenes mentén. Az egyeneseink az $[a, b, c]^T$ homogén koordinátahármasokkal adottak és figyelembe kívánjuk venni a függőleges egyenesek speciális esetét is, így az egyenesen elhelyezkedő tetszőleges pont

$$\mathbf{P_0} = \begin{cases} \left(-\frac{c}{a}, 0\right)^T & \text{ha } a \neq 0 \text{ és } b = 0 \\ \left(0, -\frac{c}{b}\right)^T & \text{egyébként} \end{cases} \tag{2.19}$$

és az egyenes irányvektora

$$\mathbf{P_0} = \begin{cases} [1, 0]^T & \text{ha } a = 0 \text{ és } b \neq 0 \\ [0, 1]^T & \text{ha } a \neq 0 \text{ és } b = 0 \\ \left(-\frac{a}{b}, -\frac{c}{b}\right)^T & \text{egyébként} \end{cases} \tag{2.20}$$

módon adódik. A metszéspont egyenes menti helyzetét ekkor a

$$t_P = \mathbf{r}^{-1} \begin{pmatrix} x_P - x_{P_0} \\ y_P - y_{P_0} \end{pmatrix} \tag{2.21}$$

paraméter jellemzi.

Egy adott egyenes mentén elhelyzekedő összes metszéspont paraméterének meghatározása után rendezzük sorba a metszéspontokat azok nem az egyeneshez tartozó földrajzi koordinátái szerint (szélességi kör esetén a hosszúságokat figyelembe véve és fordítva), majd képezzük ezeknek a különbségét:

$$\delta^{[geo]} = \{\kappa_j - \kappa_i \mid \forall i, j \in [1, N], j = i + 1\} \tag{2.22}$$

ahol N a metszéspontok száma és κ a vizsgált földrajzi koordináta. Képezzük hasonlóképpen az egyenes menti paraméterek különbségeit is:

$$\delta^{[\mathbf{t}]} = \{t_j - t_i \mid \forall i, j \in [1, N], j = i + 1\}. \tag{2.23}$$

Vizsgáljuk a paramétereltérések koordinátaeltérésekkel normalizált változatait:

$$\delta = \{\frac{\delta^{[\mathbf{geo}]^2}_i}{\delta^{[\mathbf{t}]^2}_i} \mid i \in [1, N]\} \tag{2.24}$$

Ezzel a fokhálózat menti távolságtól független különbségekhez jutunk, így akkor is vizsgálhatjuk az egyenközűséget, ha a felhasználó által megrajzolt fokhálózati vonalak nem azonos koordinátaértéknyire helyezkednek el egymástól.

Hogy megállapítsuk, a vizsgált fokhálózati vonal mentén egyenközűen helyezkednek-e el a metszéspontok, tekintsük a δ normalizált különbségek variációs koefficiensét:

$$c_v = \frac{\sigma}{\mu}, \text{ ahol } \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i - \mu^2} \text{ és } \mu = \frac{1}{N}\sum_{i=1}^{N}\delta_i. \tag{2.25}$$

A variációs koefficiens megmutatja, hányadrésze az adatsor szórása a középértékének, ezzel úgy jellemzi a vizsgált statisztikai sokaság terjedelmét, hogy annak értéke az adatsor értékeitől független legyen. A variációs koefficienst valamilyen szabadon választott határértékkel összehasonlítva eldönthetjük, hogy az adott fokhálózati vonal mentén elhelyezkedő metszéspontokat egyenközűnek tekintjük-e.

2.4.2. Kúpszeletívek koncentrikusságának vizsgálata

Ahhoz, hogy megállapítsuk, a fokhálózati vonalakat alkotó kúpszeletívek középpontjai egybeesnek-e, meg kell állapítanunk ezeknek a középpontoknak a helyzetét. Ez a probléma elsősorban az „igazi” képzetes kúpvetületek csoportja és a polikónikus és pszeudopolikónikus csoportja közti különbségtétel szempontjából fontos: míg az előbbiek esetében a parallelkörök képei koncentrikusak, addig az utóbbiak esetében nem.

A kúpszelet középpontjának azt a pontot tekintjük, ahol a kúpszelet másodfokú polinomiális alakjának (2.6 kifejezés) gradiensvektora a $(0, 0)$ értéket veszi föl:

$$\nabla C = \left(\frac{\partial C}{\partial x}, \quad \frac{\partial C}{\partial y}\right) = (0, 0) \tag{2.26}$$

A parciális deriváltakat behelyettesítve az alábbi egyenletrendszerhez jutunk:

$$\begin{cases} a_1 x_1 + \frac{1}{2} a_2 x_2 + a_4 = 0 \\ \frac{1}{2} a_2 x_1 + \frac{1}{2} a_2 x_3 + a_5 = 0 \end{cases} \tag{2.27}$$

Amennyiben a két ismeretlenes egyenletrendszer mátrixának determinánsa nem nulla (azaz a kúpszelet ellipszis vagy hiperbola), úgy a kúpszelet középpontja (a_1, a_2). Parabolákra a középpont fogalma nem értelmezhető.

A középpont minden kúpszeletre való kiszámolása után a koncentrikusság vizsgálatához meg kell állapítanunk, a kapott középpontok szórását. Amennyiben ez a szórás egy tetszőlegesen választott határértéknél kisebb, úgy a vizsgált görbék középpontját azonosnak tekintjük.

2.4.3. Póluspontosság és pólusvonalasság

Több módszer kínálkozik annak vizsgálatára, hogy a szóban forgó vetületben a pólusok pontként vagy vonalként képződnek-e le. Egy lehetőség ezek közül a meridiánok közös metszéspontjának megkeresése. Jelen dolgozatban egy ennél egyszerűbb módszert választottunk: lehetőséget biztosítunk a felhasználó számára a póluspontok kézi kijelölésére (ld. 3.2 szakasz). Amennyiben létezik póluspont, úgy ezt a meridiánok görbeillesztésénél is felhasználjuk illesztőpontként. Ha a felhasználó nem jelölt meg póluspontot, megvizsgáljuk, létezik-e $\varphi = \pm 90°$ értékű szélességi kör. Ha igen, feltételezzük, hogy a vetület pólusvonalas, ha nem, akkor a póluspontosság/-vonalasság kérdését eldönthetetlennek ítéljük.

3. Az alkalmazás bemutatása

3.1. A választott szoftverkörnyezet bemutatása és indoklása

A dolgozatban tárgyalt módszerek megvalósítására egy kliens-szerver alapon működő alkalmazást fejlesztettünk. A kétszintű szoftverarchitektúra:

- lehetővé teszi az üzleti logika és a felhasználói felület funkcionális szétválasztását,
- rákényszerít a szoftverrétegek közti interfészek szabatos specifikációjára,
- ugyanakkor szabad kezet hagy azok belső működését, implementációs részleteit illetően.

Az alábbiakban bemutatjuk az alkalmazás vázlatos szerkezetét (3.1 ábra) és a használt technológiákat.

3.1.1. Háttér

A háttérben futó, adatfeldolgozásért felelős réteg esetén az elsődleges szempont a teljesítmény volt, de szempont volt a fejlesztőkörnyezet korszerűsége és kényelme is. Ezeket a szempontokat figyelmebe véve a háttérrendszer a Julia[1] programozási nyelven készült.

A Julia első változata 2012-ben látott napvilágot egy, a Massachusets Institute of Technology-n folytatott kutatási program eredményeképp. Az azóta folyamatosan, szabad szoftverként fejlesztett nyelv legfontosabb célja, hogy olyan eszközt nyújtson a műszaki és tudományos célú programozáshoz, ami a felhasználóját nem kényszeríti kompromisszumokra: egyszerre képes a C-nyelven írt, fordított programokhoz hasonló teljesítmény elérésére, de megtartja azoknak a dinamikus típusellenőrzésű nyelveknek (MATLAB, R, Wolfram Language) az előnyeit is, amelyek a műszaki programozás területén az elmúlt időszakban jelentős népszerűségre tettek szert. Ennek megfelelően a Julia szintaxisa elsősorban a fenti nyelvekéből, valamint az olyan széles körben használt dinamikus nyelvekéből merít, mint a Python, a Lua, a Ruby és a Lisp programozásinyelv-család tagjai.

[1] `http://julialang.org`

A nyelv a MATLAB-hoz hasonlóan gyengén típusos és dinamikus típusellenőrzésű, azonban lehetővé teszi a kódban opcionális típusdeklarációk elhelyezését is. Ezt egy (LLVM segítségével megvalósított) just-in-time fordító egészíti ki – ezek kombinációjával a C-nyelvű programokét megközelítő teljesítmény érhető el.

A Julia utóbbi években elért sikeréhez ezen kívül hozzájárult az is, hogy jól illeszkedik az őt körülvevő környezetbe: szabad szoftverként bármilyen modern számítógéparchitektúrára és operációs rendszerre lefordítható, többletkód megírása nélkül támogatja C és FORTAN nyelvű függvények direkt meghívását, lineáris algebrai függvénykönyvtárának megvalósításához a de-facto szabvány LAPACK-ot használja, karakterlánc-típusa támogatja a Unicode szabványt (és annak több karakterkódolását), valamint kiterjeszthető csomagkezelő rendszerrel rendelkezik, amely a Git-verziókezelőre épül.

3.1.2. Interfész

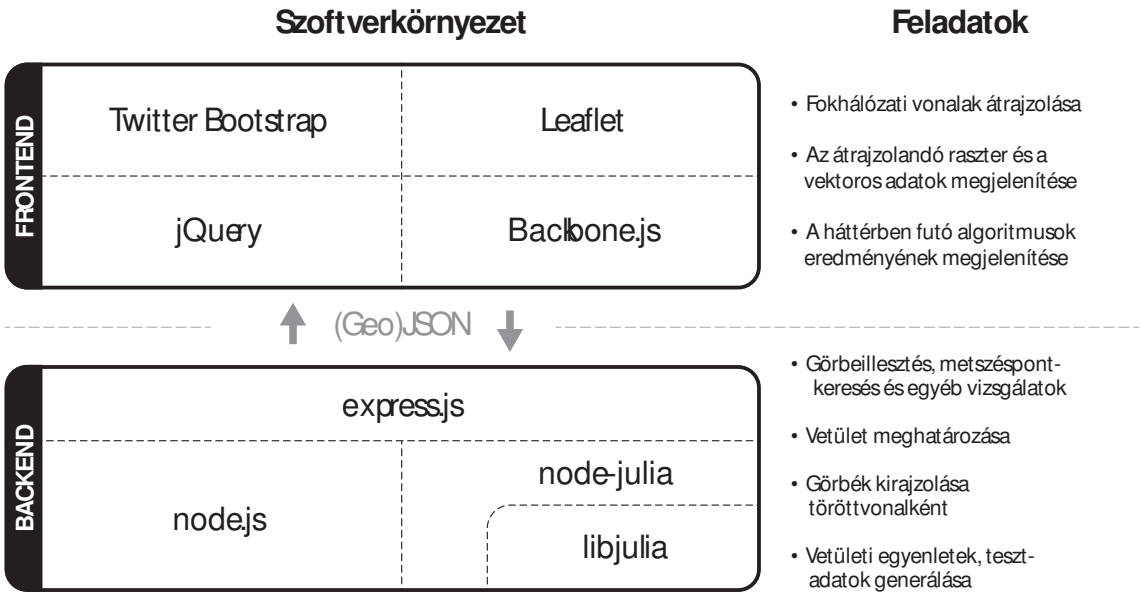
Az adatfeldolgozásért felelős réteg és a felhasználói felület közti kommunikáció a JSON[2] (JavaScript Object Notation) adatcsereszabvány segítségével történik. A JSON egy kompakt, szöveg-alapú, strukturált adatok átvitelére használt formátum, amely alkalmas ember általi olvasásra és gépi feldolgozásra is. A JSON a Javascript szabványos változatának, az ECMAScriptnek egy részhalmazára épül, de mára minden fontosabb programozási nyelvhez elérhető JSON-értelmező és -generáló függvénykönyvtár. A JSON az adatokat (számokat, karakterláncokat, logikai értékeket, tömböket és kulcs-érték-párok gyűjteményeit) egy fastruktúrába szervezve kódolja.

Az alkalmazás épít a GeoJSON[3] formátumra is, ami egy térvonatkozású adatok leírására alkalmas JSON-sémát specifikál. Az általunk tárolt és feldolgozott koordinátapárok az alkalmazás jelenlegi állapotában soha nem földi vagy térképi koordinátarendszerekben értelmezettek. Azonban mind a háttérrendszer, mind a felhasználói felületen használt Leaflet webtérkép-könyvtár rábírható egyfajta „koordinátarendszer-agnosztikus" működésre, amelyben az átvitt pozíciók egy, a megjelenítő pixelkoordinátarendszeréből képzett logikai síkkoordinátarendszerben léteznek. Ebben a környezetben a vonatkozási rendszer meghatározatlanságát szintén jól toleráló GeoJSON a legegyszerűbben használható vektoros adatcsereszabvány.

3.1.3. Felhasználói felület

A felhasználói felületet egy böngészőben futó webalkalmazásként valósítottuk meg. A döntést két szempont motiválta: Egyrészt az olyan – napjaink webfejlesztésében alapvető fontosságú – technológiákkal és keretrendszerekkel mint a node.js, az npm csomagkezelő, a jQuery, a Twitter Bootstrap, a Backbone.js, vagy a Leaflet töredékére csökkenthető a felhasználói felület prototipizálásával töltött idő. A fenti könyvtárak a webfejlesztés során gyakran használt absztrakciókat és felületi elemeket biztosítanak – levéve ezzel a fejlesztő válláról ezek újbóli és újbóli megírásának

[2] `http://json.org`
[3] `http://geojson.org`



3.1. ábra. *Az alkalmazás szerkezeti vázlata*

a terhét, minimalizálva a hibalehetőségeket és ügyelve a böngészők közti széleskörű kompatibilitásra.

Másrészt a web, mint platform teszi lehetővé a legegyszerűbben operációsrendszer- és számítógéparchitektúra-független alkalmazások készítését. A webalkalmazás nem igényel telepítést, azonnal hozzáférhető és használható, így a legtöbb felhasználó számára elérhető.

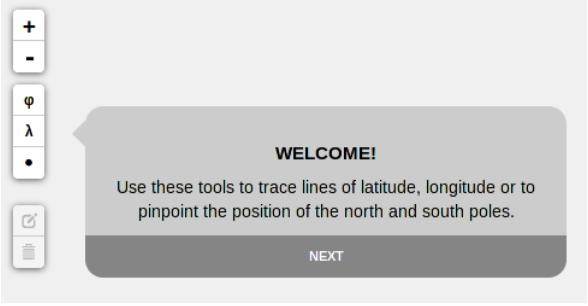
3.2. Az alkalmazás működése

A webes felületének megnyitása után az alkalmazás kétféleképpen használható: automatikusan generált fokhálózati görbék segítségével tesztelhetőek a görbefelismerés algoritmusai (4.1 szakasz), vagy egy saját raszteres térkép betöltésével elkezdhető a fokhálózati vonalak átrajzolása és illesztése.

Az alkalmazás első használatakor egy üdvözlő üzenet jelenik meg, amely végigvezeti a felhasználót a felületi elemeken és röviden bemutatja azok használatát.

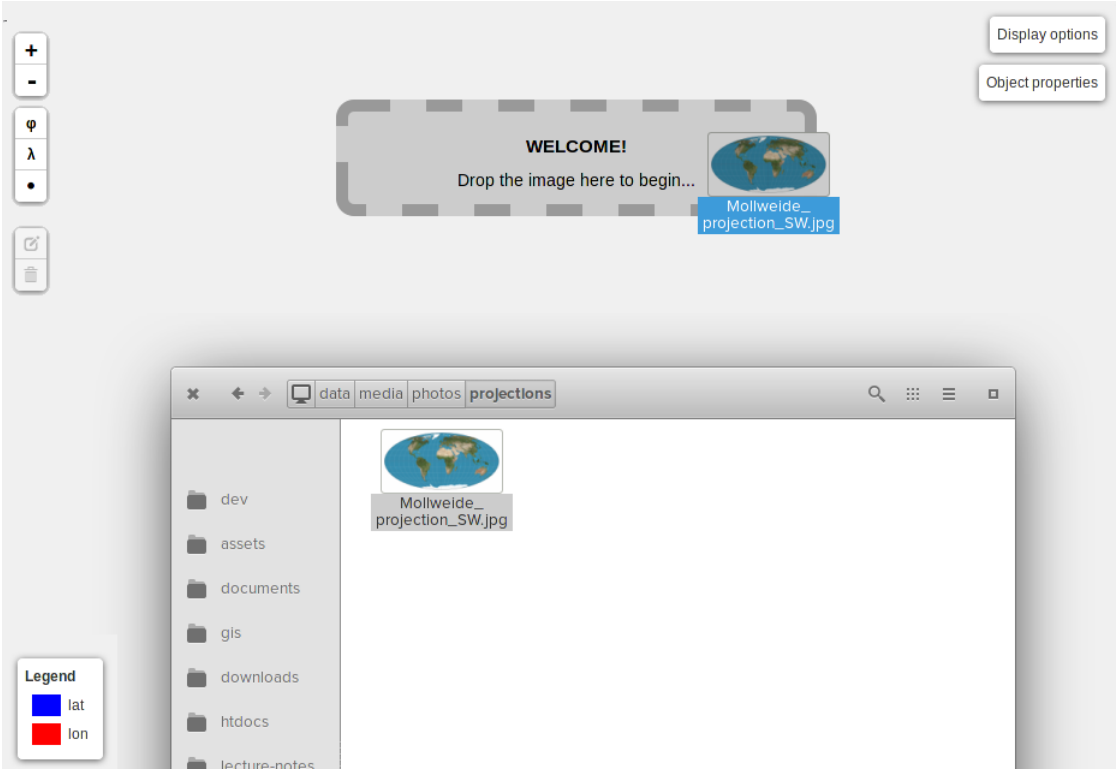
3.2.1. Fokhálózati vonalak átrajzolása

A fokhálózati vonalak átrajzolására az alkalmazás három eszközt nyújt. Ezek a térképi megjelenítéshez használt Leaflet függvénykönyvtár egy kiterjesztésére (Leaflet.draw[4]) épülnek, így jól illeszkednek a Leaflet által alapértelmezetten biztosított vezérlőelemek (pl. zoom) közé (3.3 ábra).

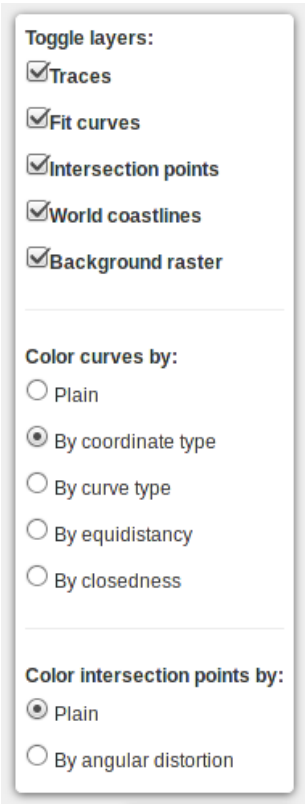


3.3. ábra. *Eszközök a fokhálózati vonalak átrajzolásához.*

[4] `https://github.com/Leaflet/Leaflet.draw`



3.2. ábra. *Az alkalmazás kezdőképernyője – átrajzolandó raszter betöltése*



3.4. ábra. *Megjelenítési beállítások*

A hosszúsági és szélességi vonalakat töröttvonalként lehet átrajzolni, míg a pólusok pontként kerülnek a térképre. Az elemek megrajzolása után egy felugró ablakban lehet megadni az adott körhöz tartozó szélességi vagy hosszúsági értéket. Mivel a kijelölt póluspontokat jelenleg csak a vetület póluspontosságának megállapítására és a görbeillesztés pontosítására használjuk, nem teszünk különbséget északi és déli pólus között. A görbeillesztés elvégzése előtt lehetőség van a már megrajzolt vonalak futásának és a pontok pozíciójának módosítására, valamint az elemek törlésére is.

3.2.2. Megjelenítési beállítások

A görbeillesztés elvégzése után a felületen lehetőség van az eredmények különböző módon való megjelenítésére. Megjeleníthetőek és elrejthetőek:

- a fokhálózati vonalak átrajzolt nyomvonalai;
- az ezekre illesztett görbék;
- a görbék metszéspontjai;
- a feltöltött raszteres kép;

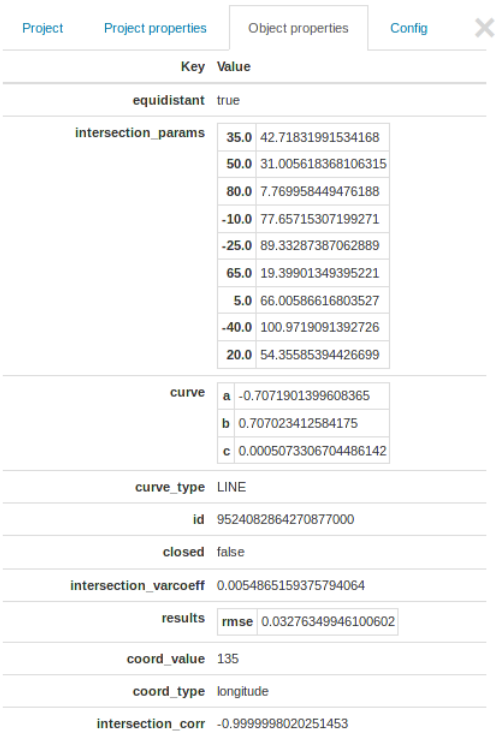
- a teszteléshez használt, automatikusan generált vetületi képek esetén egy egyszerű világtérkép.

Beállítható a fokhálózati vonalak különböző tulajdonságok alapján való színezése is:

- egy színnel;
- földrajzi koordinátatípus alapján (szélesség/hosszúság);
- görbetípus (ellipszis, kör, vonal, parabola, hiperbola, ismeretlen) alapján;
- egyenesek esetén az adott egyenes mentén elhelyezkedő metszéspontok egyenközűsége alapján;
- a görbe zártsága alapján.

Ki- és bekapcsolható végül a metszéspontok helyén a fokhálózati vonalak mentén fellépő szögtorzulás megjelenítése is (A.3 ábra). A fenti beállítások módosításával egyszerűen megjeleníthető a görbefelismerés eredménye és lehetővé válik a hibák kiszűrése.

3.2.3. Eredmények és adatok megjelenítése



3.5. ábra. *Eredmények és adatok megjelenítése az oldalsávban.*

Az alkalmazás egy oldalsávban jeleníti meg a görbeillesztés eredményeit, a különböző objektumok tulajdonságait és a konfigurációs változók értékeit. Az oldalsáv a jobb felső sarokban található gombra, vagy bármely térképi objektumra (görbék, illesztőpontok) kattintva előhívható.

A fokhálózati vonalak alábbi tulajdonságai jelennek meg:

- Objektumazonosító
- Földrajzi koordináta szerinti típus (szélesség/hosszúság)
- Földrajzi koordináta
- Görbetípus (ellipszis, kör, vonal, parabola, hiperbola, ismeretlen)
- A görbére vonatkozó (egyenes esetén három, kúpszelet esetén hat elemből álló) paramétervektor
- A görbeillesztés futásának eredménye (iterációk száma, hibanagyságok stb.)

- A görbe zártsága
- Egyenesek esetén az azon elhelyezkedő metszéspontok helyparamétere
- A metszéspontok közeinek varianciája és a földrajzi koordináta megváltozásával való korrelációjuk

Metszéspontok esetén az alábbi tulajdonságok jelennek meg:

- Objektumazonosító
- A metsző fokhálózati vonalak objektumazonosítói
- A metszéspont pozíciója
- A metsző görbék érintőjének egyenese a metszéspontban (háromparaméteres homogén alakban)
- A görbék metszési szöge

4. Eredmények

4.1. Tesztelési módszerek

Az alkalmazás tesztelését egy részben automatikus módszerrel végeztük. A vizsgált vetületek vetületi egyenleteit programkódként megvalósítottuk, majd ezek segítségével automatikusan generált fokhálózati képeket hoztunk létre, és ezekre alkalmaztuk a görbefelismeréshez használt algoritmusokat. A vetületi egyenleteket a `d3.js`[1] vizualizációs függvénykönyvtár `d3-geo-projection`[2] kiegészítéséből vettük át.

A fokhálózatot generáló algoritmus a következő beállításokkal paraméterezhető:

- **Vetülettípus** (p) – A generálandó fokhálózat vetülete. A program a kódban szereplő vetületeken kívül ismeri azoknak típusát (valódi és képzetes sík-, henger- és kúpvetületek), valamint a generálandó fokhálózat szélesség- és hosszúságbeli határait. Így kiküszöbölhetőek például a poláris helyzetű síkvetületek Egyenlítőnél, vagy a normál helyzetű Mercator-vetület pólusvonalakhoz közeli szélességeinél fellépő szingularitások miatti numerikus problémák.

- **Fokhálózati vonalak intervalluma** (n) – A fokhálózati kép vonalainak osztásköze. A sűrűbb osztásköz több vonalat, így magasabb feldolgozási időt jelent, azonban segít a program más területein előforduló hibák kiszűrésére, mint pl. a metszéspontkeresés, vagy a beállítható toleranciaértékektől függő algoritmusok.

- **Méretarány** (s) – A programban szereplő vetületi egyenletek egységsugarú gömbön értelmezettek, értékük kiszámításánál nem vesszük figyelembe az alapfelületi méreteket. Azonban a megjelenítéshez használt síkkoordinátarendszer alapértelmezett beosztása túl kis méretű az így kiszámolt vetületi koordináták szemléletes megjelenítéséhez, így ezeket egy – mindkét tengely menti – méretezéssel transzformáljuk a megjelenítő koordinátarendszerébe.

- **Zaj** (σ) – A felhasználó által megrajzolt nyomvonalak a dolgozatban korábban már tárgyalt okokból véletlenszerű hibákat tartalmaznak. Az automatikus tesztelés során ezeket a hibákat egy álvéletlen zaj hozzáadásával modellezzük. Az egyszerűség kedvéért feltételezzük, hogy a rajzolás, mint statisztikai folyamat hibái egy $\mathcal{N}(\mu = 0, \sigma)$ normáleloszlást követnek, így az algoritmusunkat a σ varianciával paraméterezzük.

[1] `http://d3js.org`
[2] `https://github.com/d3/d3-geo-projection`

Ezek alapján a teszteléshez használt fokhálózat Λ hosszúsági vonalának vetületi koordinátái az alábbi módon állnak elő:

$$P_\Lambda = \left\{ \begin{bmatrix} x_p(arc\Lambda, arc\varphi) \\ y_p(arc\Lambda, arc\varphi) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix} * s \;\middle|\; \begin{array}{c} \varphi \in \{\varphi_{min} \leq x \leq \varphi_{max} \mid x \bmod n = 0\} \text{ és} \\ \epsilon_{1,2} \sim \mathcal{N}(\mu = 0, \sigma) \end{array} \right\}, \tag{4.1}$$

míg a Φ szélességi körök:

$$P_\Phi = \left\{ \begin{bmatrix} x_p(arc\lambda, arc\Phi) \\ y_p(arc\lambda, arc\Phi) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix} * s \;\middle|\; \begin{array}{c} \lambda \in \{\lambda_{min} \leq x \leq \lambda_{max} \mid x \bmod n = 0\} \text{ és} \\ \epsilon_{1,2} \sim \mathcal{N}(\mu = 0, \sigma) \end{array} \right\}. \tag{4.2}$$

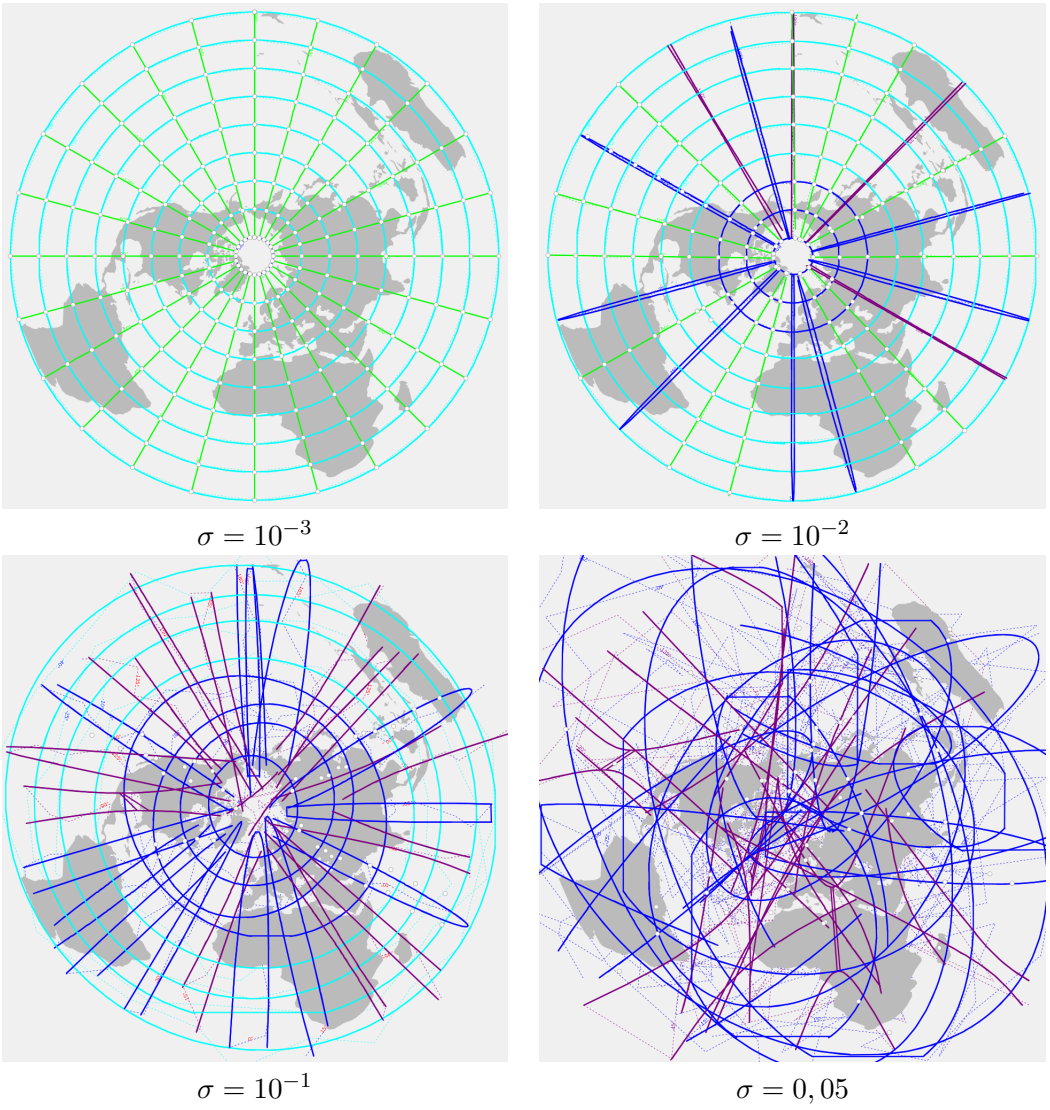
4.2. A tesztelés során felmerült problémák

4.2.1. A zaj hatása a görbefelismerésre

A tesztelés során azt tapasztaltuk, hogy a görbefelismerés sikere – nyilvánvalóan – fordítottan arányos az álvéletlen zaj varianciájával. A görbeillesztési algoritmus sajátosságainak köszönhetően a zaj növelésével először az egyenes vonalak felismerése hiúsul meg (4.1 ábra). A következő lépésben elvégzett kúpszeletillesztés ezekben az esetekben is sikeres: ez a legtöbb esetben olyan ellipszis- és hiperbolaíveket eredményez, amelyek nagytengelye az egyenes szakasz hosszával egyezik meg, míg kistengelyük elhanyagolható hosszúságú. Megfigyelhető továbbá, hogy (valószínűleg a zaj eloszlásának normális volta miatt) a zárt körívek a zaj növelésével is viszonylag sokáig felismerhetőek maradnak.

A zaj növelésével és az egyenes vonalak felismerhetetlenné válásával a metszéspontkeresés is meghiúsul, mivel az alkalmazás jelenlegi állapotában csak egyenesek és kúpszeletek közti metszéspontokat vizsgálunk, kúpszelet–kúpszelet metszétpontokat nem.

A 4.1 ábrán látható továbbá, hogy a $\sigma = 10^{-1}$ és $\sigma = 0,05$ esetben a nyomvonalakhoz adott zaj jóval meghaladja azt az ésszerű mértéket, amely egy átlagos felhasználót, és a nyomvonal rajzolásához használt mutatóezközt (egeret) figyelembe véve várható. (A webes felületen az átrajzolni kívánt térkép tetszőlegesen nagyítható, amely így további pontosítást tesz lehetővé.) Egy ténylegesen a felhasználó által rajzolt nyomvonalat tekintve a zaj mértéke várhatóan inkább a 4.1 ábrán $\sigma = 10^{-3}$ és $\sigma = 10^{-2}$ értékekkel jelölt tartományba esne. Látható, hogy – a következő szakaszban tárgyalt paraméterek jó megválasztásával – az algoritmus robosztus, jól használható eredményt nyújt átlagosan zajos környezetben.



4.1. ábra. *Az álvéletlen zaj hatása a görbefelismerés sikerére (Postel-féle síkvetület).*

4.2.2. Numerikus toleranciák és egyéb paraméterek hatása a görbefelismerésre

Megjegyezzük, hogy a görbefelismerést végző algoritmusaink több esetben olyan – konfigurálható – numerikus toleranciákra támaszkodnak, amelyek értékei alapvetően befolyásolhatják azok működésének zajfüggését. Ilyen paramétertől függ:

- Az egyenesillesztés sikeressége;
- a kúpszeletillesztés során a konkrét kúpszelettípus (ellipszis, parabola stb.) felismerése;
- ellipszisek és körök közti különbségtétel;
- a görbe zártságának megállapítása;
- több metszéspont közül a valós metszéspont kiválasztása;

- a metszéspontok egyenközűségének vizsgálata;
- az egyenes vonalak függőlegességének vizsgálata (azok kirajzolásához).

Az alkalmazás jelenlegi állapotában ezeknek a paramétereknek az értéke kézzel határozható meg. Az algoritmus egy lehetséges jövőbeni javítása lehet ezen paraméterek automatikus beállítása, oly módon, hogy az:

- független legyen a nyomvonal koordinátaértékeinek nagyságától;
- figyelembe vegye a pontokat terhelő zaj átlagos mértékét;
- a lehető legkevesebb téves görbefelismerést produkálja.

4.3. Vetületek felismerhetősége

A vetületek felismerése során a zaj okozta problémákon kívül egyéb problémák is felmerülhetnek, amelyek részben vagy egészében megakadályozhatják a fokhálózati kép és így a vetület típusának felismerését.

4.3.1. Szinuszíves vetületek felismerése

Az Érdi-Krausz által megalkotott rendszer képzetes hengervetületeket magába foglaló főcsoportján belül megkülönböztethető vetületek egy csoportja, amelynek meridiánképei szinuszívek. Ezek felismeréséhez igénybe vehetnénk egy külön szinuszívillesztő algoritmust, azonban azt tapasztaltuk (például az Érdi-Krausz féle rendszerben nem szereplő Craster-féle parabolikus vetület és a Mercator–Sanson-vetület összehasonlításánál), hogy a szinuszíves fokhálózati vonalakra jellegüknél fogva szinte teljesen pontosan illeszthető egy parabola- vagy hiperbolaív. Az illesztés ezekben az esetekben mindig pontos annyira, hogy belül essen az illesztés sikerességét vizsgáló, reálisan megválasztott numerikus toleranción.

Ez a gyakorlatban azt jelenti, hogy jelenleg a szinuszíveket nem tudjuk megkülönböztetni a kúpszeletektől. Azonban észrevéve azt, hogy az Érdi-Krausz féle rendszerben nem szerepel olyan vetület, amelynek meridiánjai parabola- vagy hiperbolaívek lennének, ezek az esetek kiküszöbölhetőek azzal, ha a parabolikus vagy hiperbolikus meridiániveket szinuszívnek tekintve haladunk tovább a döntési fán.

4.3.2. Nem azonosítható görbék felismerése

Az Érdi-Krausz féle rendszer több olyan vetületet is felsorol, amelyek fokhálózati vonalai sem egyenesek, sem kúpszeletek (vagy szinuszívek). Ilyen például Eckert I. és II. vetülete, minden összetett képzetes hengervetület és az utolsó főcsoportba sorolt vetületek. Pusztán matematikai szempontokat figyelembe véve ezekre a fokhálzóati vonalakra is illeszthető olyan egyenes és kúpszelet, amely egy választott távolságmetrikát figyelembevéve a lehető legjobb illeszkedést produkálja. Az illeszkedésre vonatkozó hibaérték azonban ilyenkor nagyon magas értéket vesz föl. Hogy ezeket az eseteket elkülönítsük, definiálnunk kell egy olyan numerikus toleranciát, amelynél ha az illesztés hibája nagyobb, akkor a görbét ismeretlennek tekintjük.

4.3.3. A javasolt módszer alkalmazhatósága az Érdi-Krausz féle rendszer vetületeire

Végül megvizsgáltuk a dolgozatban javasolt módszer alkalmazhatóságát az Érdi-Krausz féle vetületmeghatározási rendszer vetületeire. A 4.1. táblázatban a vetületeket a Györffy János által átalakított (Györffy, 2012) változat főcsoportjai szerint rendszereztük. A táblázat első oszlopában szereplő jelek megmutatják, hogy az alkalmazás jelenlegi állapotában az adott vetület felismerhető-e:

✓	A vetület felismerhető.
•	A vetület az alkalmazás jelenlegi állapotában nem ismerhető fel pontosan, de a felismerésnek nincs elvi akadálya.
×	Nem várható, hogy az általunk javasolt módszerrel a vetület felismerhető lesz.

Azoknál a vetületeknél, amelyeket jelenleg nem tudunk felismerni, a kudarc okát három nagy kategóriába sorolhatjuk (lásd a 4.1. táblázat „Megjegyzés” oszlopát):

①	A vetület azért nem ismerhető fel, mert jelenleg csak a metszéspontok egyenközűségét ismerjük fel, az osztásközök megváltozásának tendenciáit (növekvő/csökkenő) nem.
②	A felismerést a szinuszívek illesztésével kapcsolatos problémák hátráltatják (lásd 4.3.1 szakasz).
③	Összetett görbéket és egy pontban megtörő, egyébként egyenes fokhálózati vonalakat nem tudunk felismerni (lásd 4.3.2 szakasz).

	Vetület neve	Megjegyzés
	1. Normális valódi hengervetületek	
✓	Meridiánokban hossztartó valódi hengervetület	
•	Területtartó valódi hengervetület	(1)
•	Szögtartó valódi hengervetület	(1)
	2. Normális képzetes hengervetületek	
✓	Apianus I. vetülete	
✓	Ortelius vetülete	
✓	Apianus II. vetülete	
✓	van der Grinten III. vetülete	
✓	Mollweide-vetület	
✓	Eckert III. vetülete	
✓	Kavrajszkij II. vetülete	
✓	Eckert IV. vetülete	
×	Baranyi II. vetülete	(1)
×	Baranyi IV. vetülete	(1)
×	Robinson-féle vetület	(1)
•	Mercator–Sanson-vetület	(2)
•	Eckert V. vetülete	(2)
•	Eckert VI. vetülete	(2)
•	Kavrajszkij I. vetülete	(2)
×	Érdi-Krausz-féle vetület	(3)
×	Goode-féle vetület	(3)
×	Eckert 1. vetülete	(3)
×	Eckert 2. vetülete	(3)
×	Donis-féle vetület	(3)
×	Collignon-féle vetület	(3)
	3. Normális valódi síkvetületek	
✓	Postel-féle meridiánokban hossztartó valódi síkvetület	
•	Lambert-féle területtartó síkvetület	(1)
•	Ortografikus síkvetület	(1)
•	Sztereografikus síkvetület	(1)
	4. Normális képzetes síkvetületek	
✓	Normális képzetes síkvetület	
	5. Normális valódi kúpvetületek	
✓	Meridiánokban hossztartó valódi kúpvetület	
•	Albers-féle területtartó kúpvetület	(1)
•	Lambert-féle területtartó kúpvetület	(1)
•	Szögtartó valódi kúpvetület	(1)

	Vetület neve	Megjegyzés
	6. Képzetes kúpvetületek	
✓	Bonne-féle vetület	
✓	Polikónikus vetület	
✓	Pszeudopolikónikus vetületek	
	7, 8. Gnomonikus síkvetület	
✓	Transzverzális gnomonikus síkvetület	
✓	Ferdetengelyű gnomonikus síkvetület	
	9. Minden fokhálózati vonal ellipszisíves	
×	Ferdetengelyű ortografikus vetület	(3)
×	Raisz Ervin-féle Armadillo vetület	(3)
	von der Mühl vetületcsalád	
✓	Littrow-féle vetület	
	11. Egyéb vetületek	
✓	Aitoff vetülete	
•	Hammer vetülete	(1)
•	Transzverzális síkvetületek	(1)
•	Transzverzális hengervetületek	(1)
×	Winkel-féle (Winkel Tripel) vetület	(3)
×	Ferdetengelyű síkvetületek	(3)
×	Ferdetengelyű kúpvetületek	(3)
×	Ferdetengelyű hengervetületek	(3)
×	Transzverzális kúpvetületek	(3)

4.1. táblázat. *Az Érdi-Krausz által javasolt rendszer vetületeinek felismerhetősége a dolgozatban leírt módszerrel.*

5. Összefoglalás

5.1. Elért eredmények, tapasztalatok, a munka értékelése

Dolgozatunkban megvizsgáltuk az ismeretlen térképi vetületek felismerésének problémáját. A probléma automatizálása – mint láthattuk – jelentős gyakorlati haszonnal kecsegtet. Kielégítő megoldása lehetővé tenné, hogy régi térképeinkre ne csak a kultúrkincs részeként tekintsünk, hanem szabatos georeferencia automatikus hozzárendelésével fel tudjuk őket használni térképszerkesztési alapanyagként és geoinformációs rendszerekben is. A vetület megállapítása és a georeferálás ma vagy kézzel végezhető karometriai mérésekkel (Érdi-Krausz, 1958) lehetséges, vagy a – jóval gyakoribb –, a térkép eredeti vetüli tulajdonságait figyelmen kívül hagyó, illesztőpontpárok vizsgált és referenciatérképi képeire támaszkodó interpolációs technikával, amely a legtöbb asztali GIS szoftverben megtalálható.

Az utóbbi módszer alapvető hátránya azonban, hogy érzékeny az illesztőpontok egymáshoz viszonyított helyzetére és számára, valamint nem megfelelően használva olyan torzulások léphetnek föl, amelyek a georeferált térképet pontatlanná vagy használhatatlanná tehetik. Valószínűleg ez a probléma és az egyre nagyobb és egyre szélesebb körben rendelkezésre álló számítási kapacitás megjelenése motiválta olyan módszerek kidolgozását az elmúlt években, amelyek egy számítógép segítségével képesek az ismeretlen vetületek félautomatikus meghatározására. Három ilyen módszert vizsgáltunk meg, melyek közül kettő az akadémiai szférában fogant, így szabad szoftverként is hozzáférhető (Jenny és Hurni, 2011, Bayer, 2014). Ezeket összehasonlítva láttuk, hogy bár konkrét implementációjukban eltérnek, alapgondolatuk ugyanaz: haladjunk végig vetületek egy lehetséges listáján, állapítsuk meg a vetület kontrollpontjainkra való illeszkedését, majd rendeljünk egy értéket az illeszkedés jóságához, amely a vetület egészét jellemzi.

Dolgozatunkban egy olyan megoldást javasoltunk, amely a fenti paradigmától alapvetően eltér: az Érdi-Krausz György által javasolt, majd Györffy János és Gede Mátyás által továbbgondolt vetületosztályozási rendszert alapul véve a vetületek fokhálózati vonalainak képét vizsgáljuk meg. Ezek alakja és egyéb tulajdonságai alapján egy döntési fán végighaladva egyre pontosabban állapíthatjuk meg a vizsgált vetület típusát. A pontos vetületi típus meghatározása után lehetőségünk adódik a konkrét vetületi paraméterek kiszámítására is.

Bemutattuk a módszer más módszerekkel szembeni előnyeit és hátrányait, majd vázoltuk a probléma megoldásához szükséges alkalmazott matematikai módszereket

és algoritmusokat. Elkészítettünk egy webes felhasználói felülettel rendelkező konkrét implementációt is, amelyet szintetikusan generált fokhálózati képek segítségével teszteltünk.

Az elkészült munka nem nyújt teljes körű megoldást a vizsgált problémára, mert erőforrások hiányában a jelenlegi dolgozat csak a fokhálózati kép vizsgálatára koncentrál, de nem tárgyalja:

- a döntési fa megvalósítását;
- a konkrét vetületi paraméterek meghatározását;
- a módszer valódi térképekre való alkalmazhatóságát.

Ezzel együtt a szintetikus tesztek során úgy találtuk, hogy az általunk javasolt módszernek bár vannak hibái, az alapjául szolgáló algoritmusok robosztusak és gyorsak, így továbbfejlesztésre érdemesek.

5.2. További kutatási lehetőségek

Véleményünk szerint a dolgozatban javasolt módszer több módon is kiterjeszthető:

- **A fokhálózati vonalak menti osztásközök alaposabb vizsgálatával.** A vetületi osztályozásra használt legfontosabb módszerek közül szinte mindent megvalósítottuk, azonban a metszéspontok egyenközűségét vizsgáló algoritmusunk hiányos. A pontos vetületi típus felismeréséhez szükséges nemcsak az egyenközűség vizsgálata de az is, hogy a nem egyenközű körök metszéspontjainak osztásköze a kör mentén csökken vagy nő-e.
- **A döntési fa implementációjával.** Alapvető fontosságú, hogy a fokhálózati kép dolgozatban tárgyalt módszerekkel megállapított tulajdonságai alapján azt be tudjuk sorolni valamely konkrét vetületi kategóriába.
- **A pontos vetületi paraméterek kiszámításával.** Érdi-Krausz, 1958 a vetületosztályozási rendszer bemutatásán kívül támpontokat ad egy-egy konkrét vetület paramétereinek kartometriai úton való kiszámítására. Ezek implementációjával az alkalmazás képessé válna nemcsak a vizsgált térképek vetületének meghatározására, de szabatos, GIS rendszerekben felhasználható georeferencia megállapítására is.
- **A numerikus toleranciák zajfüggő megválasztásával.** A
- **A felhasználói felület továbbfejlesztésével.** Az alkalmazás jelenlegi felhasználói felülete kevés segítséget nyújt annak használatához: eszközeinek körét, megjelenését inkább a fejlesztés közben felmerülő, mintsem a végfelhasználói igények motiválták. Az alkalmazás kényelmes használatához ezért a felület átalakítása szükséges.

- **A fokhálózati vonalak képeinek automatikus felismerésével a számítógépes látásban használt technológiák segítségével.** A raszteres térképek fokhálózati vonalainak gépi felismerése a félautomatikus módszer teljesen automatikussá tehetné, nagyban csökkentené az egy-egy térkép georeferálásához szükséges időt, valamint megnyitná az utat térképek kötegelt feldolgozásához is.

- **A módszer nagyobb méretarányú térképeken való vizsgálatával.** A dolgozatban vizsgált módszerünket csak világtérképek fokhálózati képein teszteltük. Mivel a használt algoritmus elsősorban a fokhálózati vonalak görbetípusának felismerésére épít, nem reális cél annak közepesnél nagyobb méretarányú térképeken való használata. A méretarány növekedésével és a kivágat méretének csökkenésével a fokhálózati görbék hamar felismerhetetlenné válnak – azonban jelenleg nem ismert, hol helyezkedik el pontosan ez a határ.

- **Az Érdi-Krausz féle hierarchia kiterjesztésével.** A dolgozatban javasolt módszer hátránya, hogy néhány esettől eltekintve nem veszi figyelembe a vetületek ferdetengelyű és transzverzális helyzeteit. Érdemes lehet megvizsgálni, vannak-e olyan feredetengelyű és transzverzális vetületek, amelyeket a hierarchia jelenleg nem tartalmaz, de geokartográfiai jelentőségük okán fontos lehet azokat felismernünk.

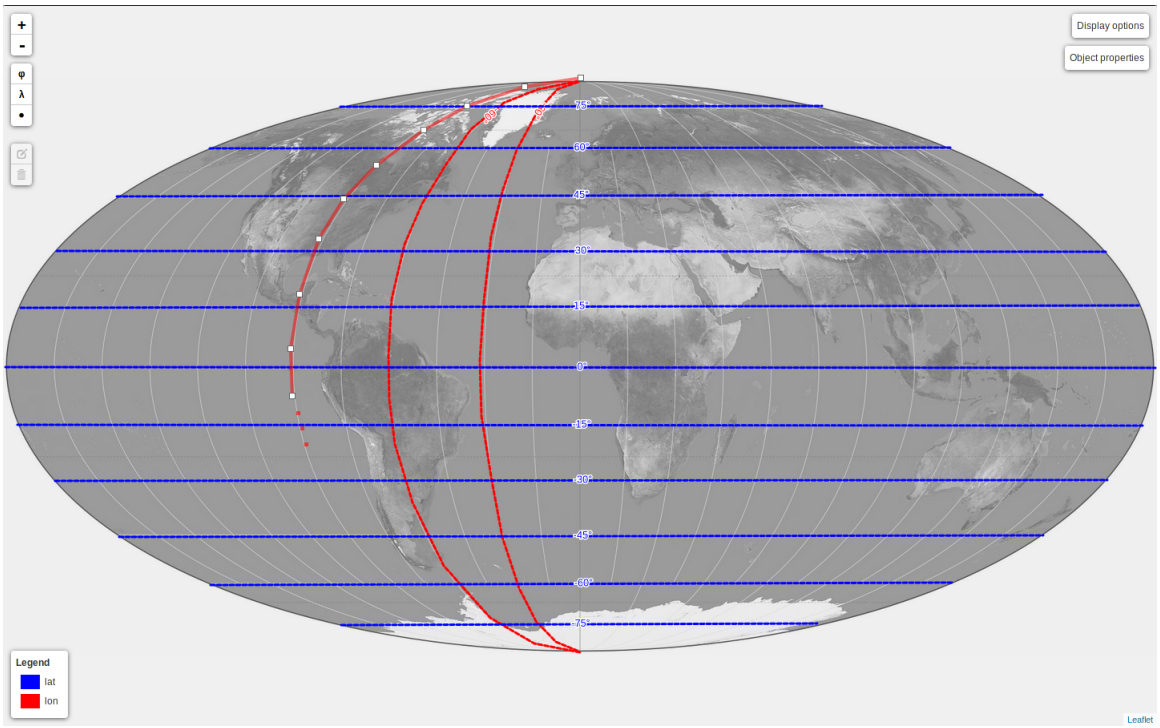
5.3. Megjegyzés szoftverlicenccel kapcsolatban

A kutatás során elkészített szoftvert a GNU General Public Licenc (GPLv3)[1] feltételei alatt nyílt forrású szoftverként elérhetővé tesszük. A licenc egy példánya és a szoftverhez kapcsolódó minden dokumentáció és forráskód elérhető a szakdolgozathoz csatolt CD-lemezen, vagy letölthető a
`https://github.com/brncsk/graticule-fitting` címről.

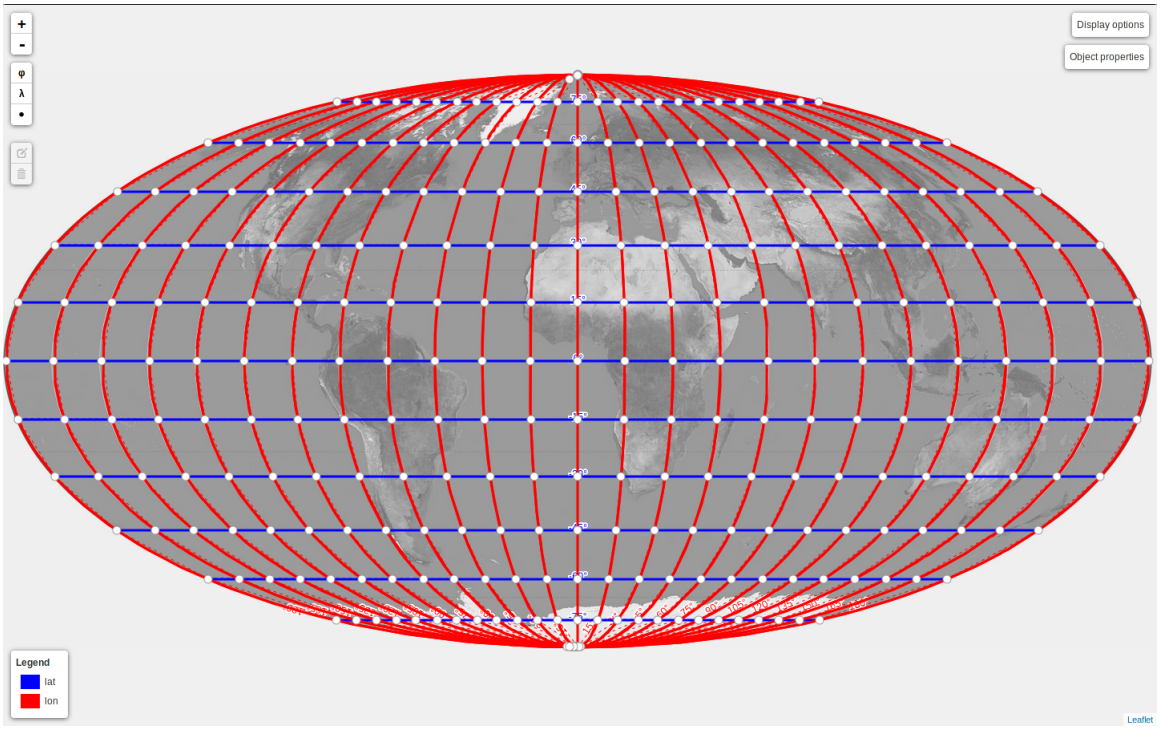
[1] `http://gplv3.fsf.org/`

A. Mellékletek

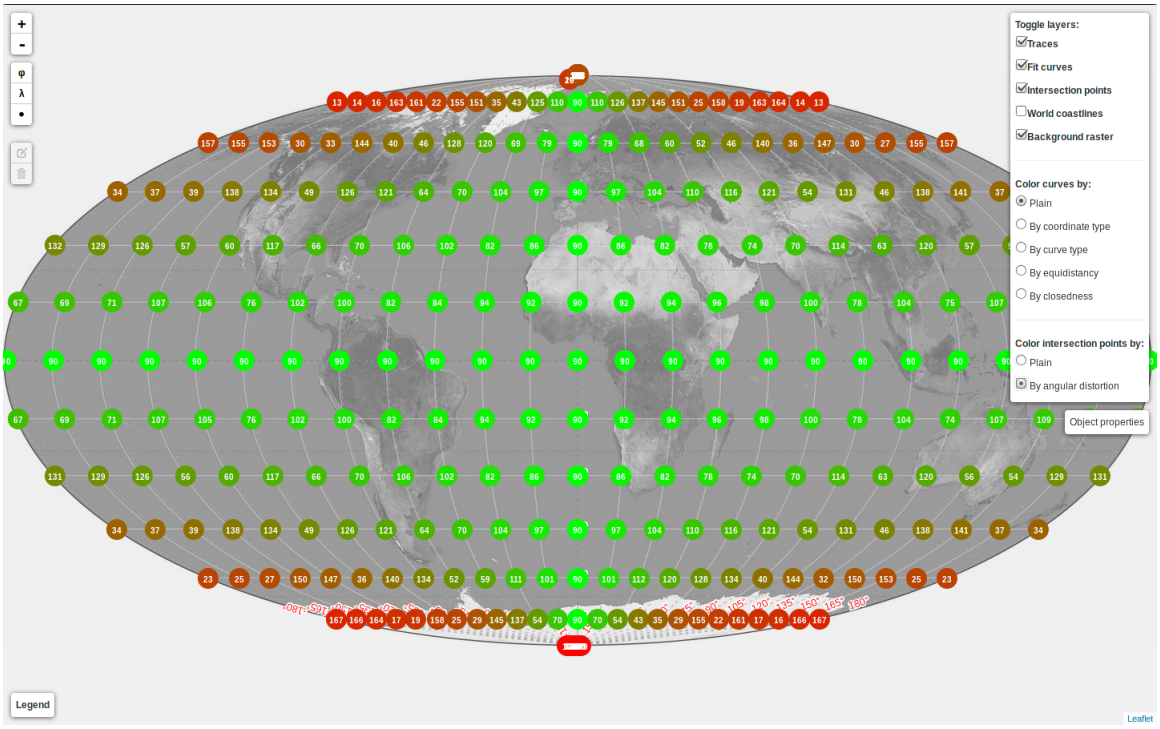
A.1. Képernyőképek



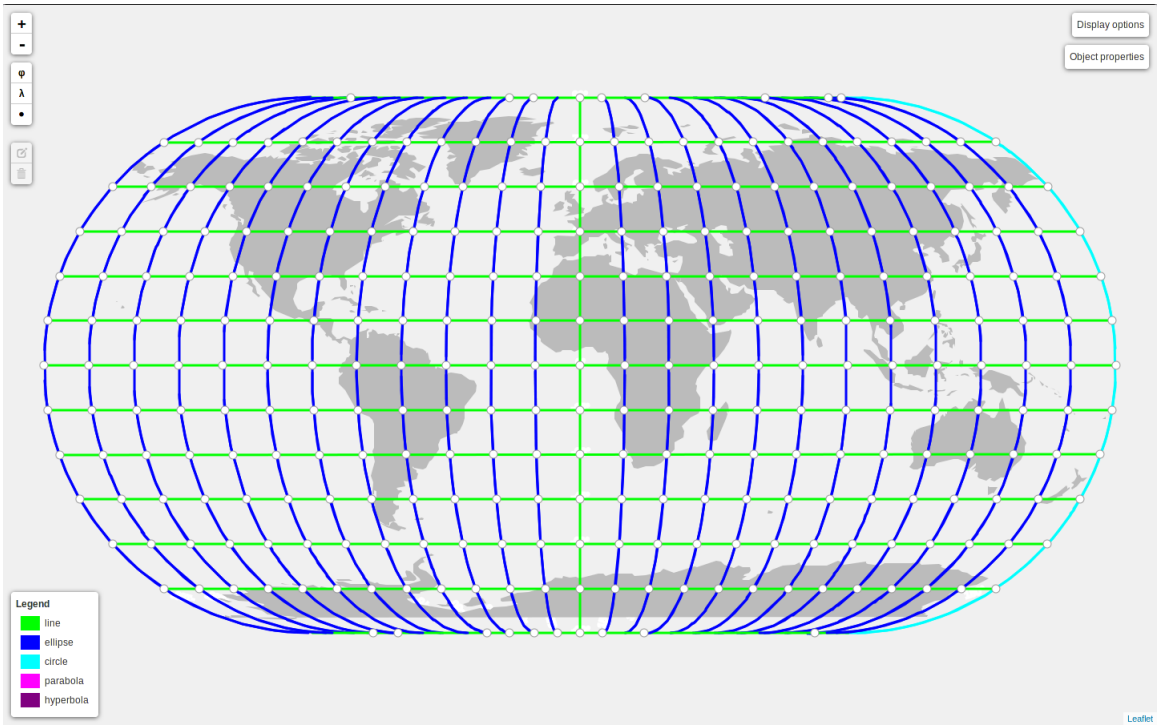
A.1. ábra. *Görbék nyomvonalának megrajzolása egér (vagy más mutatóeszköz) segítségével a feltöltött raszteres térképen.*



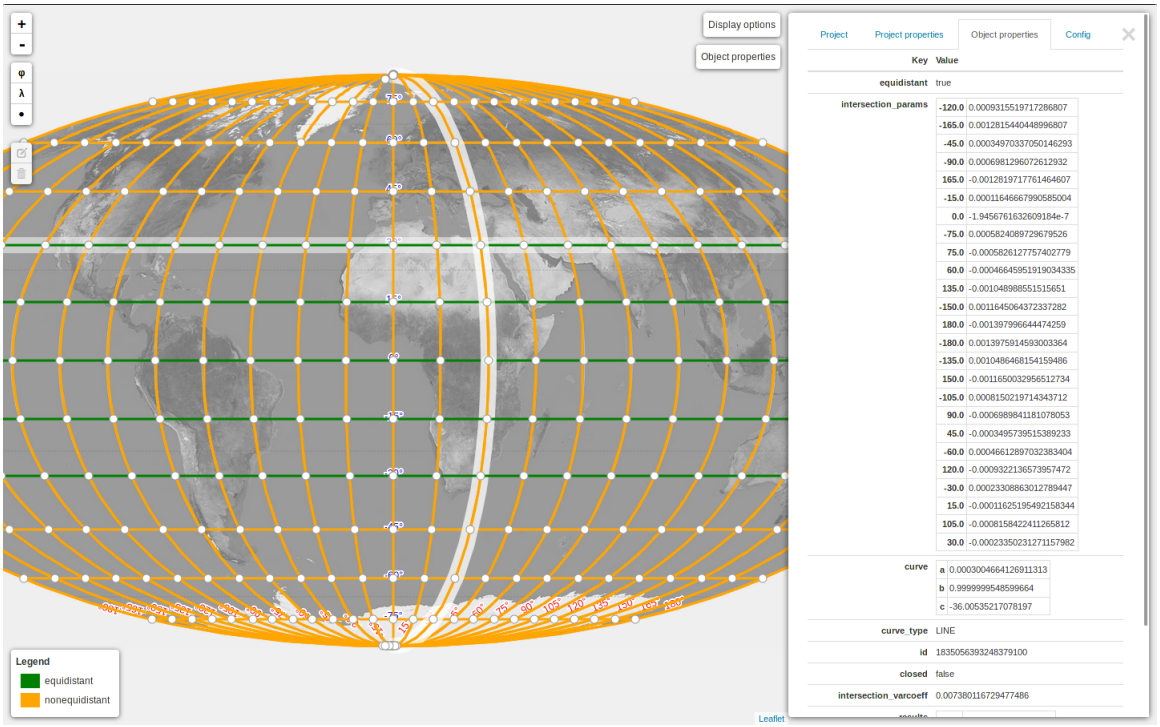
A.2. ábra. *Egy sikeres görbeillesztés eredménye a nyomvonalak megrajzolása után.*



A.3. ábra. *A fokhálózat menti szögtorzulások megjelenítése.*



A.4. ábra. *Tesztelés céljára automatikusan generált fokhálózati kép (Eckert 3. vetülete) és az illesztett görbetípusok kijelzése.*



A.5. ábra. *A fokhálózati vonalak egyéb tulajdonságai: metszéspontok egyenközűsége egy egyenes mentén.*

A.2. A dolgozathoz kapcsolódó programkód könyvtárstruktúrája

Alább – a teljesség igénye nélkül – közöljük a szakdolgozathoz tartozó CD-lemezen és a 5.3 szakaszban feltüntetett weboldalon található forráskód és a kapcsolódó adatok könyvtárstruktúrájának legfontosabb elemeit.

`app` – Az alkalmazás forráskódja

- `backend` – A háttérben futó, feldolgozást végző alkalmazásrész
  - `js` – Az `express.js` alapú szerver
    - `app.js` – HTTP-kérések belépési pontjai
    - `nj.js` – A `node-julia` biztosította funkcionalitás körüli wrapper
  - `backend` – A háttérben futó, feldolgozást végző alkalmazásrész
    - `config.jl` – Konfigurációs értékek betöltése
    - `direct.jl` – Kúpszeletillesztés kezdeti paramétervektorának meghatározása
    - `frontend.jl` – A `node-julia` által meghívott függvények belépési pontja
    - `gen.jl` – Fokhálózati képek automatikus generálása
    - `GraticuleFitting.jl` – Az alkalmazást magában foglaló Julia modul deklarációja
    - `intersect.jl` – Metszéspontkeresés
    - `leastsquares.jl` – Görbeillesztés legkisebb négyzetek módszerével
    - `plotting.jl` – Görbék kirajzolása
    - `projdefs.jl` – Vetületi definíciók
    - `projections.jl` – Általános keret vetületek definiálásához
    - `startup.jl` – Futási környezet konfigurációja parancssoros eléréshez
    - `types.jl` – Típusdeklarációk
    - `utils.jl` – Kiegészítő függvények
- `frontend` – A felhasználói felületet biztosító webalkalmazás
  - `styles` – Stíluslapok
    - `main.scss` – A webalkalmazáshoz tartozó fő stíluslap

- `scripts` – A felhasználói felületet alkotó JavaScript alkalmazás forráskódja
  - `app.js` – Az alkalmazás belépési pontja
  - `controls.js` – A felhasználói felület vezérlőelemei
  - `layers.js` – A megjelenítéshez használt térképi rétegek
  - `utils.js` – Kiegészítő függvények
  - `models` – Modellek
    - `Project.js` – Egy önálló vetületfelismerési munkafolyamatot reprezentáló objektum
  - `views` – A felhasználói felület elemeit reprezentáló objektumok
    - `MapView.js` – Térképi nézet
    - `ObjectPropertiesView.js` – Objektumok tulajdonságait megjelenítő nézet
    - `SidebarView.js` – Oldalsáv
    - `WelcomeView.js` – Az alkalmazás bemutatása

`data` – Az alkalmazáshoz kapcsolódó adatok
`tools` – Eszközök a futási környezet konfigurációjához

Felhasznált irodalom

Ahn Sung Joon (2004). *Least squares orthogonal distance fitting of curves and surfaces in space*. Vol. 3151. Springer Science & Business Media.

Bayer Tomáš (2014). "Estimation of an unknown cartographic projection and its parameters from the map". In: *Geoinformatica* 18.3, pp. 621–669.

Bayer Tomáš, Potůčková Markéta, Čábelka Miroslav (2010). *Cartometric Analysis of Old Maps on the Example of Vogt's Map*. Springer.

Bookstein Fred L (1979). "Fitting conic sections to scattered data". In: *Computer Graphics and Image Processing* 9.1, pp. 56–71.

Chernov Nikolai (2010). *Circular and linear regression: Fitting circles and lines by least squares*. CRC Press.

Érdi-Krausz György (1958). "Vetületanalízis". In: *Térképtudományi Tanulmányok (Studia Cartologica)*. Honvédelmi Minisztérium Térképészeti Intézet, Budapest, pp. 194–270.

Fitzgibbon Andrew, Fisher Robert B. (1995). "A Buyer's Guide to Conic Fitting". In: *Proceedings of the British Machine Vision Conference*, pp. 513–522.

Fitzgibbon Andrew, Pilu Maurizio, Fisher Robert B (1999). "Direct least square fitting of ellipses". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21.5, pp. 476–480.

Fogaras Dániel (2002). *Singular Value Decomposition and its Applications in Information Retrieval*. Tech. rep.

Gede Mátyás, Barancsuk Ádám (2015). "Determining the Projection of Small Scale Maps Based on Grid Line Shapes". In: *Proceedings of the 10th Conference and Workshop on Digital Approaches to Cartographic Heritage*.

Györffy János (2012). *Térképészet és Geoinformatika II. – Térképvetületek*. Ed. by István Klinghammer. ELTE Eötvös Kiadó. ISBN: 978-963-312-138-2.

Jenny Bernhard, Hurni Lorenz (2011). "Studying cartographic heritage: Analysis and visualization of geometric distortions". In: *Computers & Graphics* 35.2, pp. 402–411.

Kanatani Ken-ichi (1994). "Statistical bias of conic fitting and renormalization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.3, pp. 320–326.

Kessler Fritz (2006). "Working with projections and datum transformations in ArcGIS: theory and practical examples by Werner Flacke and Birgit Kraus". In: *Cartography and Geographic Information Science* 33.3, pp. 233–237.

Lagarias Jeffrey C, Reeds James A, Wright Margaret H, Wright Paul E (1998). "Convergence properties of the Nelder–Mead simplex method in low dimensions". In: *SIAM Journal on optimization* 9.1, pp. 112–147.

Marquardt Donald W (1963). "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the Society for Industrial & Applied Mathematics* 11.2, pp. 431–441.

National Geographic Society (2002). *Family Reference Atlas of the World*. National Geographic Family Reference Atlas of the World. National Geographic Society. ISBN: 9780792269304.

Nelder John A, Mead Roger (1965). "A simplex method for function minimization". In: *The computer journal* 7.4, pp. 308–313.

Semple, Kneebone (1998). *Algebraic projective geometry*. Oxford University Press.

Sturm Peter, Gargallo Pau (2007). "Conic fitting using the geometric distance". In: *Computer Vision–ACCV 2007*. Springer, pp. 784–795.

Wijewickrema Sudanthi, Esson Charles, Papliński Andrew (2006). *Orthogonal distance fitting revisited*. Tech. rep. Technical report 2006/205, Clayton School of Information Technology, Monash University, Melbourne, Australia.

– (2010). "Orthogonal Distance Least Squares Fitting: A Novel Approach". English. In: *Computer Vision, Imaging and Computer Graphics. Theory and Applications*. Vol. 68. Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 255–268. ISBN: 978-3-642-11839-5. DOI: 10.1007/978-3-642-11840-1_19.

Zhang Zhengyou (1997). "Parameter estimation techniques: A tutorial with application to conic fitting". In: *Image and vision Computing* 15.1, pp. 59–76.

Köszönetnyilvánítás

Köszönettel tartozom elsősorban a témavezetőmnek *Gede Mátyásnak* a témafelvetésért, a dolgozat alapját képző kutatás során adott iránymutatásaiért, javaslataiért és a dolgozat témájához kapcsolódó publikációs lehetőségért. Köszönettel tartozom továbbá *Györffy Jánosnak* a rendelkezésemre bocsátott szakirodalomért.

Nyilatkozat

Alulírott, **Barancsuk Ádám (Neptun kód: OCLJRV)** 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 témavezető által benyújtásra elfogadott diplomamunka PDF formátumban való elektronikus publikálásához a tanszéki holnapon **hozzájárulok**.

Budapest, 2015. június 8.

(a hallgató aláírása)