ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEODÉZIE, KARTOGRAFIE A GEOINFORMATIKA

BAKALÁŘSKÁ PRÁCE

TRANSFORMACE MAP I. VOJENSKÉHO MAPOVÁNÍ METODOU THIN PLATE SPLINE

Vedoucí práce: doc. Ing. Jiří Cajthaml, Ph.D.

Katedra geomatiky

červen 2018

Jan Myslivec

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Myslivec Jméno: Jan Osobní číslo: 458960

Zadávající katedra: Katedra geomatiky

Studijní program: B3646 Geodézie a kartografie

Studijní obor: 3646R011 Geodézie, kartografie a geoinformatika

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce: Transformace map I. vojenské mapování metodou Thin Plate Spline

Název bakalářské práce anglicky: Transformation of First Military Mapping Survey maps by TPS method

Pokyny pro vypracování:
Cílem práce je transformovat rastry I. vojenského mapování z oblasti Čech metodou Thin Plate Spline do souvislé mapy v systému JTSK. Jako programovací prostředí pro transformaci bude využit Matlab. Po vytvoření souvislé mapy bude následovat zhodnocení výsledku a porovnání s dosavadní mapou získanou polynomickou transformací.

Seznam doporučené literatury:
Bookstein, F.L.: Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. IEEE Trans. Pattern Anal. Mach. Intell. 11, 567-585
J. Cajthaml: Analýza starých map v digitálním prostředí například Müllerových map Čech a Moravy, 2012, ČVUT v Praze.

Jméno vedoucího bakalářské práce: Doc. Ing. Jiří Cajthaml, Ph.D.

Datum zadání bakalářské práce: 19. 2. 2018 Termín odevzdání bakalářské práce: 27. 5. 2018
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

[redacted]
Podpis vedoucího práce

[redacted]
Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v bakalářské práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce" a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací".

19. 2. 2018
Datum převzetí zadání

[redacted]
Podpis studenta(ky)

ABSTRAKT

Tato práce se zabývá transformací naskenovaných map I. vojenského mapování metodou „thin plate spline“ v programu MATLAB od firmy MathWorks. Výstupem by měla být sada georeferencovaných rastrů, které lze jednoduše otevřít v softwaru ArcMap od ESRI a vytvořit z nich mozaiku publikovatelnou na webovém rozhraní. Práce je pojata spíše z praktického hlediska.

KLÍČOVÁ SLOVA

thin plate spline, TPS, I. vojenské mapování, MATLAB, worldfile, afinní transformace, projektivní transformace, kolineární transformace, polynomická transformace, georeferencování rastrů, ArcMap

ABSTRACT

This work concentrates on transformation of scanned I. military survey maps, using ”thin plate spline“ method in MATLAB, a software by MathWorks company. The results of the work should be a set of georeferenced rasters, easy to open in ArcMap software by ESRI and to use them for creating a mosaic, published on web later. The work is considered rather practical.

KEYWORDS

thin plate spline, TPS, 1st military mapping survey, MATLAB, worldfile, affine transformation, homography, projective transformation, projective collineation, polynomial transformation, raster georeferencing, ArcMap

PROHLÁŠENÍ

Prohlašuji, že bakalářskou práci na téma „Transformace map I. vojenského mapování metodou thin plate spline“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne . . . .

(podpis autora)

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu své práce, doc. Ing Jiřímu Cajthamlovi, Ph.D., který mi během jejího zpracování ochotně poskytoval konzultace a věnoval mi drahocenný čas. Díky patří i mým rodičům a sourozencům, kteří mě během celého studia podporovali, ať už materiálně nebo duševně. V neposlední řadě děkuji za motivaci ze strany spolužáků a kamarádů.

Jan Myslivec

OBSAH

SEZNAM OBRÁZKŮ

1 ÚVOD

Vzhledem k tomu, že První vojenské mapování [3] v Čechách probíhalo mezi lety 1764 a 1783, kdy zde ještě nebyly zbudované polohové základy, a bylo při něm hojně využíváno nepříliš přesných měřických metod, jako jsou krokování a měření „od oka“, je jasné, že výsledek tomu odpovídá. Na druhou stranu však skýtají mapy I. vojenského mapování cenný historický obsah. O některých objektech na nich zakreslených lze tvrdit, že se za dvě stě padesát let víceméně nepohnuly (kostely, hrady, …). Ty lze využít jako identické body pro transformaci, čímž by se zpřesnil obsah map a tím pádem i zvýšil jejich informační potenciál. Jinak řečeno jedná se o to staré mapy zdeformovat tak, aby se poloha těchto objektů co nejméně lišila od jejich polohy na dnešních mapách.

Pokus o takové zdeformování byl již úspěšně učiněn pomocí afinní a projektivní či polynomické transformace. Všechny z nich ale využívají metody nejmenších čtverců, kdy mezi objekty po transformaci a jejich polohou na dnešních mapách zůstávají rezidua neboli zbytkové odchylky. Takovým metodám se říká reziduální.

Náplní mojí práce bylo využít transformaci nereziduální, kdy každý identický bod po transformaci „sedne“ na svůj vzor. Takovou transformací je mimo jiné i tzv. thin plate spline (dále jen „TPS“). TPS je výpočetně náročná, zvláště pro velký počet identických bodů, jak jsem se během práce sám přesvědčil.

Nejdříve je potřeba se hlouběji věnovat I. vojenskému mapování, předchozím reziduálním metodám a popsat princip TPS.

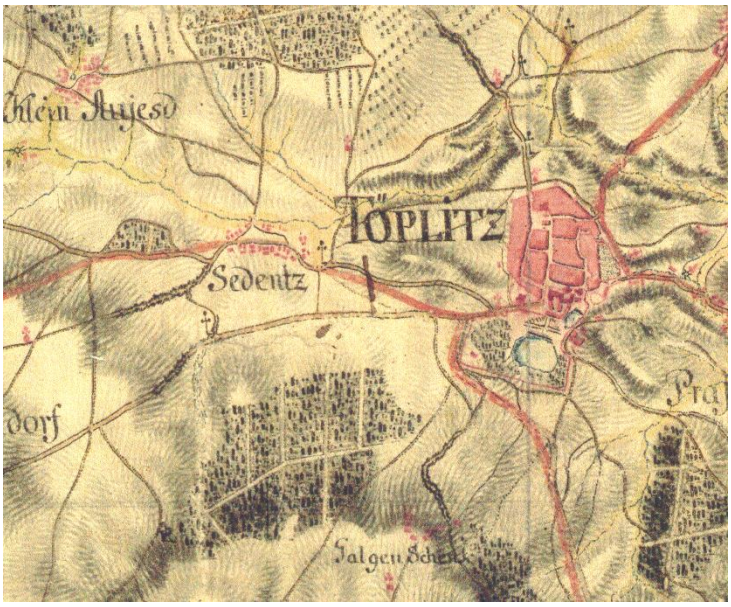
2 I. VOJENSKÉ MAPOVÁNÍ

V disertační práci doc. Ing Jiřího Cajthamla, Ph.D. (vedoucího mé práce), [3] je možné se dočíst, že I. vojenské mapování bylo zahájeno na základě potřeby kvalitních vojenských map, kterých byl v Habsburské monarchii nedostatek. Mapování v Čechách probíhalo v letech 1764–1767, později (1780–1783) byla severní část Čech zmapována znovu, opět kvůli nekvalitě.

Mapy tedy obsahují vojensky významné prvky, jako jsou cesty, budovy, mosty, louky, lesy, řeky, potoky, vodní plochy apod. Neobsahují výškopis, pouze nepřesné ztvárnění reliéfu pomocí nepravých sklonových šraf. Jsou obdélníkového tvaru, na jejich pravém okraji se nachází tabulka zakreslených obcí a na horním okraji název mapového listu a další náležitosti. (To všechno je potřeba k tvorbě bezešvé mozaiky oříznout).

Mají jednotné měřítko, a sice 1: 28 800 na požadavek Habsburské monarchie. Rozměry mapových listů jsou 618 x 408 mm a území Čech je zobrazeno na 273 takovýchto listech. K nahlédnutí je ukázka jednoho mapového listu (obr. 1) a jeho detail (obr. 2).

Obr. 1 - ukázka mapového listu I. vojenského mapování



Obr. 2 – detail mapového listu I. vojenského mapování

3 REZIDUÁLNÍ TRANSFORMAČNÍ METODY

Nyní je potřeba probrat matematický princip souřadnicových transformací v rovině. V této kapitole to budou transformace reziduální, kdy není možné vyhovět požadavku, aby po transformaci všechny identické body perfektně doléhaly, na rozdíl od transformací nereziduálních, kdy to možné je. Docent Cajthaml se ve své monografii [2] věnuje kormě dalších i těmto reziduálním způsobům:

3.1 AFINNÍ TRANSFORMACE

Afinní transformace je v podstatě druh polynomické transformace. Jedná se o polynomickou transformaci prvního stupně. Stejně tak i shodnostní a podobnostní transformace jsou zjednodušením afinní transformace. Nicméně afinní transformace je matematickým vyjádřením nejen otočení, změnou měřítka a posunu obrazu v souřadnicové soustavě, ale i jeho zkosením. Velice se tak hodí pro georeferencování rastrů postižených srážkou, kdy se naskenovaný papír smršťuje v jednom směru více než v jiném a může dojít ke zkosení či protažení. (Georeferencováním rozumějme případnou transformaci a umístění rastrů do souřadnicové soustavy). Dvě rovnoběžné přímky zůstanou i po transformaci nadále rovnoběžné.

Rovnice afinní transformace vypadá následujícím způsobem:

$$X = X_0 + a_{11}x + a_{12}y \tag{1}$$

$$Y = Y_0 + a_{21}x + a_{22}y \tag{2}$$

… a_{ij} jsou obecné koeficienty vyjadřující vliv měřítka, stočení a zkosení (měřítko osy *x*, měřítko osy *y*, rotace osy *x* a rotace osy *y*,)
… *x, y* jsou souřadnice bodu ve vstupní soustavě
… *X, Y* jsou souřadnice transformovaného bodu ve výstupní soustavě
… X_0, Y_0 je vyjádření posunu ve výstupní soustavě

3.2 PROJEKTIVNÍ TRANSFORMACE

Projektivní transformace (nebo také kolineární) se běžně využívá ve fotogrammetrii. Lze ji zapsat rovnicemi:

$$X = \frac{ax+by+c}{gx+hy+1} \tag{3}$$

$$Y = \frac{dx+ey+f}{gx+hy+1} \tag{4}$$

… *a–h* jsou koeficienty projektivní transformace

K výpočtu těchto osmi transformačních koeficientů je potřeba alespoň čtyř identických bodů. Při vyšším počtu by úloha mohla být řešena i vyrovnáním. Dvě rovnoběžné přímky po transformaci obecně nemusejí zůstat rovnoběžné. Je-li potřeba transformovat jeden obecný čtyřúhelník (např. rohy mapových listů) do tvaru a rozměru jiného čtyřúhelníku, použijeme právě tuto metodu.

3.3 POLYNOMICKÁ TRANSFORMACE

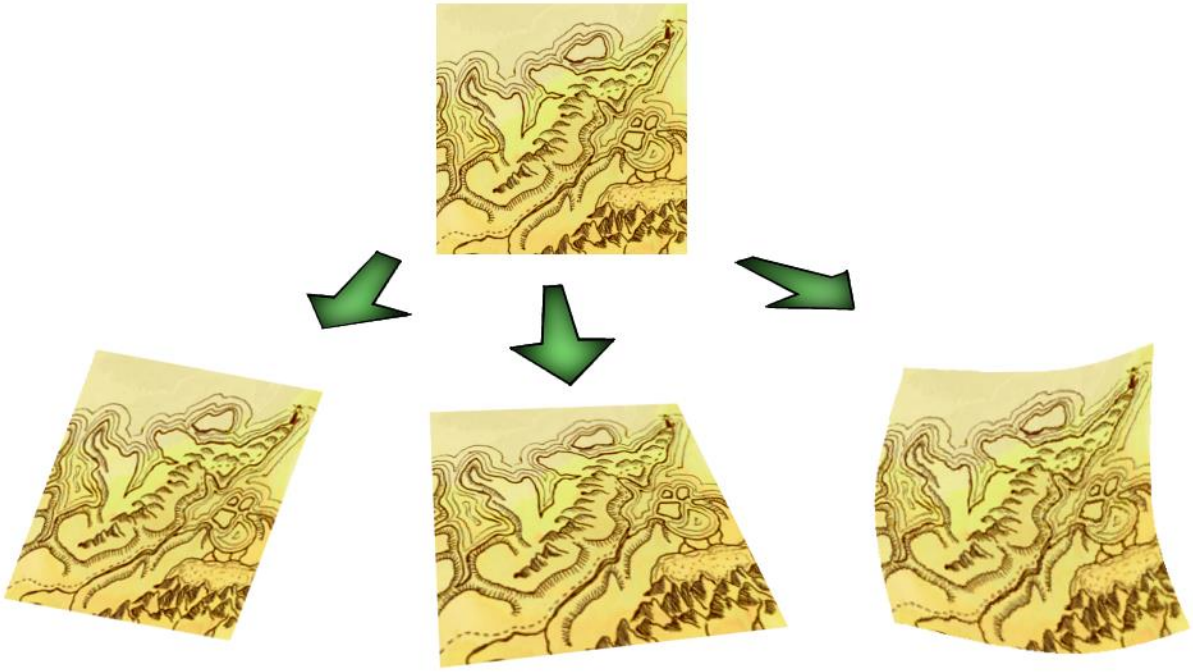
Polynomické transformace můžeme rozdělit podle stupně na transformaci 1. stupně (afinní), 2. stupně až *n*-tého stupně, přičemž se zpravidla používá stupně 2. a 3. Uvedeme si rovnice polynomické transformace 2. stupně:

$$X = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 \quad (5)$$

$$Y = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2 \quad (6)$$

... a_i, b_i jsou koeficienty polynomické transformace

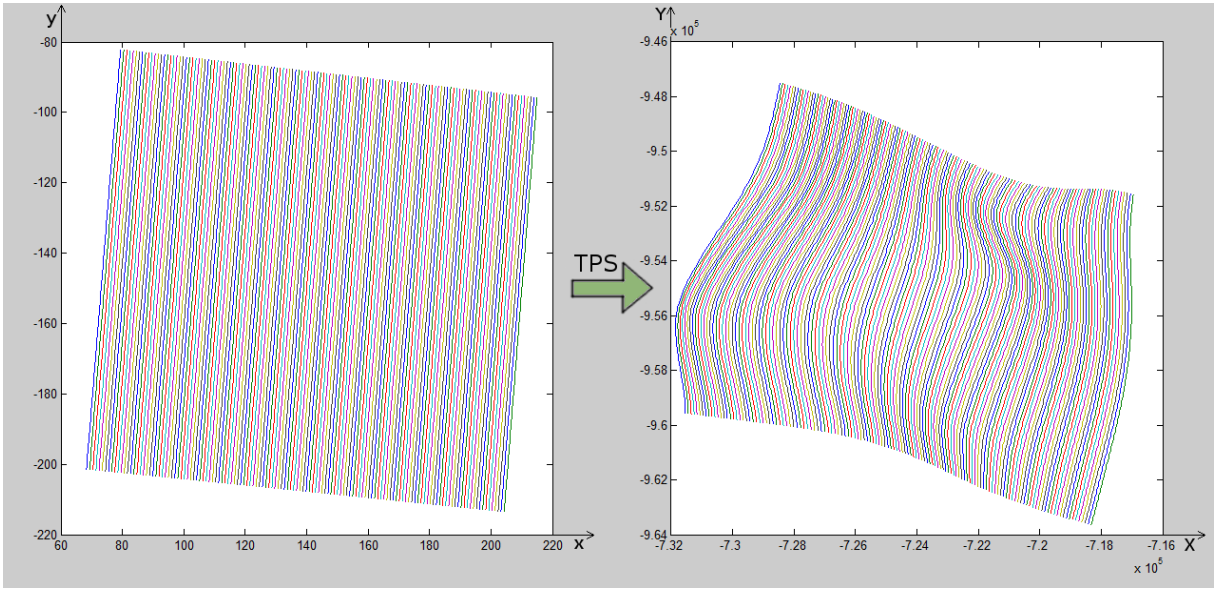
Na rozdíl od obou předchozích transformací, při polynomické transformaci se přímky zakřivují. Pro ilustraci všech tří transformací slouží obrázek 3.



Obr. 3 – afinní, projektivní a polynomická transformace (postupně, zleva doprava)

4 THIN PLATE SPLINE – NEREZIDUÁLNÍ TRANSFORMAČNÍ METODA

Pro splnění požadavku doléhajících identických bodů je možné využít transformaci metodou TPS. Kromě ní existují ještě další potenciálně použitelné metody, jako je metoda IDW (z angl. „inverse distance weighting“, česky „vážené inverzní vzdálenosti“) nebo krigování. Těmi se však zabývat nebudeme. Pro představu fungování TPS uvádím ilustrační obrázek sítě před a po transformaci (obr. 4).



Obr. 4 – ilustrace transformace metodou TPS

O TPS se lze dočíst v řadě odborných publikací. [1, 7, 5] Obecně ji lze zapsat jako afinní transformaci bodu j o souřadnicích x_j, y_j, obohacenou o vyrovnávací členy δ_{Xj} a δ_{Yj}. Souřadnice bodu j ve výstupní soustavě po transformaci jsou X_j, Y_j. Transformační rovnice zapíšeme takto:

$$X_j = a_{X0} + a_{X1}x_j + a_{X2}y_j + \delta_{Xj} \tag{7}$$

$$Y_j = a_{Y0} + a_{Y1}x_j + a_{Y2}y_j + \delta_{Yj} \tag{8}$$

Velikost vyrovnávacích členů závisí na vzdálenosti bodu j od každého identického bodu. Tato závislost není lineární, jak je vidět v rozepsaných transformačních rovnicích:

$$X_j = a_{X0} + a_{X1}x_j + a_{X2}y_j + 0.5\sum_{i=1}^{n} b_{Xi}\, r_{ij}^2 \log r_{ij}^2 \tag{9}$$

$$Y_j = a_{Y0} + a_{Y1}x_j + a_{Y2}y_j + 0.5\sum_{i=1}^{n} b_{Yi}\, r_{ij}^2 \log r_{ij}^2 \tag{10}$$

...kde malé n je počet identických bodů a i je číslo konkrétního identického bodu.

Proč má ale vyrovnávací člen právě takovýto tvar? Představme si, že původní obraz, který se pokoušíme transformovat, je tenký železný plát (z toho plyne „thin plate“ v názvu metody). Optimálně provedená transformace je taková, která minimalizuje pomyslné pnutí v tomto plátu. Pnutí je vyjádřeno následujícím integrálem:

$$\iint_{R^2} \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right) dxdy = min \tag{11}$$

... kde R^2 jsou meze ohraničující původní obraz a f je transformační funkce. Po dalším odvozování je možné za tuto funkci prohlásit obě z výše uvedených transformačních rovnic (vzorce (9) a (10)).

Předpokládejme tedy, že máme předpisy transformačních rovnic, obě sady souřadnic identických bodů (x_i, y_i i X_i, Y_i), ale hledáme koeficienty a_{X0}, a_{X1}, a_{X2}, a_{Y0}, a_{Y1}, a_{Y2}, b_{Xi} a b_{Yi}. Rozdělme v tomto bodě naše hledání na dvě části. Nejprve se zabývejme koeficienty s indexem X. Pro koeficienty s indexem Y platí totiž obdobné vztahy:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & x_1 & x_2 & \cdots & x_n \\ 0 & 0 & 0 & y_1 & y_2 & \cdots & y_n \\ 1 & x_1 & y_1 & 0 & r_{1,2}^2 \log r_{1,2}^2 & \cdots & r_{1,n}^2 \log r_{1,n}^2 \\ 1 & x_2 & y_2 & r_{1,2}^2 \log r_{1,2}^2 & 0 & \cdots & r_{2,n}^2 \log r_{2,n}^2 \\ 1 & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & y_n & r_{1,n}^2 \log r_{1,n}^2 & r_{2,n}^2 \log r_{2,n}^2 & \cdots & 0 \end{bmatrix} * \begin{bmatrix} a_{X0} \\ a_{X1} \\ a_{X2} \\ b_{X1}/2 \\ b_{X2}/2 \\ \vdots \\ b_{Xn}/2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ X_1 - x_1 \\ X_2 - x_2 \\ \vdots \\ X_n - x_n \end{bmatrix} \tag{12}$$

Označme každou matici jedním písmenem:

$$L * w_X = z_X \tag{13}$$

Potom platí:

$$w_X = L^{-1} * z_X \tag{14}$$

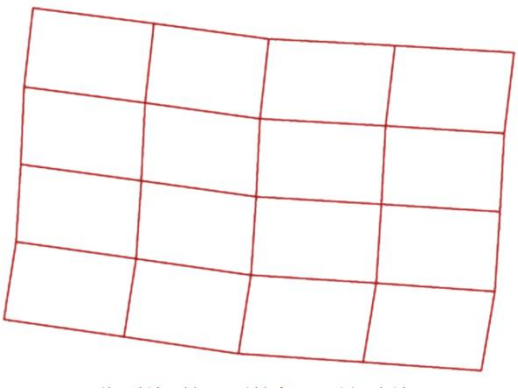
Hledané koeficienty transformační rovnice pro souřadnici X jsou obsaženy ve vektoru w_x. Při zkušebním opětovném transformování identických bodů za použití vypočtených koeficientů bychom měli dosáhnout těch samých souřadnic X_j, Y_j, čili neměla by vznikat rezidua.

5 PŘEDCHOZÍ PRÁCE VĚNUJÍCÍ SE GEOREFERENCOVÁNÍ I. VOJENSKÉHO MAPOVÁNÍ

Hovoříme-li o georeferencování mapy, máme na mysli určení jejího umístění v geodetickém souřadnicovém systému. V Geografickém a kartografickém obzoru z roku 2013 [4] se můžeme dočíst o práci doc. Ing. Jiřího Cajthamla, Ph.D., který testoval metodu afinní transformace na mapových listech I. vojenského mapování pro oblast ústeckého kraje. Afinní transformace, jak už bylo řečeno, vhodně eliminuje srážku papíru. Jako výstupní systém volil S-JTSK. Transformaci provedl dvěma způsoby:

5.1 PRVNÍ ZPŮSOB: HROMADNÉ VYROVNÁNÍ

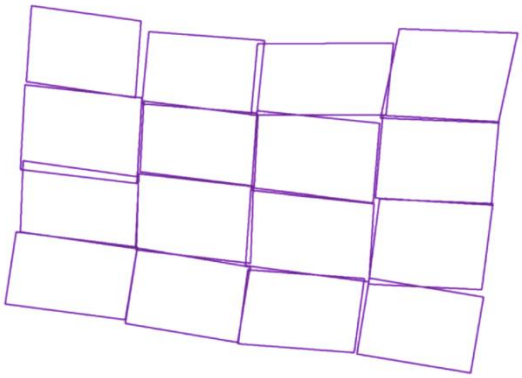
Jednalo se o hromadné vyrovnání zprostředkujících veličin s podmínkami. Podmínky vyjadřovaly požadavek návaznosti rohů mapových listů, tedy vytvoření souvislé, bezešvé mapy. Výsledkem samotného hromadného vyrovnání byly textové soubory obsahující pro každý mapový list šest transformačních parametrů definujících afinní transformaci. Těmito parametry pak byly listy transformovány. Výpočetní software vznikl jako výsledek diplomové práce. Práci dobře ilustrují obrázky 5, 6 a 7, které jsem převzal z Geografického a kartografického obzoru.



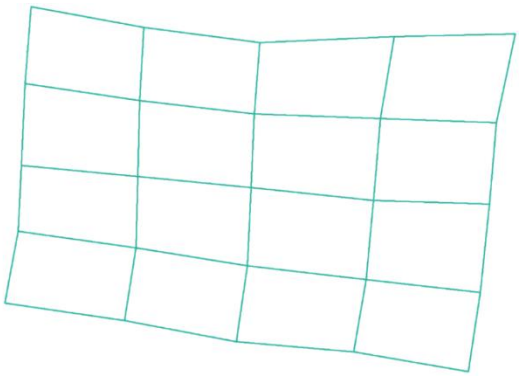
Obr. 5 – ilustrace kladu transformovaných listů (hromadné vyrovnání)

5.2 DRUHÝ ZPŮSOB: PROJEKTIVNÍ + AFINNÍ TRANSFORMACE

Docent Cajthaml nejdříve vyrovnal afinní transformací každý mapový list podle souřadnic identických bodů, aniž by řešil návaznost rohů. Vznikla tak mozaika nedoléhajících či překrývajících se listů (obr. 5). V dalším kroku zprůměroval vyrovnané souřadnice rohů odpovídajících si čtveřic mapových listů. (Na okrajích území šlo o dvojice, popř. samostatné rohy.) V posledním kroku „roztáhl“ jednotlivé mapové listy projektivní transformací, tak aby se jejich rohy dotýkaly právě v těchto zprůměrovaných místech (obr. 6).



Obr. 6 – ilustrace kladu nedoléhajících a překrývajících se listů (pouze afinní transformace)



Obr. 7 – ilustrace kladu transformovaných listů (projektivní + afinní transformace)

5.3 POROVNÁNÍ

Následně porovnal oba způsoby. U prvního způsobu mu posloužila střední souřadnicová chyba na identických bodech. U druhého způsobu to byl průměr těchto středních souřadnicových chyb, spočtených pro jednotlivé rastry, nikoli pro celé území.

Výsledkem byly dvě hodnoty: střední chyba 552 metrů pro první způsob a průměr středních chyb 237 metrů pro druhý způsob, jež je méně než poloviční. Závěrem uvedl, že pro práci na území jednoho mapového listu je výhodnější použít druhý způsob.

5.4 APLIKACE MULTIGEOREF

Na výzkum docenta Cajthamla navázala Ing. Tereza Fiedlerová ve své diplomové práci. [6] V programovacím jazyce C++ vytvořila aplikaci MultiGeoref sloužící ke georeferencování vícelistých mapových děl. Aplikace transformuje rastry prvním způsobem, hromadným vyrovnáním metodou nejmenších čtverců. Umožňuje výběr mezi afinní a polynomickou transformací 2. stupně, navíc umožňuje i zvolit buď klasický výpočet metodou nejmenších čtverců, nebo výpočet, který odhalí odlehlé id. body (body, na kterých vznikají největší odchylky) a přiřadí jim menší váhu, se kterou vstupují do výpočtu. Tato iterativní metoda se nazývá IRLS (z angl. „Iterative Reweighted Least Squares").

Pomocí programu MultiGeoref byly transformovány listy I. vojenského mapování pro území Čech (celkem 250 listů, ty samé, co používám já v této práci), použitá transformace – polynomická, metoda – IRLS. Výsledná střední souřadnicová chyba na id. bodech byla 280 m, zaokrouhleno na celé metry.

6 WORLDFILE

Chceme-li načíst rastrovou mapu do programu ArcMap, musíme znát, kde je v souřadnicovém systému umístěna. K tomu slouží soubor *worldfile*, jehož podrobný popis je možné si přečíst v internetové dokumentaci k programu ArcMap [9]. V něm je pod sebe zapsáno šest parametrů. Načítaný rastr totiž může být:

- škálovaný v každém směru jinak (vyjádřeno pomocí *mx* a *my* [v jednotkách souř. systému])

- stočený (o úhel *t* [v libovolných jednotkách])

- zkosený (vyjádřeno pomocí koeficientu k [bez jednotky])

- posunutý v obou osách [v jednotkách souř. systému]

Worldfile bude potom obsahovat šest pod sebou jdoucích parametrů:

1) $mx \cdot \cos t$ (15)
2) $my \cdot (k \cdot \cos t - \sin t)$ (16)
3) $mx \cdot \sin t$ (17)
4) $-my \cdot (k \cdot \sin t + \cos t)$ (18)
5) posun v ose x, vyjádřený souřadnicí x levého horního rohu
6) posun v ose y, vyjádřený souřadnicí y levého horního rohu

V případě, že při umisťování rastru není potřeba jej ani zkosit, ani natočit, bude schéma následující:

1) šířka pixelu
2) 0
3) 0
4) záporná výška pixelu
5) souřadnice x levého horního rohu
6) souřadnice y levého horního rohu

Příklad worldfilu:

```
 94.3826247
 0
 0
-94.3826247
-712588.816
-951943.153
```

Aby mohl být rastr správně načten, musí být wordlfile ve stejné složce se svým rastrem a musí mít shodný název a příslušnou příponu. Například pro rastr formátu *tiff* je to *tfw*, pro *png* je to *pgw* atd.

7 VSTUPNÍ DATA

Teď se dostáváme k praktické části práce. Mým úkolem bylo v Matlabu (verze R2014a) provést transformaci celkem 250 naskenovaných mapových listů prvního vojenského mapování a zachovat zhruba velikost původních rastrů, aby zbytečně nedocházelo ke ztrátě informací. Listů pro Čechy je celkem 273, jak jsme se dočetli v kapitole věnované I. vojenskému mapování, na některých je však vykreslen tak malý kousek území, že bylo rozhodnuto je ani netransformovat. Tudíž vstupními daty pro mě byla tato:

- 250 rastrových souborů ve formátu *tiff* s indexovanou paletou, rozměry v rozpětí cca 7 500 x 11 500 – 10 000 x 12 500 pixelů. Číslovány od 1001 do 1273 (např. „1001.tiff“).

- Textový soubor nazvaný „rozlozeni_bezkrajnich.txt“ obsahující schéma pravidelného rozložení mapových listů. Ukázka (bohužel se nevejde celá):

```
   0     0     0     0     0     0     0     0  1001  1002     0     0     0     0     0   ...
   0     0     0     0     0     0     0     0  1003  1004     0  1006  1007     0     0   ...
   0     0     0     0     0     0     0     0  1009  1010  1011  1012     0     0     0   ...
   0     0     0     0     0     0  1015  1016  1017  1018  1019  1020  1021  1022     0   ...
   0     0     0     0  1023  1024  1025  1026  1027  1028  1029  1030  1031  1032  1033   ...
   0     0     0  1035  1036  1037  1038  1039  1040  1041  1042  1043  1044  1045  1046   ...
   0  1049  1050  1051  1052  1053  1054  1055  1056  1057  1058  1059  1060  1061  1062   ...
1065  1066  1067  1068  1069  1070  1071  1072  1073  1074  1075  1076  1077  1078  1079   ...
1082  1083  1084  1085  1086  1087  1088  1089  1090  1091  1092  1093  1094  1095  1096   ...
1099  1100  1101  1102  1103  1104  1105  1106  1107  1108  1109  1110  1111  1112  1113   ...
   0  1117  1118  1119  1120  1121  1122  1123  1124  1125  1126  1127  1128  1129  1130   ...
   0  1135  1136  1137  1138  1139  1140  1141  1142  1143  1144  1145  1146  1147  1148   ...
   0  1153  1154  1155  1156  1157  1158  1159  1160  1161  1162  1163  1164  1165  1166   ...
   0  1170  1171  1172  1173  1174  1175  1176  1177  1178  1179  1180  1181  1182  1183   ...
   0     0  1188  1189  1190  1191  1192  1193  1194  1195  1196  1197  1198  1199  1200   ...
   0     0  1204  1205  1206  1207  1208  1209  1210  1211  1212  1213  1214  1215     0   ...
   0     0     0  1218  1219  1220  1221  1222  1223  1224  1225  1226     0     0     0   ...
   0     0     0  1228  1229  1230  1231  1232  1233  1234  1235  1236     0     0     0   ...
   0     0     0     0  1238  1239  1240  1241  1242  1243  1244  1245     0     0     0   ...
   0     0     0     0     0  1247  1248  1249  1250  1251  1252  1253     0     0     0   ...
   0     0     0     0     0  1254  1255  1256  1257  1258  1259     0     0     0     0   ...
   0     0     0     0     0     0  1262  1263  1264  1265     0     0     0     0     0   ...
   0     0     0     0     0     0     0  1268  1269  1270     0     0     0     0     0   ...
   0     0     0     0     0     0     0     0  1272     0     0     0     0     0     0   ...
```

- Ke každému rastru textový soubor „XXXX_ib.txt“ („XXXX“ značí název konkrétního rastru) obsahující souřadnice x_i a y_i identických bodů odečtených v rastru (v palcích), k nim odpovídající souřadnice X_i a Y_i v systému S-JTSK (v metrech). Kvalita skenování byla 400 dpi, vynásobíme-li tedy rastrové souřadnice hodnotou kvality v dpi, získáme rastrové souřadnice v pixelech. S těmi lze dále pracovat. Veškeré operace se souřadnicemi z textových souborů jsou tedy s takto přenásobenými hodnotami. Ukázka souboru:

```
 6.655769      9.380445     -729629.622355       -937230.873104
 3.574689      7.854949     -731631.561454       -937744.443306
 8.818595      9.238849     -727771.185305       -937460.531896
 6.404063      5.394535     -729937.133326       -939861.014987
 4.816803      5.997131     -730921.947535       -939344.712652
 6.265019     14.009842     -729552.581536       -934028.708590
 ...
```

- Ke každému rastru textový soubor „XXXX_rohy.txt“ obsahující rastrové souřadnice rohů mapového listu (v palcích). Každý mapový list byl totiž naskenován lehce nakřivo s bílou plochou kolem, navíc papír ani netvoří pravidelný obdélník. Ukázka:

```
 0.086468      20.498108
24.523389      20.820803
24.716512       4.486386
 0.410215       4.320780
```

8 POSTUP PRÁCE

8.1 KONVERZE RASTRŮ

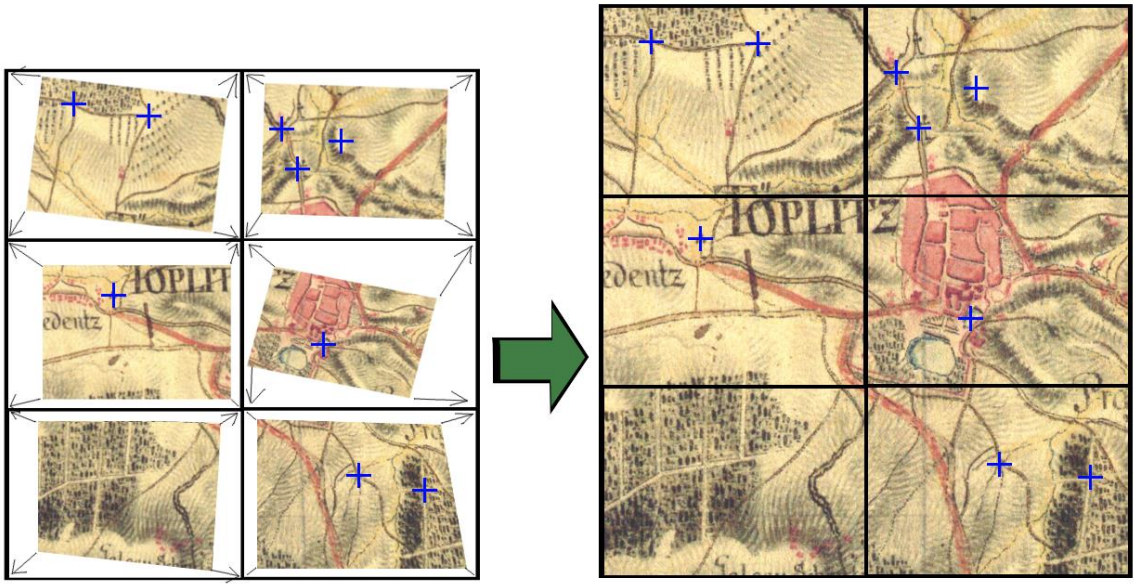
Nejdříve bylo potřeba převést rastry z indexované palety do režimu RGB. Tento krok je nutný pro pozdější interpolaci jednotlivých pixelů, kdy bude potřeba znát skutečné hodnoty barev každého pixelu, nikoli jen číslo jeho barvy v paletě. Provedl jsem tedy konverzi z formátu *tiff* s indexovanou paletou do formátu *jpg* v režimu RGB, a to v programu IrfanView vytvořeném Irfanem Skiljanem.

8.2 VÝPOČET MĚŘÍTKA VÝSTUPU

Následně bylo potřeba určit měřítko výsledných transformovaných rastrů. Skutečnou vzdálenost kterýchkoli dvou transformovaných id. bodů v systému JTSK v metrech známe. Ale neznáme jejich vzdálenost ve výsledném rastru v pixelech. Tyto vzdálenosti si obecně neodpovídají, nemají jednotné převodní měřítko. Je zbytečné, aby byl transformovaný rastr příliš velký, zároveň ale nechceme ztrácet cenné pixely. Proto jsem volil následující postup. Načetl jsem všechny soubory se souřadnicemi id. bodů do zadaného výpočetního prostředí, do Matlabu. Spočetl jsem vzdálenosti po sobě jdoucích id. bodů v rastru (získal jsem vektor r) i v S-JTSK (vektor R). Okleštil jsem r o příliš malé hodnoty, zatížené relativně největší chybou z nepřesnosti souřadnic. Pak jsem vypočetl vektor měřítek $scale = r/R$ a jako výsledné použil to největší. V podstatě jsem zhruba našel místo s největší ztrátou pixelů a přizpůsobil mu měřítko, aby se zde po transformaci žádné pixely neztratily a v ostatních místech jen přibývaly.

8.3 SÍŤ – VÝPOČET PARAMETRŮ PROJEKTIVNÍ TRANSFORMACE

Dále bylo potřeba seřadit rastry podle jejich schématu ze souboru „rozlozeni_bezkrajnich.txt“. Za předpokladu, že jednotlivé mapové listy na sebe svými rohy navazují, postupoval jsem tak, jak naznačuje obrázek 8. Vytvořil jsem pomyslnou pravidelnou síť s rozměry buňky zhruba odpovídajícími rozměrům vstupních rastrů a vypočetl parametry projektivní transformace tak, aby se rohy listů stýkaly v rozích buněk. Parametry jsem uložil do proměnné p. Malé p je datového typy *struct* a obsahuje prvky a až h a posun v osách x a y, obojí podle polohy listu ve schématu. Pro každý rastr tedy jiné parametry. Těmito parametry jsem pak transformoval id. body všech rastrů.

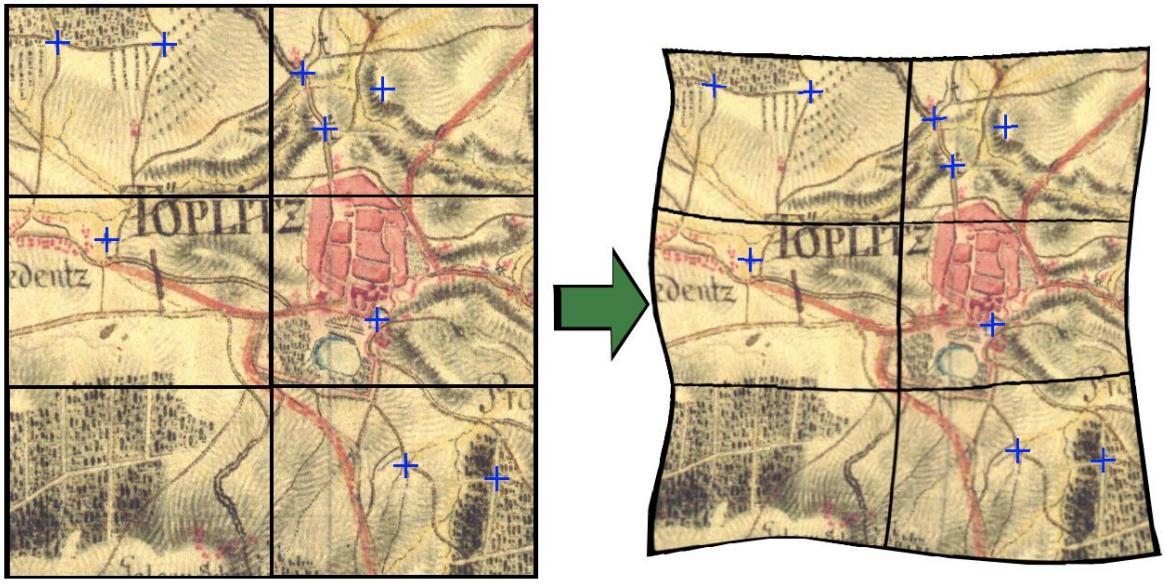


Obr. 8 – ilustrace projektivní transformace (id. body modře)

Rozměry buněk nejsou pro výpočet důležité. Nezáleží ani na tom, zda je síť pravidelná. Záleží jen na tom, zda se rohy listů dotýkají.

8.4 VÝPOČET PARAMETRŮ TPS

Poté jsem spočetl parametry TPS. Použil jsem pro to v Matlabu integrovanou funkci *tpaps*. Jejími vstupními argumenty jsou projektivně transformované souřadnice x_i a y_i, výsledné souřadnice X_i nebo Y_i (funkci je nutné použít dvakrát pro obě dvě) a vyhlazovací koeficient p, který musí být v našem případě nejvyšší možný, tedy roven jedné. Při nulové hodnotě by totiž koeficienty b_{Xi} vyrovnávacího členu byly nulové a jednalo by se o pouhou afinní transformaci. Vypočtené parametry TPS jsem uložil do proměnných pX a pY (opět typu *struct*). Jimi jsem už jen kontrolně transformoval id. body (obr. 9). (K transformaci bodů bylo potřeba použít funkci *fnval* a jako její argumenty zadat proměnnou pX nebo pY a vstupní souřadnice id. bodů.)



Obr. 9 – ilustrace transformace id. bodů (modře) metodou TPS

8.5 POZNÁMKA K ORIENTACI OS

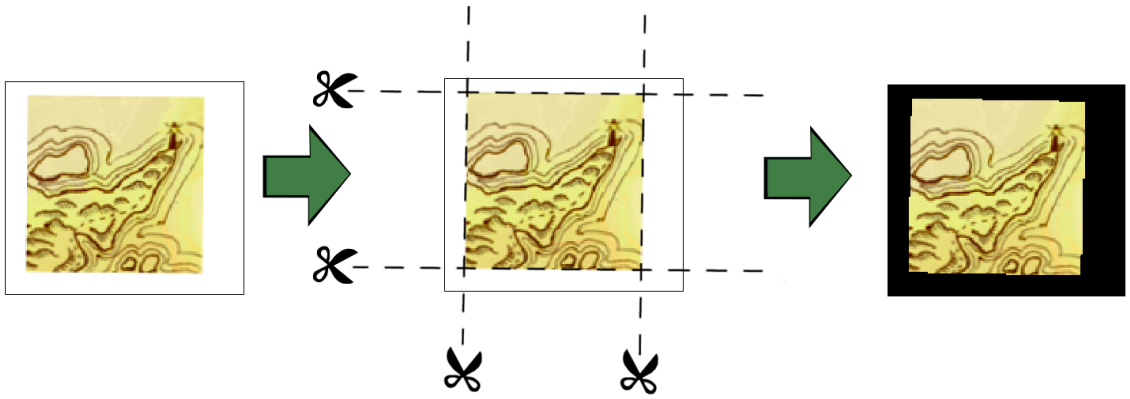
Ještě malá poznámka: vzhledem k tomu, že při odečítání souřadnic rohů a id. bodů z rastrů byla osa y určena směřující nahoru, zatímco v Matlabu se prvky matic indexují směrem dolů, musel jsem při načítání souřadnic rohů a id. bodů načítat i velikosti příslušných rastrů a tyto souřadnice převracet. Každou ypsilonovou souřadnici jsem odečetl od výšky rastru, a tak zařídil správnou orientaci os.

8.6 CYKLUS PRO JEDNOTLIVÉ RASTRY

Ve zbytku postupu popíši cyklus, který byl pro každý vstupní rastr stejný:

8.6.1 Ořezání vstupního rastru

Načetl jsem rastr a souřadnice jeho rohů. Podle souřadnic rohů jsem definoval 4 řezné přímky ohraničující prostor mapového listu uvnitř celého rastru. V iteračním for-cyklu jsem pro každý pixel určil, zda se nalézá uvnitř nebo vně tohoto prostoru, a podle toho jsem ho buď zachoval, nebo ho přepsal hodnotou {0 0 0} v režimu RGB (absolutní černě). Tato hodnota bude později v programu ArcMap představovat stoprocentní průhlednost. Tak jsem se zbavil bílého pozadí mapového listu, které by ve výsledné mozaice jen překáželo (obr. 10).



Obr. 10 – ilustrace ořezání rastru

Abych toto mohl udělat, musel jsem předtím ošetřit, že se absolutní černá nenachází také uvnitř listu. Tato místa jsem opět ve for-cyklu vyhledal a přepsal na hodnotu {1 1 1}, čímž jsem je nepatrně zesvětlil.

8.6.2 Aplikace projektivní transformace a TPS

Vytvořil jsem dvě matice představující souřadnice jednotlivých pixelu v původním rastru. Jejich rozměry odpovídaly rozměrům rastru.

Matice *x* se souřadnicemi x vypadala takto:

$$x = \begin{bmatrix} 1 & 2 & 3 & \cdots \\ 1 & 2 & 3 & \cdots \\ 1 & 2 & 3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

... a matice *y* se souřadnicemi y takto:

$$y = \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 2 & 2 & 2 & \cdots \\ 3 & 3 & 3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Všechny prvky těchto matic jsem transformoval projektivními transformačními rovnicemi za dosazení parametrů uložených v proměnné *p*, vypočtených pro daný rastr. Tím jsem získal dvě matice *xp* a *yp*. Ty jsem transformoval pomocí TPS za použití parametrů z proměnných *pY* a *pX* a dostal matice *Yw* a *Xw*. Tyto matice představují souřadnice místa v S-JTSK, které každý pixel v původním rastru zobrazuje. Za zmínku stojí, že kvůli nestandardní, nematematické orientaci souřadnicových os systému S-JTSK musely být souřadnice rohů listů i id. bodů už ve vstupních souborech prohozené a ještě s opačným znaménkem.

8.6.3 Tvorba a interpolace výstupního rastru

Poté jsem využil dvou funkcí napsaných panem Fitzgeraldem J. Archibaldem, jejichž kódy jsou dostupné na webových stránkách společnosti MathWorks. [8] Šlo o funkci *interp2_edited*, kterou jsem musel lehce poupravit, a o funkci *nearestInterp*, nástroj interpolace metodou nejbližšího souseda. K dispozici byla kromě *nearestInterp* i funkce *idwMvInterp*, aplikující interpolaci metodou IDW, která ovšem nedávala tak pěkné výsledky.

Argumenty funkce *interp2_edited* byly dvě matice souřadnic původních pixelů, *x* a *y*, a dvě matice souřadnic transformovaných pixelů, *Yw* a *Xw*. Dále proměnná typu *struct* určující metodu interpolace nejbližšího souseda, a měřítko výsledných rastrů, *scale*. Funkce vynásobila matice *Yw* a *Xw* hodnotou měřítka *scale*, odečetla od nich jejich nejmenší hodnotu (dostala je do kladných hodnot) a zaokrouhlila. Tím určila, který pixel ve výsledném rastru odpovídá kterému pixelu v rastru původním. Pak vytvořila třívrstvou matici *imgw* o šířce a výšce výstupního rastu (pro každý kanál režimu tříbarevného RGB jednu vrstvu) a příslušné pixely obarvila barvami původního rastru. Úplně na konec vyinterpolovala hodnoty zbylých neobarvených pixelů, a to metodou nejbližšího souseda. Výstupem funkce byla matice transformovaného rastru *imgw*.

8.6.4 Ukotvení výstupního rastru

Už před použitím funkce *interp2_edited* bylo potřeba přemýšlet nad ukotvením výsledného rastru pomocí worldfilu. Definoval jsem tedy každému rastru jeden bod, tzv. kotvu, určenou dvěma skalární mi proměnnými *anchor_Y* a *anchor_X*. Tou byl v pořadí první id. bod, který se na rastru nacházel. V základu stačilo určit, kde se kotva v rastru nachází (v pixelech), vyjádřeno souřadnicemi počítanými od počátku soustavy – levého horního rohu.

Přidal jsem tedy kotvu jako další argument funkce *interp2_edited* a v tělu funkce s ní provedl stejné operace jako se vstupními maticemi *Yw* a *Xw*. Takto upravenou kotvu jsem vydělil měřítkem *scale*, odečetl od neupravené a získal souřadnice levého horního rohu. Vytvořil jsem vektor *worldfile* určující obsah budoucího worldfilu a zapsal do něj postupně všech šest hodnot: převrácenou hodnotu měřítka *scale*, nulu, nulu, zápornou převrácenou hodnotu měřítka *scale*, souřadnici Y a souřadnici X daného rohu, obě v metrech. Funkce vrátila spolu s maticí transformovaného rastru i vektor těchto šesti hodnot.

8.6.5 Export

Posledním krokem bylo už jen zapsat matici *imgw* a vektor *worldfile* do souboru. Zvolil jsem formát rastru *png* a příslušný formát *pgw* pro worldfile. Zkušebně jsem nahrál data do programu ArcMap a zkontroloval, zda na sebe jednotlivé listy navazují, jestli mají identické body v rastru správné souřadnice, kvalitu rastrů, souřadnice levého horního rohu a další. V prostředí ArcMapu jsem nastavil hodnotu pixelu {0 0 0} jako stoprocentní průhlednou.

9 POTÍŽE PŘI PRÁCI

Nyní už by mohl následovat závěr, kdyby celý můj výpočetní skript nefungoval pouze na zkušebních datech. Abych urychlil proces tvorby a ladění skriptu, vytvořil jsem si zkušební data, konkrétně 7 rastrů o velikostech 120x135 pixelů, tedy výrazně méně a výrazně menší. K nim jsem dotvořil patřičné textové soubory. Nicméně po nasazení skriptu na „ostrá data" jsem musel čelit hned několika problémům způsobeným rozsahem těchto dat. Vyřešil-li jsem jeden, objevil se další.

9.1 NEDOSTATEK OPERAČNÍ PAMĚTI

Při aplikaci projektivní transformace a TPS na matice *x* a *y* došlo k selhání zapříčiněnému nedostatkem operační paměti doprovázeným chybovou hláškou „Out of memory. Type HELP MEMORY for your options." Můj počítač postrádal potřebné množství operační paměti. Byl jsem nucen rozdělit matice *x* a *y* na dvě části *A* a *B* svisle a exportovat obě poloviny obrázku zvlášť. To však nestačilo.

Spustil jsem výpočet, který nedoběhl. Po několika hodinách jsem ho přerušil a podíval se na aktuální stav. Zjistil jsem, že jen aplikace TPS na každý pixel (prováděl jsem ji postupně, v iteraci pro jednotlivé řádky) by trvala 48 hodin. Pro jeden rastr. S mým vedoucím práce jsme se po zralé úvaze rozhodli přenechat finální výpočet výkonnějším počítačům a jako demonstraci funkčnosti provést transformaci alespoň čtyř rastrů, s využitím pouze jejich id. bodů. Tím by se náročnost TPS snížila, protože by se snížil i počet koeficientů b_{Yi} a b_{Xi} pro každý id. bod. Ani tentokrát se záměr nezdařil.

Aplikování TPS se sice zrychlilo, ovšem při interpolaci rastru funkcí od F. J. Archibalda se ukázalo, že rozdělení na *A* a *B* je nedostatečné. Opět z důvodu chybějící operační paměti. Rozdělil jsem každou část ještě na čtvrtiny, to znamená celý list na osm částí, *a* až *f*. Jenže ani to nepomohlo. Učinil jsem proto následující.

Všechny rastry jsem v programu IrfanView 10x zmenšil. Ve výpočetním skriptu jsem změnil hodnotu kvality skenování ze 400 na 40, čímž se načtené souřadnice rohů a id. bodů zmenšily také 10x. Více jsem upravovat nemusel. Transformace jednoho rastru trvala přibližně 25 minut a výsledek měl 10x menší rozlišení, než by bylo ideální, avšak stále dostatečné. Načetl jsem transformované rastry do programu ArcMap a zkontroloval souřadnice identických bodů. Zjistil jsem, že id. body po transformaci neodpovídají své poloze v S-JTSK.

9.2 REZIDUA NA IDENTICKÝCH BODECH

Provedl jsem následující pokus. Třemi způsoby jsem spočetl parametry TPS v proměnných *pY* a *pX* a kontrolně transformoval id. body. Vše jsem udělal dvakrát, za použití id. bodů z listů z celých Čech (6849 bodů) a za použití id. bodů pouze z výřezu přibližně odpovídajícího ústeckému kraji (1200 bodů). Zde jsou zjištěné maximální souřadnicové odchylky na id. bodech:

1) Parametry vypočtené pomocí funkce *tpaps*, aplikace transformace funkcí *fnval*.
   Maximální rezidua na id. bodech:
   pro 6849 id. bodů: $V_{Y\,max}$ = 14.0 km, $V_{X\,max}$ = 10.8 km
   pro 1200 id. bodů: $V_{Y\,max}$ = 10.4 km, $V_{X\,max}$ = 8.4 km

2) Parametry vypočtené analyticky, podle vzorců uvedených v kapitole 4 (o TPS), aplikace transformace funkcí *fnval*.
   Maximální rezidua na id. bodech:
   Pro 6849 id. bodů: $V_{Y\,max}$ = 1.5 km, $V_{X\,max}$ = 3.8 km
   Pro 1200 id. bodů: $V_{Y\,max}$ = 0.4 mm, $V_{X\,max}$ = 0.5 mm

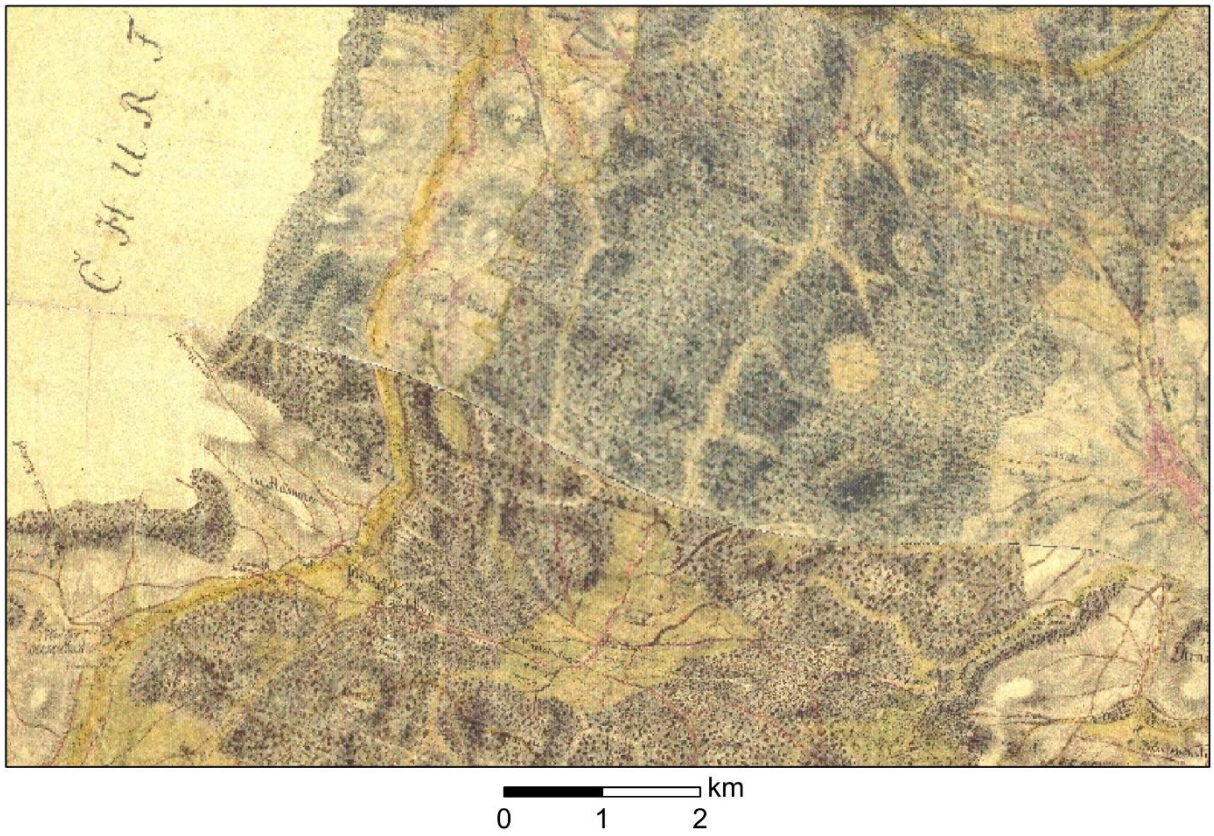
3) Parametry vypočtené analyticky, podle vzorců uvedených v kapitole 4 (o TPS), aplikace transformace také tak.
   Maximální rezidua na id. bodech:
   pro 6849 id. bodů: $V_{Y\,max}$ = 0.65 km, $V_{X\,max}$ = 1.78 km
   pro 1200 id. bodů: $V_{Y\,max}$ = 0.2 mm, $V_{X\,max}$ = 0.1 mm

Přesnost výpočtu funkce *tpaps* tedy selhává na množství použitých id. bodů. Analytický výpočet uvedený v kapitole o TPS nicméně také, ne však v takové míře. Důvod jsem z nedostatku času nezjistil. Nahradil jsem funkci *tpaps* analytickým výpočtem. Časová náročnost se tím v podstatě nezměnila.

10 VÝSLEDKY

Celý výpočetní skript funguje bez závad pouze pro menší oblast, než jsou celé Čechy. Mými výsledky je 44 transformovaných mapových listů zhruba z oblasti ústeckého kraje, o rozlišení 10x menším, než je rozlišení originálu.

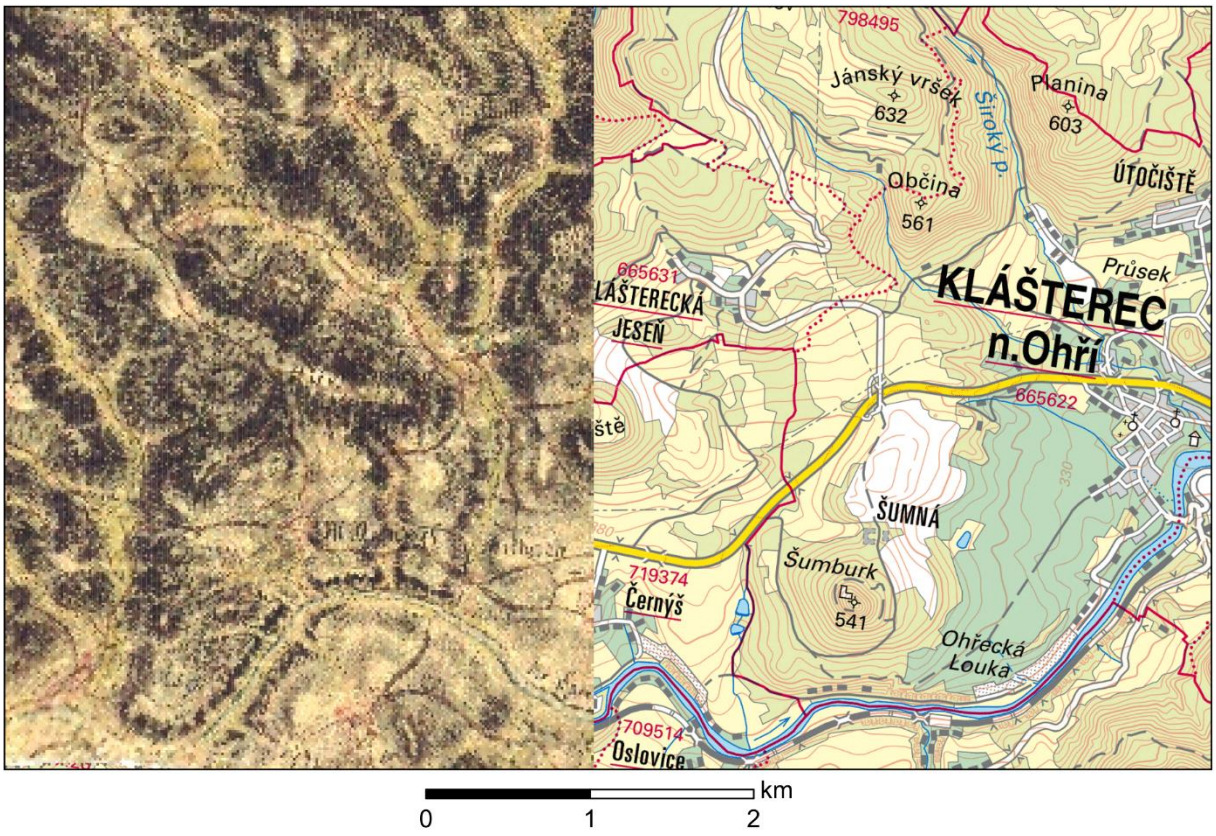
Výstupní rastry jsem nahrál do programu ArcMap a prohlédl si, jak na sebe jednotlivé mapové listy navazují (obr. 11).



Obr. 11 – ukázka návaznosti mapových listů

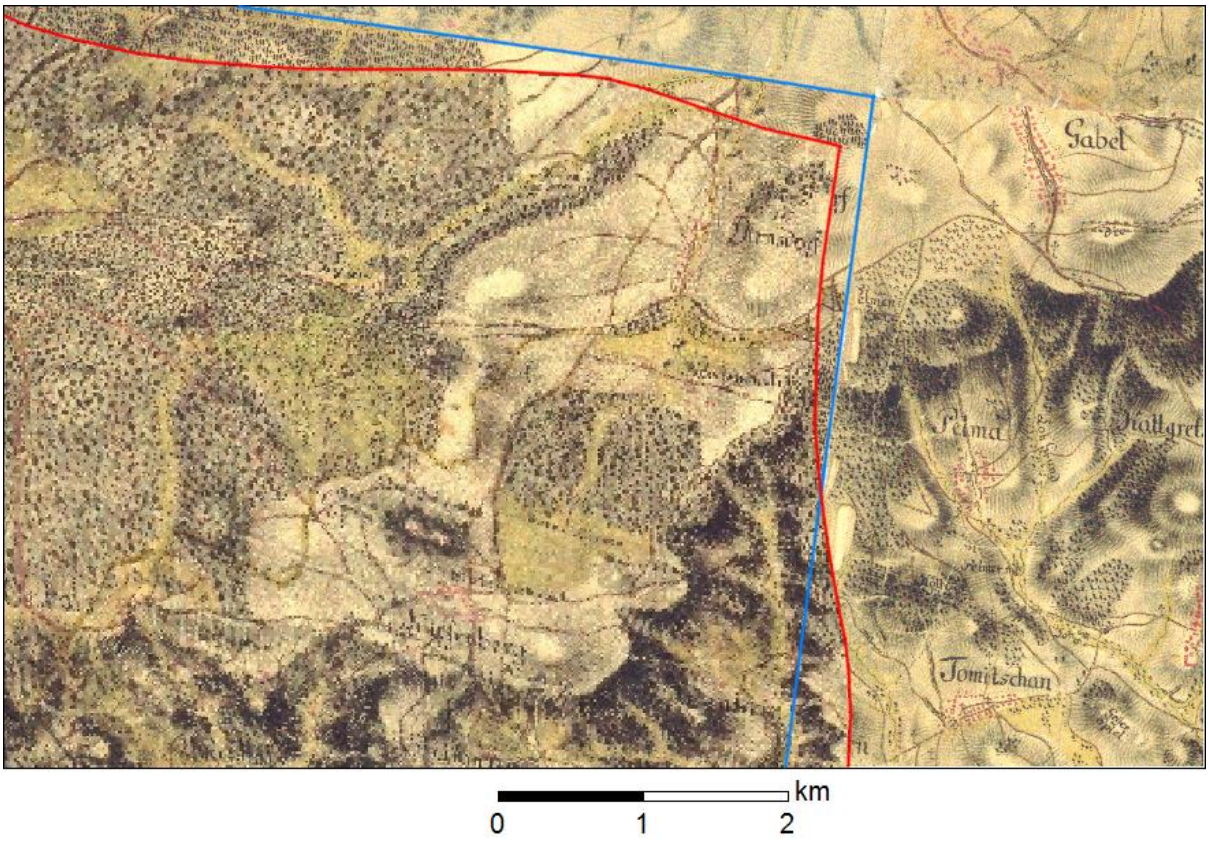
Poté jsem je porovnal s dalšíma dvěma mapovými podklady.

Zaprvé jsem vytvořil mapovou kompozici mnou transformovaného I. vojenského mapování a Základní mapy České Republiky 1 : 50 000 [10] (obr. 12).



Obr. 12 – mnou transformovaný rastr (vlevo) a Základní mapa 1 : 50 000 (vpravo)

Za druhé jsem své výsledky porovnal s výsledky aplikace MultiGeoref, poskytovanými na mapovém serveru Fakulty stavební [11] (obr. 13), jejichž charakteristiky jsou pro připomenutí: hromadné vyrovnání polynomickou transformací, metoda IRLS, střední souřadnicová chyba 280 m.



Obr. 13 – mnou transformovaný rastr (ohraničený červeně) a odpovídající rastr vytvořený aplikací MultiGeoref (ohraničený modře)

11 ZÁVĚR

Cílem mého úsilí bylo transformovat 250 rastrů I. vojenského mapování. Práce však selhala na výpočtech, které dávaly nejen nedostatečně přesné výsledky, ale navíc vyžadovaly mnohem výkonnější výpočetní hardware, než jakým jsem disponoval. Výpočetní skript pro Matlab sice funguje, ale zatím není použitelný na celou oblast Čech. Musel jsem se spokojit s výsledkem, který je stále ve fázi vývoje. Tento vývoj může i nadále pokračovat, ovšem už ne jako součást této závěrečné práce. Je potřeba analyzovat problém přesnosti výpočtu transformačních parametrů TPS a učinit kroky k jeho odstranění. Za zvážení také stojí, zda je Matlab dobrým výpočetním prostředím pro tento typ úlohy a zda by nešlo použít jiná, daleko vhodnější prostředí. Pak zbývá rozběhnout výpočet na zařízení, které je schopno jej v rozumné době dovést ke zdárnému cíli.

LITERATURA

[1] BOOKSTEIN, F.L. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. **11**(6), 567-585 [cit. 2018-05-27]. DOI: 10.1109/34.24792. ISSN 01628828. Dostupné z: http://ieeexplore.ieee.org/document/24792/

[2] CAJTHAML, Jiří. *Analýza starých map v digitálním prostředí na příkladu Müllerových map Čech a Moravy*. Praha: České vysoké učení technické v Praze, 2012. ISBN 978-80-01-05010-1.

[3] CAJTHAML, Jiří. *Nové technologie pro zpracování a zpřístupnění starých map*. Praha, 2007. Doktorská disertační práce. České vysoké učení technické v Praze.

[4] CAJTHAML, Jiří. Tvorba souvislé mapy I. vojenského mapování Habsburské monarchie – testovací oblast Ústecký kraj. *Geodetický a kartografický obzor*. 2013, **59/101**(8), s. 212 – 219.

[5] ČADA, Václav. *Robustní metody tvorby a vedení digitálních katastrálních map v lokalitách sáhových map*. Plzeň, 2003. Habilitační práce. Západočeská univerzita v Plzni.

[6] FIEDLEROVÁ, Tereza. *Aplikace pro georeferencování vícelistých mapových děl*. Praha, 2015. Diplomová práce. České vysoké učení technické v Praze.

[7] SIXTA, Tomáš. Cvičení 12: Elastická registrace pomocí klíčových bodů. In: *Center for Machine Perception*[online]. 2011 [cit. 2018-05-27]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/ZMO/labs/thin_plate_splines/cv12_thinplatesplines.pdf

[8] Warping Using Thin Plate Splines. *MathWorks* [online]. The MathWorks, 2009 [cit. 2018-05-27]. Dostupné z: https://www.mathworks.com/matlabcentral/fileexchange/24315-warping-using-thin-plate-splines

[9] World files for raster datasets. *ArcGIS Desktop* [online]. Environmental Systems Research Institute, 2016 [cit. 2018-05-27]. Dostupné z: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/world-files-for-raster-datasets.htm

[10] *Prohlížecí služba WMS – ZM 50* [online]. ČÚZK, 2010 [cit. 2018-05-27]. Dostupné z: http://geoportal.cuzk.cz/WMS_ZM50_PUB/WMService.aspx

[11] *Mapová služba – 1VM* [online]. [cit. 2018-05-27]. Dostupné z: http://gis.fsv.cvut.cz/arcgis

SEZNAM PŘÍLOH

A VÝPOČETNÍ SKRIPT PRO MATLAB (S KOMENTÁŘI)

```
%==========================================================================
%% TRANSFORMACE MAP I. VOJENSKEHO MAPOVANI METODOU THIN PLATE SPLINE - VYPOCETNI
SKRIPT
%...Autor: Jan Myslivec
%...Datum: 27.5.2018
%==========================================================================
%% NACTENI NEKTERYCH TEXTOVYCH DAT

% Zacatek mereni casu vypoctu
tic

% Kvalita skenovanych rastru
dpi = 40; %...tohle ma byt puvodne 400, 40 je pro zmensene rastry

% Schema rastru vstupujicich do vypoctu
scheme_fid = fopen('IN/rozlozeni_bezkrajnich_usti.txt');
scheme = fscanf(scheme_fid,'%f\n',[18,inf]);
scheme = scheme';

% Nacteni id. bodu
identicals_xyYX_matrix = [];
for col_j=1:size(scheme,2)
for row_i=1:size(scheme,1)
    sheet = sub2ind([size(scheme,1),size(scheme,2)],row_i,col_j);
    %...nacitame po sloupcich
    if scheme(row_i,col_j) > 0
    ids_fid = fopen(['IN/',num2str(scheme(sheet)),'_ib.txt']);
    identicals_xyYX_i = fscanf(ids_fid,'%f\n',[4,inf]);
    n = size(identicals_xyYX_i,2);
    identicals_xyYX_matrix = [identicals_xyYX_matrix;...
                      scheme(row_i,col_j)*ones(n,1),...
                      identicals_xyYX_i'];
    end
end
end

%--------------------------------------------------------------------------
%% VYPOCET ROZMERU A MERITKA ZPRACOVANYCH RASTRU

disp('...pocitam meritko vystupu')
% Vytvoreni sady id. bodu
xx = identicals_xyYX_matrix(:,2)*dpi;
yy = identicals_xyYX_matrix(:,3)*dpi;
YY = identicals_xyYX_matrix(:,4);
XX = identicals_xyYX_matrix(:,5);
XX_proj = zeros(size(xx));
YY_proj = zeros(size(xx));
yy_o = zeros(size(xx));

% Vypocet vsech vzdalenosti uvnitr kazdeho obrazku
sheet = identicals_xyYX_matrix(:,1);
R = zeros(length(sheet),1);
r = zeros(length(sheet),1);
for i=1:length(sheet)-1
    if sheet(i,1) == sheet(i+1,1)
        r(i) = sqrt((xx(i)-xx(i+1))^2+(yy(i)-yy(i+1))^2);
        R(i) = sqrt((XX(i)-XX(i+1))^2+(YY(i)-YY(i+1))^2);
    end
end
rmax = max(r);
```

```
for i=1:length(sheet)-1
    if r(i)<0.5*rmax
        r(i) = 0;
    end %...nejkratsi vzd na obrazku jsou nejmene relativne presne
        %...proto je potreba je eliminovat
end
% Vypocet meritka
%...melo by byt 1 : co nejmene, aby jsme zbytecne neztraceli pixely
scale = max(r./R);

%-------------------------------------------------------------------------------
%% VYPOCET TRANSFORMACNIHO KLICE THIN PLATE SPLINE

% Cyklus dlazdicovani rastru a vytvoreni sady id. bodu pro plosny...
% ...vypocet transformacniho klice TPS

disp('Zahajuji vypocet transformacniho klice:')
for col_j=1:size(scheme,2)
for row_i=1:size(scheme,1)
    sheet = sub2ind([size(scheme,1),size(scheme,2)],row_i,col_j);
    %...nacitame po sloupcich
if scheme(row_i,col_j) > 0
fprintf('...pocitam parametry proj. transf. pro rastr %4.0f\n',scheme(row_i,col_j))

% Nacteni tiffu
tiff_file = ['IN/',num2str(scheme(row_i,col_j)),'.jpg'];
tiff = imread(tiff_file);
imgH = size(tiff,1);  %...vyska
imgW = size(tiff,2);  %...sirka

% Nacteni souradnic rohu
corners_file = ['IN/',num2str(scheme(row_i,col_j)),'_rohy.txt'];
corners_fid = fopen(corners_file);
corners = fscanf(corners_fid,'%f %f \n',[2,inf]);
%...nacitame po sloupcich
corners = corners' * dpi;
% Otoceni osy y v obrazku (v textovem souboru jde nahoru, v matlabu jde dolu)
corners(:,2) = imgH - corners(:,2);

% Projektivni transformace - tvorba obdelniku z orezanych rastru
xc = corners(:,1);
yc = corners(:,2);
rectW = 101;
rectH  = 100;   %...na techto rozmerech nezalezi (rectH,rectW)...
%...roztahne vyrez na obdelnik vysoky rectH a siroky rectW.
XC = [0 rectW rectW 0]';
YC = [0 0 rectH rectH]';

% Vypocet parametru p proj. transformace
syms a b c d e f g h
[a,b,c,d,e,f,g,h] = solve(...
(a*xc(1)+b*yc(1)+c)/(g*xc(1)+h*yc(1)+1) == XC(1),...
(a*xc(2)+b*yc(2)+c)/(g*xc(2)+h*yc(2)+1) == XC(2),...
(a*xc(3)+b*yc(3)+c)/(g*xc(3)+h*yc(3)+1) == XC(3),...
(a*xc(4)+b*yc(4)+c)/(g*xc(4)+h*yc(4)+1) == XC(4),...
(d*xc(1)+e*yc(1)+f)/(g*xc(1)+h*yc(1)+1) == YC(1),...
(d*xc(2)+e*yc(2)+f)/(g*xc(2)+h*yc(2)+1) == YC(2),...
(d*xc(3)+e*yc(3)+f)/(g*xc(3)+h*yc(3)+1) == YC(3),...
(d*xc(4)+e*yc(4)+f)/(g*xc(4)+h*yc(4)+1) == YC(4));
p.a(sheet)=eval(a);p.b(sheet)=eval(b);p.c(sheet)=eval(c);p.d(sheet)=eval(d);
p.e(sheet)=eval(e);p.f(sheet)=eval(f);p.g(sheet)=eval(g);p.h(sheet)=eval(h);
```

```
% Proj. transformace id. bodu
for i=1:length(identicals_xyYX_matrix)
    if identicals_xyYX_matrix(i,1) == scheme(row_i,col_j)
    %...naleza-li se id. bod v danem obrazku, transformujeme ho...
    %...a posuneme podle poradi obrazku ve schematu
       p.shift_x(sheet) = (col_j - 1)*rectW;
       p.shift_y(sheet) = (row_i - 1)*rectH;
       %...otoceni osy y v obrazku (v textaku jde nahoru, v matlabu jde dolu)
       yy_o(i) = imgH - yy(i);
       XX_proj(i) = (p.a(sheet)*xx(i)+p.b(sheet)*yy_o(i)+p.c(sheet))/...
                    (p.g(sheet)*xx(i)+p.h(sheet)*yy_o(i)+1) + p.shift_x(sheet);
       YY_proj(i) = (p.d(sheet)*xx(i)+p.e(sheet)*yy_o(i)+p.f(sheet))/...
                    (p.g(sheet)*xx(i)+p.h(sheet)*yy_o(i)+1) + p.shift_y(sheet);
       %...kontrolne znovu transformovane rohy
       % XC_proj = (p.a(sheet)*xc+p.b(sheet)*yc+p.c(sheet))./...
       %           (p.g(sheet)*xc+p.h(sheet)*yc+1) + p.shift_x(sheet);
       % YC_proj = (p.d(sheet)*xc+p.e(sheet)*yc+p.f(sheet))./...
       %           (p.g(sheet)*xc+p.h(sheet)*yc+1) + p.shift_y(sheet);
    end
end

end
end
end

% Vypocet parametru TPS
disp('...pocitam parametry TPS')
xxp = XX_proj';
yyp = YY_proj';
nn = length(xxp);
ri2 = zeros(nn,nn);
for i=1:nn
ri2(i,:) = (xxp-xxp(i)).^2 + (yyp-yyp(i)).^2;
end
r1i2 = ri2;
r1i2(find(ri2==0))=realmin; % Avoid log(0)=inf
K = (ri2).*log(r1i2);
L1 = [0 0 0;0 0 0;0 0 0;];
P = [xxp;yyp;ones(1,nn)];
L  = [K P'
      P L1];
z = [YY-xxp',XX-yyp';0 0;0 0;0 0];
parameters = L\z;
parameters(end-2,1) = parameters(end-2,1) + 1;  %...lehka kosmeticka uprava
parameters(end-1,2) = parameters(end-1,2) + 1;  %...lehka kosmeticka uprava

% Ukladani parametru do TPS
pY.form = 'st-tp00';
pY.centers = [xxp;yyp];
pY.coefs = parameters(:,1)';
pY.ncenters = nn;
pY.number = nn+3;
pY.dim = 1;
pY.interv = {[min(xxp) max(xxp)]  [min(yyp) max(yyp)]};
pX.form = 'st-tp00';
pX.centers = [xxp;yyp];
pX.coefs = parameters(:,2)';
pX.ncenters = nn;
pX.number = nn+3;
pX.dim = 1;
pX.interv = {[min(xxp) max(xxp)]  [min(yyp) max(yyp)]};

% Kontrola parametru TPS:
avals_Y = fnval(pY,[xxp;yyp]);
avals_X = fnval(pX,[xxp;yyp]);
VYtpaps = avals_Y' - YY;
VXtpaps = avals_X' - XX;
```

```
% Max. odchylky kontrolni transformace id. bodu:
max_odch = max(abs([VYtpaps,VXtpaps]));

%...struktura pX.coefs: 1.... = bX/2
%                       end-2 = aX1 + 1 (?!)
%                       end-1 = aX2
%                       end   = aX0
```

```
%================================================================================
%% CYKLUS PRO JEDNOTLIVE RASTRY

% Schema rastru urcenych k transformaci
scheme_fid = fopen('IN/rozlozeni_bezkrajnich_usti.txt');
scheme = fscanf(scheme_fid,'%f\n',[18,inf]);
scheme = scheme';

disp('Zahajuji zpracovani rastru:')
for col_j=1:size(scheme,2)
for row_i=1:size(scheme,1)
    sheet = sub2ind([size(scheme,1),size(scheme,2)],row_i,col_j);
    %...nacitame po sloupcich
if scheme(row_i,col_j) > 0
fprintf('...zpracovavam rastr %4.0f\n',scheme(row_i,col_j))

%--------------------------------------------------------------------------------
%% OREZANI RASTRU PODLE SOURADNIC ROHU

% Nacteni tiffu
fprintf('......[load]')
tiff_file = ['IN/',num2str(scheme(row_i,col_j)),'.jpg'];
tiff = imread(tiff_file);
imgH = size(tiff,1);
imgW = size(tiff,2);

% Neupraveny obrazek:

            %>>> OBRAZEK <<<
            figure(scheme(row_i,col_j))
            imshow(tiff)

% Nacteni souradnic rohu
fprintf('...[crop]')
corners_file = ['IN/',num2str(scheme(row_i,col_j)),'_rohy.txt'];
corners_fid = fopen(corners_file);
corners = fscanf(corners_fid,'%f %f \n',[2,inf]);
corners = corners' * dpi;
% Otoceni osy y v obrazku (v textaku jde nahoru, v matlabu jde dolu)
corners(:,2) = imgH - corners(:,2);

% Vypocet reznych primek
syms x y
%...cropline: y = (y2-y1)/(x2-x1) * (x-x1) + y1
%...cropline: y = a * (x-b) + c
a1 = (corners(2,2)-corners(1,2))/(corners(2,1)-corners(1,1));
b1 = corners(1,1);
c1 = corners(1,2);
cropline1 = a1*(x-b1)+c1;
a2 = (corners(2,1)-corners(3,1))/(corners(2,2)-corners(3,2));
b2 = corners(3,2);
c2 = corners(3,1);
cropline2 = a2*(y-b2)+c2;
a3 = (corners(4,2)-corners(3,2))/(corners(4,1)-corners(3,1));
b3 = corners(3,1);
c3 = corners(3,2);
cropline3 = a3*(x-b3)+c3;
a4 = (corners(4,1)-corners(1,1))/(corners(4,2)-corners(1,2));
b4 = corners(1,2);
c4 = corners(1,1);
cropline4 = a4*(y-b4)+c4;
```

```
% Orezani
x=1:imgW;
y=1:imgH;
lim_up = eval(cropline1);
lim_down = eval(cropline3);
lim_right = eval(cropline2);
lim_left = eval(cropline4);
%...Lehka kosmeticka uprava:
if size(lim_up)<2
    lim_up = ones(1,length(x))*lim_up;
end
if size(lim_down)<2
    lim_down = ones(1,length(x))*lim_down;
end
if size(lim_right)<2
    lim_right = ones(1,length(y))*lim_right;
end
if size(lim_left)<2
    lim_left = ones(1,length(y))*lim_left;
end
%...je-li totiz rezna primka vodorovna ci svisla,
%...pak z ni matlab udela jedno cislo.

[x_grid, y_grid] = meshgrid(0.5:imgW-0.5,0.5:imgH-0.5); %...uvazujme stredy pixelu
for yi=y
for xi=x
    if tiff(yi,xi,:)==0
        tiff(yi,xi,:)=1;
        %...zesvetli absolutni cernou v obrazku, aby nebyla pozdeji...
        %...povazovana za pruhlednou
    end
    if lim_up(xi)>y_grid(yi,xi) || y_grid(yi,xi)>lim_down(xi) ...
        || lim_left(yi)>x_grid(yi,xi) || x_grid(yi,xi)>lim_right(yi)
        tiff(yi,xi,:)=0;
        %...oreze obrazek
    end
end
end

% Orezany obrazek:

            %>>> OBRAZEK <<<
            figure(scheme(row_i,col_j)+1000)
            imshow(tiff)

%-------------------------------------------------------------------------
%% THIN PLATE SPLINE

NPs = length(identicals_xyYX_matrix); %...pocet id. bodu

% Aplikace projektivni transformace
fprintf('...[proj]')
x = x_grid;
y = y_grid;
xp = (p.a(sheet)*x+p.b(sheet)*y+p.c(sheet))./...
     (p.g(sheet)*x+p.h(sheet)*y+1) + p.shift_x(sheet);
yp = (p.d(sheet)*x+p.e(sheet)*y+p.f(sheet))./...
     (p.g(sheet)*x+p.h(sheet)*y+1) + p.shift_y(sheet);

% Grid po prijektivni transformaci:

            %>>> OBRAZEK <<<
            figure(scheme(row_i,col_j)+2000)
            plot(xp,-yp)
```

```
% Aplikace TPS
fprintf('...[TPS]')

% A) analytickou cestou
% ri = zeros(imgH,imgW,NPs);
% for i=1:NPs
% ri(:,:,i) = sqrt((xp-xxp(i)).^2 + (yp-yyp(i)).^2);
% end
% r1i = ri;
% r1i(find(ri==0))=realmin; % Avoid log(0)=inf
% basis = 2*(ri.^2).*log(r1i);
% coefsY = repmat( permute(pY.coefs(1:end-3)',[3,2,1]), [imgH,imgW] );
% adjustmentY = basis .* coefsY;
% adjustmentY = sum(permute(adjustmentY,[3,2,1]));
% adjustmentY = permute(adjustmentY,[3,2,1]);
% coefsX = repmat( permute(pX.coefs(1:end-3)',[3,2,1]), [imgH,imgW] );
% adjustmentX = basis .* coefsX;
% adjustmentX = sum(permute(adjustmentX,[3,2,1]));
% adjustmentX = permute(adjustmentX,[3,2,1]);
% Yw = pY.coefs(end) + ...
%      pY.coefs(end-2)*xp + ...
%      pY.coefs(end-1)*yp + ...
%      adjustmentY;
% Xw = pX.coefs(end) + ...
%      pX.coefs(end-2)*xp + ...
%      pX.coefs(end-1)*yp + ...
%      adjustmentX;

% B) funkci fnval
Yw = zeros(imgH,imgW);
Xw = Yw;
for i=1:imgH
     Yw(i,:) = fnval(pY,[xp(i,:);yp(i,:)]);
     Xw(i,:) = fnval(pX,[xp(i,:);yp(i,:)]);
end

% Grid po TPS:

             %>>> OBRAZEK <<<
             figure(scheme(row_i,col_j)+3000)
             plot(Yw,Xw)
             pbaspect([1 1 1])
%-------------------------------------------------------------------------
%% INTERPOLACE PIXELU

%...nasledujici cast je z vetsiny prevzata
%...od Fitzgeralda J. Archibalda

% Anchor - ukotveni rastru
for i=1:length(identicals_xyYX_matrix)
    if identicals_xyYX_matrix(i,1) == scheme(row_i,col_j)
        anchor_Y = YY(i);
        anchor_X = XX(i);
        %...pro kazdy obrazek jedna kotva, a to prvni id. bod
        break
    end
end

% Stanoveni parametru interpolace
interp.method = 'nearest'; %...metoda 'nejblizsi soused'
                %...['nearest'/'invdist'/'none']
interp.radius = 5;
                %...maximalni radius pro interpolaci
                %...metodou 'nejblizsi soused'...
[x, y] = meshgrid(1:imgW,1:imgH);
```

```
% Interpolace na zaklade zvolene metody
fprintf('...[interp]')
[TIFF,TIFFr,map,worldfile] =
interp2d_edited(x(:),y(:),tiff,Yw(:),Xw(:),interp,scale,anchor_Y,anchor_X);

% Vysledny obrazek:

            %>>> OBRAZEK <<<
            figure(scheme(row_i,col_j)+4000)
            imshow(TIFF/255)

%-------------------------------------------------------------------------
%% EXPORT VYSLEDKU
% Generace rastru
fprintf('...[export]')
imwrite((TIFF/255),['OUT/',num2str(scheme(row_i,col_j)),'_trans.png'],...
        'BitDepth',8);

% Generace world filu k rastru
world_fid = fopen(['OUT/',num2str(scheme(row_i,col_j)),'_trans.pgw'],'w');
fprintf(world_fid,'%13.7f\n %1.0f\n %1.0f\n %13.7f\n %13.3f\n %13.3f\n',worldfile);
fclose(world_fid);

fprintf('...[done]\n')

end
end
end

disp('Hotovo.')
disp(' ')
toc %...konec mereni casu vypoctu
%=========================================================================
```

B SKRIPTY POUŽITÝCH FUNKCÍ (MIMO INTEGROVANÉ)

```
function [imgw, imgwr, map, worldfile] =
interp2d_edited(x,y,img,Ywr,Xwr,interp,scale,anchor_Y,anchor_X)
% Description:
% Interpolation of the warped grid using nearest neighbor and inverse
% distance weighted interpolation
%
% Inputs:
% x,y - mesh grid of input image
% Ywr, Xwr - warped grid
% img - input image
% outHplus, outWplus - expanded output warped image dimensions
% interp.method - interpolation mode('nearest', 'invdist', 'none')
% interp.radius - Max radius for nearest neighbor interpolation or
%                 Radius of inverse weighted interpolation
% interp.power - power for inverse weighted interpolation
% scale - scale between input image grid and warped grid (x,y / Ywr,Xwr)
% anchor_Y,anchor_X = anchor coordinates in JTSK system
%
% Output:
% imgw - interpolated image (warp+interpolate)
% imgwr - warped image with holes
% map - Map of the canvas with 0 indicating holes and 1 indicating pixel
%
% Author: Fitzgerald J Archibald
% Date: 23-Apr-09

%% initialization
% window dimension for filling
maxhw = fix((interp.radius-1)/2);
% input image params
color = size(img,3);
imgH  = size(img,1);
imgW  = size(img,2);

%% Round warping coordinates
% rounding warped coordinates
Ywi = round(Ywr*scale);
Xwi = round(-Xwr*scale);
anchor_imgY = anchor_Y*scale;
anchor_imgX = -anchor_X*scale;

% Bound warping coordinates to image frame
%...zbytecne orezava obrazek - to nechceme => vypustime
%...a rozsirime rozmery vystupu na outHplus a outWplus.
additional_expand = 0;
outWplus = max(Ywi)-min(Ywi)+additional_expand;
outHplus = max(Xwi)-min(Xwi)+additional_expand;
imgwr=zeros(outHplus,outWplus,color); %...output image - rozsireny
%...Xwi a Ywi jdou do minusu.
fit_edge = floor(additional_expand/2)-1;
anchor_imgY = anchor_imgY - min(Ywi)+fit_edge;
anchor_imgX = anchor_imgX - min(Xwi)+fit_edge; %...souradnice kotvy v obrazku
Ywi = Ywi - min(Ywi)+fit_edge;
Xwi = Xwi - min(Xwi)+fit_edge;
Ywi = max(Ywi,1);
Xwi = max(Xwi,1);

% Convert 2D coordinates into 1D indices
fiw = sub2ind([outHplus,outWplus],Xwi,Ywi); % warped coordinates
fip = sub2ind([imgH,imgW],y,x); % input
```

```
%% Warped image construction with holes
% Note: This mapping doesn't treat 2 or more points (from source image)
%     mapping to same point (on warped image) differently than 1-1 mapping.
o_r=zeros(outHplus,outWplus);
for colIx = 1:color,
    img_r=img(:,:,colIx);
    o_r(fiw)=img_r(fip);
    imgwr(:,:,colIx)=o_r;
end

%% Filling of holes

% Create a binary image with 0s for holes and 1s for mappings
map=zeros(outHplus,outWplus);
map(fiw)=1;

% Fill holes
if strcmp(interp.method,'nearest') % Median of nearest neigbor filling
    imgw = nearestInterp(imgwr, map, maxhw); % fill

else
    imgw = imgwr;
end


%% Anchor
worldfile = [
    1/scale
    0
    0
    -1/scale
    anchor_Y - anchor_imgY/scale
    anchor_X + anchor_imgX/scale
    ];

return;
```

```
function out = nearestInterp(imgw, map, maxhw )
% Description:
% Fill holes using nearest neighbor (median filter of neighbors) interpolation
%
% Inputs:
% imgw - input image
% map - Map of the canvas with 0 indicating holes and 1 indicating pixel
% maxhw - Max radius for nearest neighbor interpolation
%
% Output:
% out - interpolated image
%
% Author: Fitzgerald J Archibald
% Date: 23-Apr-09

outH  = size(imgw,1);
outW  = size(imgw,2);
out = imgw;

[yi_arr, xi_arr] = find(map==0); % Find locations needing fill
if isempty(yi_arr) == false
    color = size(imgw,3);
    for ix = 1:length(yi_arr),

        xi = xi_arr(ix);
        yi = yi_arr(ix);

        % Find min window which has non-hole neighbors
        nz = false;
        for h = 1:maxhw,
            yixL=max(yi-h,1);
            yixU=min(yi+h,outH);
            xixL=max(xi-h,1);
            xixU=min(xi+h,outW);

            if isempty(find(map(yixL:yixU,xixL:xixU), 1)) == false
                nz = true;
                break;
            end
        end

        % use the median of non-hole neighbors in the window for filling
        if nz == true,
            for colIx = 1:color,
                win=imgw(yixL:yixU, xixL:xixU, colIx);
                out(yi,xi,colIx) = median(win(find(map(yixL:yixU, xixL:xixU)~=0)));
            end
        end

    end
end

return;
```