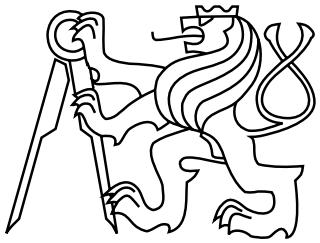
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEOMATIKA



DIPLOMOVÁ PRÁCE
NÁZEV

Vedoucí práce: Jméno vedoucího
Katedra geomatiky

červen 2020

Tomáš KLEMSA

ABSTRAKT

Abstrakt.....

KLÍČOVÁ SLOVA

klíčová slova

ABSTRACT

Abstract.....

KEYWORDS

Key worlds

PROHLÁŠENÍ

Prohlašuji, že diplomovou práci na téma „Název“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

PODĚKOVÁNÍ

Poděkování...

Obsah

1 Úvod **6**

1.1 Cíle práce 7

2 Rešerše používaných algoritmů **8**

2.1 Výpočet průsečíků množiny linií 8

2.1.1 Vzájemná poloha dvou úseček 8

2.1.2 Nalezení průsečíku dvou úseček 9

2.1.3 Bentley–Ottmannův algoritmus 9

2.2 Tvorba polygonů z množiny linií 10

2.2.1 Nalezení všech cyklů v grafu 10

2.3 GIS software 10

2.3.1 ArcGIS 10

2.3.2 QGis 11

2.3.3 Grass GIS 11

2.3.4 PostGis 11

3 Závěr **12**

Literatura **15**

A Elektornické přílohy **I**

A.1 CD disk I

1 Úvod

Fotogrammetrie je obor zabývající se geometrickými vztahy mezi objekty zachycenými na fotografickém snímku. Tento způsob se často využívá u dokumentace historických objektů, tvorby map z leteckých snímků, při dokumentaci dopravních nehod atd. Fotogrammetrii můžeme rozdělit do více kategorií.

Jednosnímková fotogrammetrie

Nejjednodušší metoda je tzv. jednosnímková fotogrammetrie. Jak název napovídá, je zpracováván pouze jeden snímek. Metoda je hojně využívána u dokumentace fasád budov. Omezení při tomto postupu je, že ze snímku získáme pouze dvojrozměrnou informaci, proto jakákoliv hloubková členění budou oproti zvolené rovině zkreslená.

Stereofotogrammetrie

Metoda využívající stereoskopického vjemu, na jehož základě získáme z dvou překrývajících se snímků s rovnoběžnou osou záběru prostorové informace. Tato metoda je hojně využívána v letecké fotogrammetrii.

Průseková fotogrammetrie

Průseková fotogrammetrie využívá více snímků s nerovnoběžnou osou záběru, na kterých jsou vyhledány identické body, ze kterých jsou získané prostorové souřadnice těchto bodů.

Obrazová korelace

Obrazová korelace, hojně nazývána také jako IBMR (*Image Based Modeling and Rendering*) je metoda využívající obrazovou korelaci pro vyhledávání identických bodů. Funguje v podstatě na principu průsekové fotogrammetrie s tím rozdílem, že je potřeba velké množství snímků a algoritmus SIFT (*Scale Invariant Feature Transformation*), který vyhledává identické body, proto je vhodná pro objekty negeometrického tvaru.

Všechny tyto metody využívají obrazová data, u kterých je předpoklad středového promítání. To ovšem u fotoaparátů s běžným objektivem neplatí. Fotografie jsou zatíženy geometrickým zkreslením, které by se projevilo na kvalitě výstupu. Proto je nutné toto zkreslení redukovat.

Tato bakalářská práce je zaměřena na tvorbu jednoduchého a uživatelsky přívětivého softwaru pro odstranění geometrického zkreslení fotografických snímků na

základě znalosti prvků vnitřní orientace včetně průběhu radiální a tangenciální distorze. Průběh distorze může být vyjádřen různými způsoby. Pro tvořený program byl však zvolen model podle Duane C. Browna, který je často využívaným modelem ve fotogrammetrických softwarech.

Existuje mnoho programů, které umí nežádoucí zkreslení ze snímků odstranit. Proč tedy vyvíjet nástroj nový?

Jako první bych zmínil grafické editory. Jedná se o nástroje, které většinou umí odstranit zkreslení na základě subjektivního vnímání uživatele. Například tak, že posuvníkem uživatel nastavuje velikost zkreslení, až stěny budovy nejsou prohnuté. Tyto nástroje jsou ovšem vhodné pouze pro fotografie, které nebudou fotogrammetricky zpracovávány. Nepracují totiž s tangenciální distorzí, souřadnicemi hlavního snímkového bodu a nastavení průběhu radiální distorze je často velmi omezené.

Do druhé skupiny bych zařadil programy fotogrammetrické. Často profesionální a velmi obsáhlý software, který je ovšem v mnohých případech finančně nákladný. V některých případech tyto programy mají funkci na poloautomatický výpočet prvků vnitřní orientace, což je značná úspora času.

1.1 Cíle práce

Cílem této práce je tedy vytvořit program, který bude volně šiřitelný, aby kdokoliv měl možnost upravovat, šířit a dále využívat zdrojový kód a bude psán v rozšířeném multiplatformním objektově orientovaném programovacím jazyce, tak aby byla snadná jeho další úprava. Program byl tedy vytvořen v jazyce *C++* na platformě *Qt*.

2 Rešerše používaných algoritmů

Tato kapitola se zabývá přehledem využívaných algoritmů pro tvorbu polygonů z množiny linií. Polygonizace, tak jak jí můžeme pozorovat v různých GIS softwarech, se zpravidla neřeší jedním algoritmem, ale používají se různé algoritmy pro jednotlivé kroky polygonizace. Popíšeme zde tedy do jakých kroků lze polygonizaci rozdělit a jak jednotlivé kroky optimálně řešit. Budeme se zabývat i řešením v jednotlivých GIS softwarech.

Polygonizaci lze nejjednodušeji rozdělit na 2 základní kroky. Jako vstupní data budeme uvažovat množinu linií, tyto linie ovšem nemusí obsahovat, v reálném použití převážně neobsahují, lomové body ve společných průsečících. V terminologii GIS se tyto průsečíky označují jako *fuzzy*, volně přeloženo jako *neostré*. Abychom mohli zahájit další krok polygonizace, je nutné tyto průsečíky doplnit. V první části se tedy budeme zabývat, jak optimálně doplnit průsečíky množin linií.

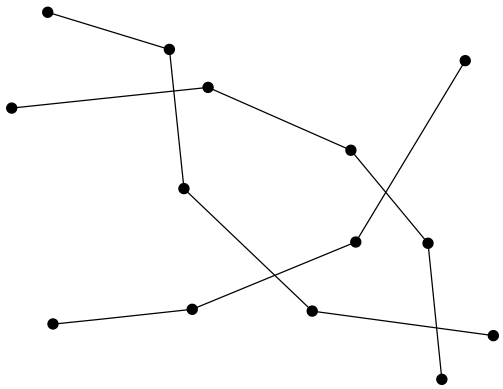
Předpokládejme že množina linií je zbavená *fuzzy* průsečíků, tedy každá dvojice úseček má společný nanejvýš jeden koncový bod. V takovéto situaci můžeme přistoupit k dalšímu kroku, tedy vlastní tvorbě polygonů z množiny linií. Polygony, které hledáme, musí splňovat podmínku, že žádný námi vytvořený polygon nemůže být rozdělen jakoukoliv linií na více menších polygonů. Pro řešení tvorby polygonů se často využívá grafových algoritmů.

2.1 Výpočet průsečíků množiny linií

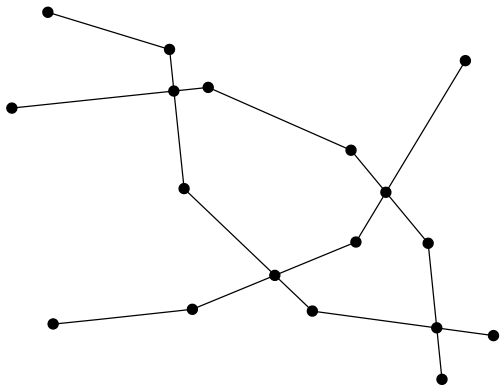
Výpočet všech průsečíků množin linií lze provést snadno, testováním všech úseček se všemi. Tímto postupem ovšem zjevně dosáhneme složitosti $\mathcal{O}(N^2)$. Pro výpočet průsečíků lze ovšem využít i algoritmus s časovou náročnosti $\mathcal{O}(n \log n)$, známý také jako Bentley–Ottmannův algoritmus [2].

2.1.1 Vzájemná poloha dvou úseček

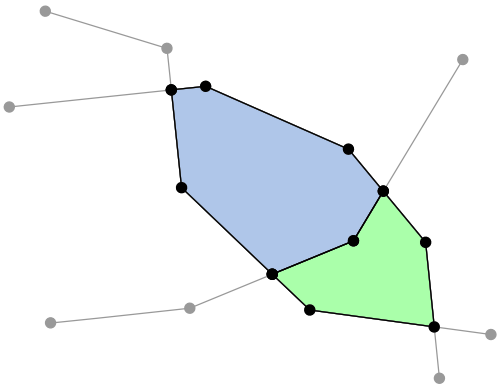
Ve 2D výpočetní geometrii jsou standardně jednotlivé segmenty linie vyjádřeny počátečním a koncovým bodem. Uvažujme tedy že máme dány 2 přímky $p_1 = |S_1E_1|$ a $p_2 = |S_2E_2|$, kde $S_1 = [x_{S1}, y_{S1}]$, $E_1 = [x_{E1}, y_{E1}]$, $S_2 = [x_{S2}, y_{S2}]$, $E_2 = [x_{E2}, y_{E2}]$ a potřebujeme provést test, zda se dané přímky protínají či nikoliv, můžeme použít takzvaný *Half-Plane* test, tedy test, který určuje zda bod leží v pravé či levé polorovině od přímky. Tento test zopakujeme celkem čtyřikrát a to na počáteční a koncový



(a) Množina linií s fuzzy průsečíky



(b) Množina linií doplněná o průsečíky



(c) Vytvořené polygony

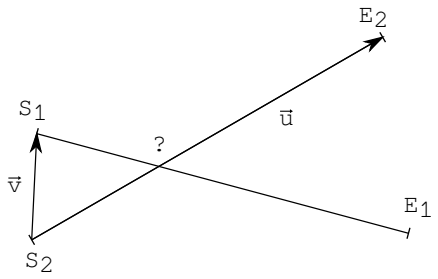
Obrázek 2.1: Znázornění postupu polygonizace

bod druhé úsečky, abysme zjistili zda se přímky protínají. Test je založen na výpočtu orientace dvou vektorů $\vec{u}$ a $\vec{v}$, kde vektor $\vec{v}$ je směrový vektor úsečky, tedy $\overrightarrow{S_1E_1}$ a vektor $\vec{v}$ je vektor $\overrightarrow{S_1S_2}$.

2.1.2 Nalezení průsečíku dvou úseček

2.1.3 Bentley–Ottmannův algoritmus

Tento algoritmus je založen na technice zvané *sweep line*, překládané jako *zametací přímka*, pro kterou je předpokladem mít setříděná vstupní data, podle jedné ze souřadnic. Algoritmus zde nebuda více popisován jelikož je velmi dobře vysvětlen v těchto publikacích [2] [1].



Obrázek 2.2: *Half-plane* test

2.2 Tvorba polygonů z množiny linií

Předpokládejme že máme množinu linií doplněnou o průsečíky, tedy každá dvojice úseček v této množině sdílí nejvýše jeden koncový bod. V předchozí sekci bylo řečeno že průsečíky linií lze doplnit pomocí *Bentley–Ottmannova* algoritmu v čase $\mathcal{O}(n \log n)$. Nyní chceme tedy v této množině linií nalézt všechny polygony, tak aby jednotlivé polygony spolu sdílely maximálně hrany a zároveň aby žádný z polygonů nemohl obsahovat jiný polygon.

2.2.1 Nalezení všech cyklů v grafu

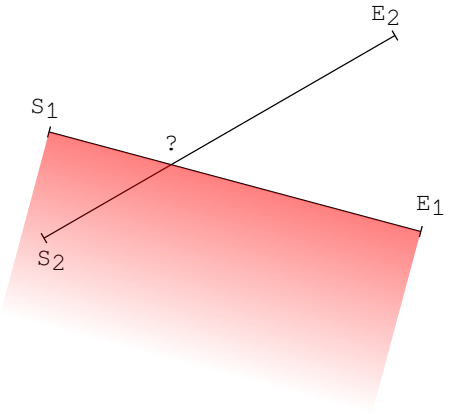
Jak již bylo řečeno, polygonizace se provádí nad daty s doplněnými průsečíky linií. Doplnění průsečíku jsme schopni vyřešit v čase $\mathcal{O}(n \log n)$. Nyní tedy nastává otázka jak řešit vlastní polygonizaci.

2.3 GIS software

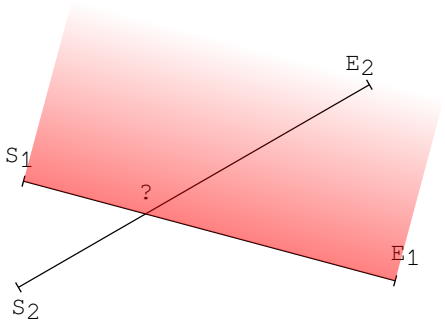
Provedení polygonizace zvládá většina současně používaného GIS softwaru. Pokusíme se tedy rozebrat postupy jednotlivých nástrojů na polygonizaci.

2.3.1 ArcGIS

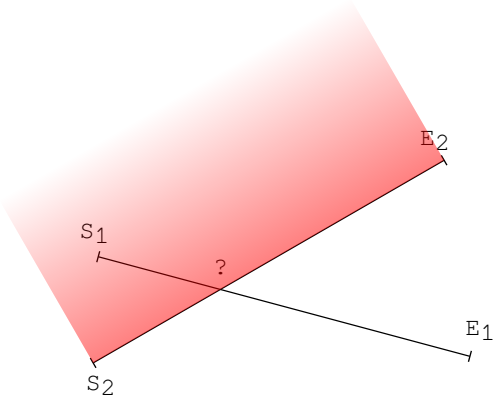
ArcGIS je vyvíjený společností Esri, v současné době se jedná o nejpokročilejší nástroj v oblasti GIS. Jedná se o proprietární software, u kterého



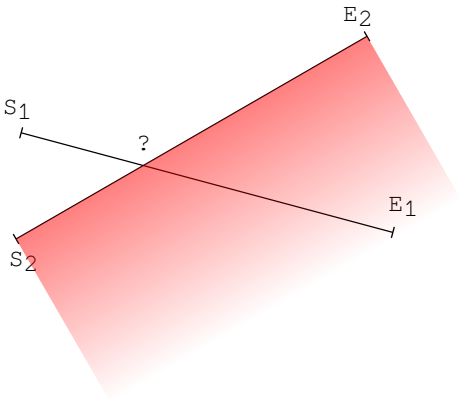
(a) Test bodu S_2 k přímce $|S_1E_1|$



(b) Test bodu E_2 k přímce $|S_1E_1|$



(c) Test bodu S_1 k přímce $|S_2E_2|$



(d) Test bodu E_1 k přímce $|S_2E_2|$

Obrázek 2.3: Grafické znázornění *Half-plane* testů pro zjištění existence průsečíku

2.3.2 QGis

QGis je nejspíše jeden z nejznámějších volných nástrojů pro práci v GIS. Je šířen pod copyleftovou licencí *GNU General Public License*, tudíž máme volný přístup ke zdrojovému kódu aplikace dostupných v online repozitářích. To nám umožňuje nahlížet do výpočetních algoritmů, které jsou v případě QGis psány v programovacím jazyce *C++* a *Python*, narozdíl od komerčních nástrojů, které si implementaci často chrání

2.3.3 Grass GIS

2.3.4 PostGis

3 Závěr

V této práci byla zpracována stručná rešerše literatury, zabývající se metodami odstranění zkreslení z fotografických snímků. Vybrané metody odstranění zkreslení jsou dále v práci popsány. Jako nejvhodnější způsob se jevilo odstranění zkreslení za pomoci *Brownova* distorzního modelu, který je v obdobných softwarech často používaný.

Hlavním cílem této práce bylo vytvořit softwarové řešení tohoto problému, které bude mít moderní grafické rozhraní. Pro tvorbu softwaru byl zvolen programovací jazyk *C++* s využitím knihoven *Qt*. Tento nástroj usnadnil tvorbu grafického rozhraní, které je zkoncipováno tak, aby mohl uživatel intuitivně program ovládat a nebylo zapotřebí podrobnější znalosti softwaru. Program byl sestaven pro platformu *Windows*, jelikož je však jazyk *C++* s knihovnami *Qt* určen pro různé platformy, nic nebrání tento program sestavit i pro *Linux* či *macOS*, což tento software dělá použitelnější pro větší okruh lidí. Dalším vylepšením vytvořeného programu *DistortionRemover* by do budoucna mohlo být obohacení o výpočet parametrů prvků vnitřní orientace kamery, které v současné době musí uživatel získat z jiného softwaru, či jiného zdroje. Velkým zlepšením by mohlo také být práce s obrazovými daty typu RAW, zamezilo by se tak velkým ztrátám kvality při opravě snímku. Samozřejmostí je opravení chyb, v programu, které budou odhaleny rozsáhlejším používáním.

Ve finální části, je předvedena práce s programem na příkladu historického stavebního objektu z databáze PhotoPa a jsou nastíněny další možné postupy zpracování.

Seznam obrázků

2.1 Znázornění postupu polygonizace 9
2.2 *Half-plane* test 10
2.3 Grafické znázornění *Half-plane* testů pro zjištění existence průsečíku . 11

/obrazky

Literatura

[1] BAYER, Tomáš. *Algoritmy v digitální kartografii.* : Karolinum, 2008.

[2] BENTLEY, Jon Louis a OTTMANN, Thomas A. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on computers.* 1979, , č. 9, s. 643–647.

[3] BERG, Mark et al. *Computational Geometry: Algorithms and Applications.* Berlin/Heidelberg: Springer Berlin Heidelberg, third edition, 2008. ISBN 9783642096815;3642096816;3540779736;9783540779735;.

[4] QGIS Development Team. *QGIS Geographic Information System.* Open Source Geospatial Foundation, 2009. Dostupné z: `http://qgis.osgeo.org`.

[5] SOJKA, Eduard; NĚMEC, Martin a FABIÁN, Tomáš. Matematické základy počítačové grafiky, 2011.

A Elektornické přílohy

A.1 CD disk

`BP-DistortionRemover.pdf`

text bakalářské práce ve formátu *.pdf,

`DistortionRemover/`

adresář s programem,

`DistortionRemover.exe`

Spustitelný soubor aplikace,

`License.txt`

textový soubor s popisem licence,

`Source/`

adresář obsahující zdrojové kódy,

`DistortionRemover.pro`

projektový soubor,

`main.cpp`

zdrojový kód funkce main,

`distortionremover.h`

hlavičkový soubor distortionremover,

`distortionremover.cpp`

zdrojový soubor distortionremover obsahující funkce pro chod programu,

`ui_distortionremover.h`

hlavičkový soubor grafického rozhraní,

`distortionremover.ui`

soubor obsahující grafické prostředí v XML,

`picturetransformation.h`

hlavičkový soubor picturetransformation,

`picturetransformation.cpp`

zdrojový soubor picturetransformation obsahující výpočetní funkce,

`resource.qrc`

soubor pro uložení binárních souborů do aplikace který obsahuje ikonu,

`DR.ico`

ikona programu.