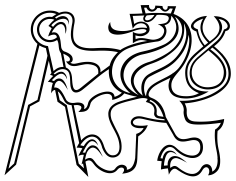
České vysoké učení technické v Praze
Fakulta stavební

Bakalářská práce

Praha 2015 Adam Laža

České vysoké učení technické v Praze
Fakulta stavební
Obor geodézie, kartografie a geoinformatika



Bakalářská práce

Interpolace metodou přirozeného souseda

Natural neighbour interpolation

Vedoucí práce: Ing. Martin Landa, Ph.D.
Katedra geomatiky

Praha 2015

Adam Laža

Abstract

Cílem této bakalářské práce je návrh a implementace interpolace metodou přirozeného souseda pro GRASS GIS 7. Starší verze 6 tohoto open-source softwaru sice tuto metodu nabízí v rámci volitelně doinstalovatelných balíčků Add-Ons, ale jako modul napsaný v bashi a s vnitřní závislostí na knihovně *nn-c*. Tato knihovna obsahuje knihovnu *Triangle*, která nedovoluje zařazení tohoto nástroje do oficiální distribuce GRASS GISu.

V rámci této bakalářské práce byl modul přepsán do jazyka Python, tak aby vyhovoval verzi 7. Následně byla vytvořena knihovna, která nevyužívá knihovnu *Triangle*, tak aby mohl tento interpolační nástroj zařazen do oficiální distribude GRASS GISu. Část textu této práce se dále zabývá porovnání rychlosti a kvality výstupu mezi GRASS GISem a ostatními gisovými softwary.

Kličová slova: GIS, GRASS GIS, interpolace, přirozený soused

Abstract

Abstract in english.

Keywords: GIS, GRASS GIS, natural neighbour interpolation

Obsah

Úvod

V reálném životě se většinou nesetkáváme s případem, kdy pro naši oblast zájmu, ať už se jedná o interval, plochu nebo prostor, máme dostatek bodových dat o daném jevu. Mnohem častěji máme k dispozici pouze soubor bodových dat, která jsou buď náhodně nebo uspořádaně rozmístěna po naší oblasti zájmu. Fyzické zhuštění takovéto sítě a sběr dalších dat může být časově či finančně náročné, příliš obtížné nebo v rámci možností metod zcela nereálné.

Obvykle ovšem potřebujeme znát hodnotu daného jevu i mimo měřené body, nejčastěji pro celé zájmové území. V takovéto chvíli se musíme použít nějaký interpolační nástroj, který vypočte přibližnou hodnotu v území mezi měřenými body. Jako příklad může být uvedeno vytvoření výškopisu, či DMT pro území, kde máme k dispozici data o výšce v pravidelné mřížce nebo teplotní mapa na základě údajů z nepravidelně rozmístěných meteostanic.

V této práci se budu zabývat interpolací přirozeného souseda a její implementací do GRASS GISu 7[1].

[1] `http://grass.osgeo.org`

1 GRASS GIS

GRASS (Geographic Resources Analysis Support System) GIS je geografický informační systém pro správu a analýzu prostorových dat, obrazových záznamů, produkci map a grafických výstupů, prostorové modelování a 3D vizualizaci. Na mnoha platformách (GNU/Linux, MS Windows, MAC OS) umožňuje práci s rastrovými i vektorovými daty a to buď pomocí příkazové řádky nebo grafického uživatelského rozhraní. GRASS GIS je otevřený a volně šiřitelný software pod licencí GNU GPL.

Historie[2] GRASS GISu začíná v roce 1982, kdy začal být vyvíjen U.S. Army Corps of Enginneer/CERL (Construction Engineering Research Lab) pro vojenské účely. Nicméně koncem osmdesátých let byly veškeré zdrojové kódy dány k dispozici veřejnosti. Na začátku devadesátých let se začal pomocí internetu celosvětově rozšiřovat. V roce 1995 CERL odstoupil od projektu a vývoje se ujal GRASS Development Team, který zahrnoval odborníky z celého světa.

GRASS je jeden z nejznámějších open-source GIS softwarů, jehož vývoj trvá déle než třicet let. Jádro softwaru je napsáno v jazyce C. Avšak snahou vývojářů je rozšíření GRASSu mezi širší odbornou veřejnost a proto v rámci snadnějšího použití jsou do programu začleněny moduly napsané v jazyce Python nebo C. Aktuálně je k dispozici verze 7, na jejímž vývoji se podílí několik vývojářů z řad dobrovolníků po celém světě.

[2]Praktická rukověť ke geografickému informačnímu systému GRASS `http://geo.fsv.cvut.cz/data/grasswikicz/grass_prirucka/grass_prirucka_0.4.pdf`

2 Postup řešení

2.1 Bash

Při řešení otázky, jak implementovat metodu přirozeného souseda pro GRASS 7, jsem vycházel z modulu napsaného pro GRASS GIS 6, který jsem měl k dispozici. Jednalo se o modul *v.surf.nnbathy* pro vektorová data. Tento modul byl napsán v Bashi. Pro novou verzi GRASS GISu 7, ve které si vývojáři kladou za cíl zpřístupnit tento software širší veřejnosti, tento modul v Bashi ovšem nebylo možno použít, neboť do nové verze se počítá pouze s moduly v jazyce Python a C.

2.1.1 v.surf.nnbathy

[3] *v.surf.nnbathy* je modul napsaný v bashi. Slouží jako interface mezi *nnbathy* z externí knihovny *nn-c* a GRASS GISem. *v.surf.nnbathy* nabízí celkem tři algoritmy interpolace. Defaultně je nastaven *Watsonův algoritmus pro Sibsonovu interpolaci*. Další možností je *Delaunayova triangulace* a poslední *Bělikovův a Semenovův algoritmus pro nesibsonovu interpolaci*. Pro Delaunayvou triangulaci, která je základem pro všechny tři algoritmy, se využívá knihovny *Triangle* napsanou Jonathanem Richardem Schewchukem. Parametry pro spuštění modulu jsou tyto (nepovinné v hranatých závorkách):

output Proměnná typu *string*, název výstupní rastrové mapy, jediný povinný parametr.

input Proměnná typu *string*, název vstupní vektorové mapy.

[file] Proměnná typu *string*, název vstupního souboru.

[zcolumn] Proměnná typu *string*, název sloupce z atributové tabulky, jehož data budou použity pro interpolaci.

[layer] Proměnná typu *integer*, nastavení, zda se jedná od 2D nebo 3D vektorová data.

[3] `http://svn.osgeo.org/grass/grass-addons/grass6/raster/v.surf.nnbathy/description.html`

[where] Proměnná typu *string*, SQL where podmínka.

[alg] Proměnná typu *string*, název použitého algoritmu.

Z výše uvedeného seznamu, lze vidět, že jako vstupní data je možné použít vektorovou mapu nebo textový soubor.

TODO Vlozit obrazek vstupni mapy a priklad vstupniho souboru

Vkladat kod do vlastni prace??, nastaveni listing

Volání v příkazové řádce pak může vypadata například takto:

Listing 1: bash version

```
user@my_comp:~$ v.surf.nnbathy input=elevation_lid792_randpts@PERMANENT output=
    raster_map zcolumn=value alg=nn
```

2.2 Python

Jako první krok pro implementaci interpolace přirozeného souseda pro GRASS GIS 7 bylo potřeba stávající modul v bashi přepsat do podporovaného programovacího jazyka. Pro verzi 7 bylo možné napsat moduly buďto v jazyce C nebo Python. Z důvodu nepříliš velké zkušenosti v programování byl pro začátek zvolen jazyk Python, který je pro méně zkušené programátory vice přívětivý.

2.2.1 v.surf.nnbathy.py

Z gitu stahnout kod bez objektu

V následující části této práce bude popsáno jak pythoní modul funguje, jaká jsou vstupní a výstupní data, jaké vytváří dočasné soubory.

Vstupní data Stejně jako původní bashový modul, i tento modul pracuje se vstupními daty buď v podobě textového ASCII souboru nebo vektorové mapy. Textový ASCII s n body musí obsahovat n řádků a 3 sloupce. V prvních dvou sloupcích je uložen údaj o poloze v podobě x a y souřadnice. Ve třetím sloupci jsou pak uloženy hodnoty veličiny, kterou chceme interpolovat.

```
638234.122902785427868 221198.4894384436775 62.817782000000001
638755.665974545176141 220976.783764891704777 9.190488000000000
638729.530741120455787 219988.669646041787928 91.799952000000005
638088.941303733270615 220228.186909802345326 76.839046999999994
638158.578729554312304 220794.514421981060877 2.037001000000000
637781.724264170858078 219988.193243810994318 8.298977000000001
638359.847223712014966 220692.375897407706361 15.550326000000000
639137.670258715632372 221096.622500944242347 16.613054999999999
637783.077698686625808 219603.568831721117022 38.046551000000001
639157.228799516917206 219692.376464909117203 82.820330999999996
```

Druhou možností vstupních dat je pak vektorová mapa s body. V tomto případě je pak při volání modulu použít parametr *zcolumn*, který určuje z jakého sloupce atributové tabulky se budou brát hodnoty k interpolaci.

Funkce region() Každá operace prováděná v GRASS GISu je prováděna pouze na určitém rozsahu území, tzv. *výpočetním regionu*. *Výpočetní region* je určen jako obdélník daný mezními kartografickými souřadnicemi a počtem řádků a sloupců.

Funkce *region()* všechna nastavení uloží do proměných. Dále vypočte plochu výpočetního regionu. Na rozdíl od GRASS GISu, který jako mezní kartografické souřadnice bere vnější rohy rohových buněk obdélníku, knihovna *nn-c* používá středy rohových buněk, a proto je třeba nastavení výpočetního regionu opravit o rozlišení buněk.

Listing 2: python version

```
def region():
    # set the region
    global area, ALG, nn_n, nn_s, nn_w, nn_e, null, ctype, cols, rows
    reg = grass.read_command("g.region", flags='p')
    kv = grass.parse_key_val(reg, sep=':')
    reg_N = float(kv['north'])
    reg_W = float(kv['west'])
    reg_S = float(kv['south'])
    reg_E = float(kv['east'])
    cols = int(kv['cols'])
    rows = int(kv['rows'])
    nsres = float(kv['nsres'])
    ewres = float(kv['ewres'])
    reg = (reg_N, reg_W, reg_S, reg_E)
    area = (reg_N-reg_S)*(reg_E-reg_W)
    ALG = options['algorithm']

    # set the working region for nnbathy (it's cell-center oriented)
    nn_n = reg_N - nsres/2
    nn_s = reg_S + nsres/2
    nn_w = reg_W + ewres/2
    nn_e = reg_E - ewres/2
    null = "NaN"
    ctype = "double"
```

Funkce initials_controls() Ve chvíli, kdy je nastavený výpočetní region,můžeme provést úvodní kontroly a přípravy před samotným výpočtem. Zejména zda plocha výpočetního regionu není nulová a je kde provádět interpolaci. Dále je třeba zajistit jednoznačné určení vstupních dat, tedy zda se bude pracovat s ASCII souborem nebo vektorovou mapou, a jejich kontrolu, popřípadě SQL podmínku. Také kontrolujeme zda z knihovny *nn-c* máme nainstalovaný program *nnbathy*, který interpolaci provádí. Také je třeba vytvořit dočasné pomocné soubory, které využijeme při práci s daty.

V případě, že pracujeme s vektorovou mapu, uložíme informace o bodových datech do dočasného proměnné *TMPcat* pomocí modulu *v.out.ascii*. Výstupem toho

modulu je ASCII soubor o n řádcích a čtyřech sloupcích. V prvních dvou sloupcích je uložená poloha bodu, ve třetím jeho id a ve čtvrtém hodnota k interpolaci.

```
638234.122902785427868 221198.4894384436775 1 62.817782000000001
638755.665974545176141 220976.783764891704777 2 9.190488000000000
638729.530741120455787 219988.669646041787928 3 91.799952000000005
638088.941303733270615 220228.186909802345326 4 76.839046999999994
638158.578729554312304 220794.5144219810608775 2.037001000000000
637781.724264170858078 219988.193243810994318 6 8.298977000000001
638359.847223712014966 220692.375897407706361 7 15.550326000000000
639137.670258715632372 221096.622500944242347 8 16.613054999999999
637783.077698686625808 219603.568831721117022 9 38.046551000000001
639157.228799516917206 219692.376464909117203 10 82.820330999999996
```

Jelikož id bodu k dalším výpočtům nepotřebujeme do dočasné proměnné *TMPYZ* si uložíme pouze informace o poloze a hodnotu k interpolaci. V případě, že nepracujeme s vektorovou mapou, ale ASCII souborem, tak tento soubor rovnou uložíme do proměnné *TMPXYZ*.

```
def initial_controls():
    # setup temporary files
    global TMP, TMPcat, XYZout, TMPXYZ
    TMPXYZ = 'tmpxyz.txt'
    TMPcat = 'TMPcat.txt'
    TMP = grass.tempfile()
    #TMPcat = grass.tempfile()
    #TMPXYZ = grass.tempfile()
    XYZout = grass.tempfile()

    if (TMPcat or TMPXYZ or XYZout or TMP) is None:
        grass.fatal("Unable to create temporary files.")

    # other controls
    if not grass.find_program('nnbathy'):
        grass.fatal('nnbathy is not available')

    if (options['input'] and options['file']):
        grass.message("Please specify either the 'input' or 'file' option, not
            both.")

    if not(options['input'] or options['file']):
        grass.message("Please specify either the 'input' or 'file' option.")

    if (options['file'] and os.path.isfile(options['file'])):
        grass.message("File "+options['file']+" does not exist.")

    if area == 0:
        grass.fatal(_("xy-locations are not supported"))
        grass.fatal(_("Need projected data with grids in meters"))
```

```
if not options['file']:
    if int(options['layer']) == 0:
        LAYER = ''
        COLUMN = ''
    else:
        LAYER = int(options['layer'])
        if options['zcolumn']:
            COLUMN = options['zcolumn']
        else:
            grass.message('Name of z column required for 2D vector maps.')

    if options['kwhere']:
        grass.run_command("v.out.ascii", flags='r', overwrite=1, input=
            options['input'], output=TMPcat, format="point", separator="
            space", precision=15, where=options['kwhere'], layer=LAYER,
            columns=COLUMN)
    else:
        grass.run_command("v.out.ascii", flags='r', overwrite=1, input=
            options['input'], output=TMPcat, format="point", separator="
            space", precision=15, layer=LAYER, columns=COLUMN)

    if int(options['layer']) > 0:
        fin = open(TMPcat, 'r')
        fout = open(TMPXYZ, 'w')
        try:
            for line in fin:
                parts = line.split(" ")
                fout.write(parts[0]+' '+parts[1]+' '+parts[3])
        except:
            grass.message("Invalid input!")
        fin.close()
        fout.close()
    else:
        grass.message("Z coordinates are used.")
else:
    TMPXYZ = options['file']
```

Funkce compute() V této části kódu je volán program *nnbathy* s následujícími vstupními paramatry:

-w proměnná typu double, omezuje extrapolaci přiřezením minimální váhy pro vrchol Delaunayovi sítě. V našem případě nastavena nula, což zamezuje extrapolaci.

-i proměnná typu string, název vstupního souboru o n řádcích, se třemi sloupci, x a y souřadnicí a hodnotou k interpolaci

-x dvojice x_min, x^max typu double, mezní hodnoty výstupní mřížky

-y dvojice x_min, x^max typu double, mezní hodnoty výstupní mřížky

-P proměnná typu string, použitá metoda interpolace

-n dvojice double x double, rozlišení výstupní mřížky

```
637725 221045 NaN
637735 221045 NaN
637745 221045 NaN
637755 221045 NaN
637765 221045 NaN
637775 221045 23.2274425578696
637785 221045 20.3234644594092
637795 221045 23.6841650075168
637805 221045 27.1014688330494
637815 221045 30.5774260647664
```

Výstupem z *nnbathy* je soubor *XYZout*. Obsahuje data o výstupní mřížce buňku po buňce ve třech sloupcích. V prvních dvou jsou x a y souřadnice, ve třetím vyinterpolovaná hodnota. V případě buňek mimo oblast, kde probíhala interpolace, je ve třetím sloupci uložena hodnota NaN.

```
def compute():
        grass.message('"nnbathy" is performing the interpolation now. This may take
             some time...')
        grass.verbose("Once it completes an 'All done.' message will be printed.")

        #nnbathy calling
        fsock = open(XYZout, 'w')
        #TODO zkontrolovat zarovnani
        grass.call(['nnbathy',
                        '-W', '%d' % 0,
                        '-i', '%s' % TMPXYZ,
                        '-x', '%d' % nn_w, '%d' % nn_e,
                        '-y', '%d' % nn_n, '%d' % nn_s,
                        '-P', '%s' % ALG,
                        '-n', '%dx%d' % (cols, rows)],
                        stdout=fsock)
        fsock.close()
```

Funkce convert() Výstupní textový soubor z *nnbathy* je třeba upravit, aby s ním bylo možné dále pracovat v GRASS GISu. Pro další práci slouží dočasný soubor *TMP*. Při vytváření na začátku tohoto souboru vznikne hlavička, která obsahuje data o hranicích, rozlišení, typu a hodnotě null.

```
north: 228495.0
south: 215005.0
east: 644995.0
west: 630005.0
rows: 1350
cols: 1500
type: double
null: NaN
```

Dále je potřeba vybrat vyinterpolované hodnoty jednotlivých buněk ze souboru *XYZout*, kde jsou uloženy ve třetím sloupci na samostatných řádcích, a vložit je do souboru *TMP* v pravidelné mřížce.

Funkce import_to_raster()

Výstupní data

2.3 OOP

V průběhu práce na přidání pythoního modulu *v.surf.nnbathy* jsem zjistil, že v GRASS GISu verze 6 je k dispozici také bash modul *r.surf.nnbathy*, který pracuje s rastrovými daty. Jelikož větší část kódu byla pro moduly v.surf.nnbathy a r.surf.nnbathy společná, rozhodl jsem se hlavní výpočetní část spojit. Na místo dvou procedurálních modulů byla objektově vytvořena knihovna *nnbathy.py* a dva moduly *v.surf.nnabthy.py* pro vektorová data a *r.surf.nnbathy.py* pro data rastrová, které knihovnu *nnbathy* volají.

2.3.1 v.surf.nnbathy

V objektově orientovaném modulu pro vektorová data, tak zůstaly úvodní část, která automaticky generuje GUI, úvodní vstupní kontroly a dále if podmínka, která vyhodnocovala, zda vstupují vektorová data v podobě vektorové mapy nebo ASCII souboru.

2.3.2 r.surf.nnbathy

Modul *r.surf.nnbathy* pracuje na podobném principu jako modul *v.surf.nnbathy*, jen pro rastrová data. Při volání je možnost použít méně parametrů.

output Proměnná typu *string*, název výstupní rastrové mapy, jediný povinný parametr.

input Proměnná typu *string*, název vstupní vektorové mapy.

[alg] Proměnná typu *string*, název použitého algoritmu.

Volání v příkazové řádce pak může vypadata například takto:

Listing 3: bash version

```
user@my_comp:~$ v.surf.nnbathy input=elevation_lid792_randpts@PERMANENT output=
    raster_map zcolumn=value alg=nn
```

I v tomto modulu zůstala část, která vytváří GUI. Protože ale modul pracuje s daty pouze v podobě rastrové mapy nebyly potřeba žádně vstupní kontroly ani if podmínka.

2.3.3 nnbathy

V knihovně nnbathy, které oba moduly volají tedy zůstala hlavní výpočetní část kódu.

3 Interpolace přirozeným sousedem

Prostudovat a probrat s T. Bayerem, po te opravit a doplnit!

3.1 Thiessenovy polygony

Thiessenovy polygony jsou sice samostatnou interpolační metodou, *metoda přirozeného souseda* (MPS) je však využívá jako základ pro výpočet vah a proto bude jejich výpočet zmíněn i zde.

Thiessenovy polygony známé taky jako *Voronoiovy diagramy* jsou definovány takto: Nechť V je množina n bodů v rovině. Rovinu rozdělíme na n oblastí R takových, že R_i obsahuje všechny body z E^2, pro něž je bod $p_i \in V$ nejbližší soused. Toto rozdělení roviny se nazývá Voronoiův diagram (VD) množiny bodů.

Nechť $\mathbf{A} = A_1, ..., A_n$ je množina n bodů. Voronoiův diagram A_i je:
$V(A_i) = X \in R^d : |X - A_i| \leq |X - A_j| \forall_j = 1, ..., n,$
kde $|X - A|$ vyjadřuje Eukleidovskou vzdálenost mezi body X, A v prostoru R^d

VD má následující vlastnosti. Všechny oblasti jsou konvexní a oblast R_i obsahuje jediný bod $p_i \in V$. Některé oblasti jsou neohraničené. Tyto obsahují body $p_i \in CH(V)$. Počet hran a uzlů VD je přímo úměrný počtu bodů v množině V. Pokud žádné 4 body neleží na kružnici, uzly mají stupeň 3. Uzel VD leží ve středu kružnice určené 3 body z V, které leží v přilehlých oblastech VD a neleží na přímce.
Algoritmus konstrukce Voronoiova diagramu

Množinu n bodů v rovině rozdělíme svislou přímkou na dvě stejně velké podmnožiny *V1* a *V2*. Při dosažení malého počtu bodů zkonstruujeme VD1 a VD2 rozdělených množin , jinak dělení rekurzivně opakujeme. Po vyřešení dílčích VD spojujeme dvojice VD následujícím způsobem:

Nechť VD1 a VD2 kspi dva vypočítáné Voronoiovy diagramy množin oddělených hraniční svislou přímkou. Pro jejich spojení využijeme skutečnosti, že hrany výsledného VD patří buď zcela do VD1 nebo VD2, nebo leží na osách úseček, spojujících body z V1 a V2 (tzv. bisektory). Tyto hrany tvoří řetěz, který je monotónní ve směru osy y.

Nalezneme konvexní obal sjednocených množin *V1* a *V2*. Nový konvexní obal obsahuje úsečky tečné k původním konvexním obalům. Sestrojíme bisektory těchto tečných úseček.

Nechť l je bisektor horní úsečky, určené vrcholykonvexního obalu. Přímka l leží současně v oblastech $R_i \subset VD1$ a $R_j \subset VD2$. Určíme místo, ve kterém l opustí buď oblast R_i nebo R_j a vstoupí do oblasti R_k. Oblast R_k patří buď k VD1, nebo k VD2 a obsahuje bod p_k. Na hranici ukončíme hranu l a vytvoříme novou hranu, která leží na bisektoru bodů p_k a zbývajícího bodu p_i nebo p_j (pokud opustíme oblast R_j nebo R_i).

Směrem dolů vytváříme další hrany VD a algoritmus ukončíme, pokud dosáhneme spodní polopřímkové hrany.

Delaunayova triangulace

Nechť V je množina n bodů. Předpokládáme, že žádné 4 body z V neleží na kružnici. Pokud propojíme všechny sousední body určené Voronoiovým diagramem úsečkami, dostaneme triangulační síť, kterou studoval Delaunay v roce 1935. Tato triangulace je výhodná např. pro interpolaci, neboť minimalizuje délky hran trojúhelníků a zpřesňuje tak výpočty přírůstků.

3.2 Samotná interpolace

MPS může být použita i pro bod x, který není součástí množiny A. V tomto případě přirození sousedé bodu A jsou body z množiny A,jejichž Voronoiovy diagramy by byly pozměneny v případě, že by bod A byl vložen do VD(S).

Průnik x vytvoří nový VD, který "ukrade"plochu VD, které by byly jeho přirozenými sousedy.