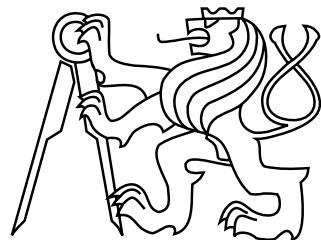


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
OBOR GEOINFORMATIKA



BAKALÁŘSKÁ PRÁCE
ZÁSUVNÝ MODUL QGIS PRO SLUČOVÁNÍ VEKTOROVÝCH DAT

Vedoucí práce: Ing. Martin Landa
Katedra mapování a kartografie

červen 2013

Tereza FIEDLEROVÁ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

studijní program: Geodézie a kartografie

studijní obor: Geoinformatika

akademický rok: 2012/2013

Jméno a příjmení studenta: Tereza Fiedlerová

Zadávající katedra: Katedra mapování a kartografie

Vedoucí bakalářské práce: Ing. Martin Landa

Název bakalářské práce: Zásuvný modul QGIS pro slučování vektorových dat

Název bakalářské práce
v anglickém jazyce: QGIS plugin for vector conflation

Rámcový obsah bakalářské práce: Cílem bakalářské práce je návrh a implementace zásuvného modulu Quantum GIS (QGIS) pro slučování vektorových dat z různých datových zdrojů. Součástí textu práce bude i stručný přehled existujících open source a proprietárních nástrojů v této oblasti. Vybrané algoritmy pro slučování vektorových dat (vector conflation) budou implementovány v rámci navržené C++ knihovny, která bude jedním z výstupů této práce.

Datum zadání bakalářské práce: 7.2.2013

Termín odevzdání: 17. 5. 2013

(vyplňte poslední den výuky
příslušného semestru)

Pokud student neodevzdal bakalářskou práci v určeném termínu, tuto skutečnost předem písemně zdůvodnil a omluva byla děkanem uznána, stanoví děkan studentovi náhradní termín odevzdání bakalářské práce. Pokud se však student rádně neomluvil nebo omluva nebyla děkanem uznána, může si student zapsat bakalářskou práci podruhé. Studentovi, který při opakovém zápisu bakalářskou práci neodevzdal v určeném termínu a tuto skutečnost rádně neomluvil nebo omluva nebyla děkanem uznána, se ukončuje studium podle § 56 zákona o VŠ č. 111/1998. (SZŘ ČVUT čl. 21, odst. 4)

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

vedoucí bakalářské práce

vedoucí katedry

Zadání bakalářské práce převzal dne: 7.2.2013

student

Formulář nutno vyhotovit ve 3 výtiscích – 1x katedra, 1x student, 1x studijní odd. (zašle katedra)

Nejpozději do konce 2. týdne výuky v semestru odešle katedra 1 kopii zadání BP na studijní oddělení a provede zápis údajů týkajících se BP do databáze KOS.

BP zadává katedra nejpozději 1. týden semestru, vlnemž má student BP zapsanou
(Směrnice děkana pro realizaci studijních programů a SZZ na FSV ČVUT čl. 5, odst. 7)

ZDE VLOŽIT ZADÁNÍ

ABSTRAKT

Tato bakalářská práce se zabývá problémem slučování vektorových dat z různých zdrojů. Cílem práce je vytvoření zásuvného modulu pro Quantum GIS. Tento modul umožní uživateli jednoduše kombinovat pomocí zvolené metody dvě různé vektorové vrstvy načtené v prostředí programu. Výsledkem je nová vrstva ve formátu *shapefile*. Modul využívá algoritmy pro spojení datasetů z externí knihovny, jejíž návrh a vytvoření je také náplní této práce. Text práce kromě popisu tvorby modulu a knihovny obsahuje i přehled existujících nástrojů v této oblasti.

KLÍČOVÁ SLOVA

GIS, Quantum GIS, zásuvný modul, C++ knihovna, slučování vektorových map

ABSTRACT

This bachelor thesis deals with the problem of vector to vector conflation. The aim is to create a Quantum GIS plugin. This plugin allows user to simply combine two different vector layers, which are loaded in the program, using selected method. The result is a new layer in *shapefile* format. The plugin uses conflation algorithms from an external library. Design and creation of the library is also the objective of this thesis. Besides the description of the plugin and the library, this text includes a list of existing tools in this field.

KEYWORDS

GIS, Quantum GIS, plugin, C++ library, vector conflation

PROHLÁŠENÍ

Prohlašuji, že bakalářskou práci na téma „Zásuvný modul QGIS pro slučování vektorových dat“ jsem vypracovala samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne

(podpis autora)

PODĚKOVÁNÍ

Chtěla bych poděkovat vedoucímu mé bakalářské práce, Ing. Martinu Landovi, za odborné rady a pomoc při zpracování této práce. Dále bych chtěla poděkovat své rodině a svému příteli za projevenou podporu a trpělivost.

Obsah

1	Úvod	19
2	Conflation	21
2.1	Definice pojmu <i>conflation</i>	21
2.2	Historie	22
2.3	Klasifikace <i>conflation</i>	23
2.3.1	Dle území zobrazovaného vstupními vrstvami	23
2.3.2	Dle typu vstupních vrstev	23
2.4	Obecný postup	24
3	Existující nástroje	27
3.1	Proprietární nástroje	27
3.1.1	ESRI ArcGIS	27
3.1.2	ConflX	28
3.1.3	Intergraph GeoMedia Fusion	29
3.1.4	MapMerger	29
3.2	Open Source nástroje	29
3.2.1	JCS - Java Conflation Suite	29
3.2.2	OpenStreetMap	34
3.2.3	Univerzitní projekty	36
4	Použité technologie	37
4.1	Quantum GIS	37
4.2	GEOS - Geometry Engine, Open Source	37
4.3	Qt projekt	38
5	Knihovna GEOC	41
5.1	Reprezentace geometrie	41
5.2	Vertex Snapper	41
5.2.1	Popis algoritmu	41
5.2.2	Implementace	42
5.2.3	Využití	43
5.3	Coverage Alignment	43
5.3.1	Popis algoritmu	43
5.3.2	Implementace	46

5.3.3	Využití	46
5.4	LineMatcher	46
5.4.1	Popis algoritmu	47
5.4.2	Implementace	48
5.4.3	Využití	48
6	Zásuvný modul pro QGIS	49
6.1	Tvorba QGIS zásuvného modulu	49
6.2	Zásuvný modul Conflate	50
7	Problémy a jejich řešení	53
7.1	Použití knihovny GEOS	53
7.2	Rychlosť zpracování	53
7.2.1	Porovnání rychlosťi s a bez použití prostorových indexů	54
7.3	Vertex Snapper	55
7.4	Line Matcher	56
7.5	Matching Geometry	56
7.6	Coverage Alignment	57
7.7	Automatická oprava nevalidních geometrií	59
8	Další vývoj	61
8.1	Knihovna GEOC	61
8.2	Zásuvný modul Conflate	62
9	Závěr	63
Seznam zkratek		65
Literatura		67
Seznam příloh		71
A Porovnání rychlosťi algoritmů		73
B UML diagramy tříd		77
C Uživatelská příručka		81
C.1	Načtení zásuvného modulu	81
C.2	Spuštění a nastavení dialogu	81
C.2.1	Výběr vstupních vrstev	81

C.2.2	Metoda zpracování	81
C.2.3	Další nastavení	82
C.2.4	Výstupní vrstva	83
C.3	Výsledek	83
D	Screenshoty zásuvného modulu	85
E	Ukázky zpracování dat	87
E.1	VertexSnapper	87
E.2	CoverageAlignment	88
E.3	LineMatcher	90
F	Obsah CD	93

Seznam obrázků

2.1	Klasifikace <i>conflation</i> dle vstupních vrstev	24
3.1	Integrate - princip	28
3.2	Architektura JCS	30
3.3	JOSM conflation	35
3.4	MBPConflate	36
4.1	QGIS logo	37
4.2	Qt logo	38
5.1	Postup přichycení vrcholů	42
5.2	Postup zarovnání vrstev	44
5.3	Porovnání prvků na základě jejich obalových zón	45
5.4	Porovnání prvků na základě obalových zón jejich hranic	45
6.1	Architektura zásuvného modulu <i>Conflate</i>	50
7.1	Vznik nevalidní geometrie při přichycení vrcholů	55
7.2	Příklad situace, kdy vzniká „zub“	56
7.3	Vznik neplatné trojúhelníkové sítě	58
7.4	Oprava křížících se segmentů	59
7.5	Oprava slepých větví	59
B.1	UML diagram tříd pro VertexSnapper	77
B.2	UML diagram tříd pro CoverageAlignment	78
B.3	UML diagram tříd pro LineMatcher	79
C.1	Ikonka zásuvného modulu	81
C.2	Princip jednotlivých algoritmů	82
C.3	Formát protokolu	84
D.1	Dialog zásuvného modulu - nastavení	85
D.2	Dialog zásuvného modulu - nápověda	85
D.3	Dialog zásuvného modulu - upozornění	86
D.4	Dialog zásuvného modulu - protokol	86
E.1	VertexSnapper - vstupní data	87
E.2	VertexSnapper - výsledek zpracování	87
E.3	VertexSnapper - porovnání	88
E.4	CoverageAlignment - vstupní data	88
E.5	CoverageAlignment - před zarovnáním	89
E.6	CoverageAlignment - po zarovnání	89
E.7	CoverageAlignment - porovnání	90

E.8	LineMatcher - vstupní data	90
E.9	LineMatcher - výsledek	91
E.10	LineMatcher - detail	91

Seznam tabulek

7.1	Výsledky testování	54
A.1	Parametry počítače využitého pro testování	73
A.2	Vstupní vrstvy	73
A.3	VertexSnapper - čas zpracování bez prostorových indexů	74
A.4	VertexSnapper - čas zpracování s prostorovými indexy	74
A.5	CoverageAlignment - čas zpracování bez prostorových indexů	75
A.6	CoverageAlignment - čas zpracování s prostorovými indexy	75

1 Úvod

Spojování geografických dat z různých zdrojů je jedním z aktuálních problémů v oblasti geografických informačních systémů (GIS). S rostoucí dostupností digitálních dat začala vzrůstat i potřeba tato data kombinovat a porovnávat. Na internetu je volně k dispozici spousta různých prostorových dat. Jejich přesnost, úplnost, atributy a další vlastnosti jsou však velmi odlišné. Některé jsou geometricky přesné, ale neobsahují potřebné atributy nebo některé žádoucí detaily. Méně přesné mapy s potřebnými informacemi lze upravit pomocí těch přesnějších, abychom získali požadovaný výsledek. Takové a další podobné úlohy jsou předmětem slučování map.

Pod anglickým pojmem *conflation* se skrývá mnoho různých procesů, úloh a činností. Jejich cílem může být obohacení jedné mapy o atributy či prvky z mapy druhé, aktualizace datasetu, kombinace dvou map o stejně přesnosti, detekce změn a jiné. Záměrem je vždy vytvoření nové mapy, z níž lze vyčíst informace, které by nebylo možné získat pouze z jednoho vstupního zdroje.

Vzhledem k velkému množství datových sad a jejich rozsahu je automatizace výše uvedených úloh logickým a nevyhnutelným krokem. Ne vždy je však automatické zpracování zárukou úspěchu. Každý algoritmus má své limity a nikdy nelze identifikovat sto procent odpovídajících si prvků. To může být ještě ztěženo nejednoznačností a nízkou kvalitou vstupních dat. Proto je stále třeba dodatečná ruční úprava a kontrola dat. Ta je však oproti úspoře času díky automatickému zpracování často zanedbatelná.

Výstupem práce je zásuvný modul, který umožnuje spojení vektorových map. Hlavním cílem je však vyhledání způsobů řešení tohoto problému a implementace některých z nich. V rámci práce vzniká také knihovna s algoritmy pro sjednocení vektorových dat, kterou bude možné dále rozšiřovat a doplňovat další funkcionality. Zásuvný modul využívá tuto knihovnu a s jejím rozšiřováním bude možné i do něj přidávat další funkce.

V následující kapitole je podrobněji popsána problematika slučování nejen vektorových map (*conflation*). V části 3 pak jsou představeny některé existující nástroje v této oblasti. Po krátkém výčtu použitých technologií (kapitola 4) už se práce zabývá samotnou tvorbou knihovny a zásuvného modulu. Kapitola 5 popisuje jednotlivé algoritmy nově vznikající knihovny a jejich implementaci. Následuje popis zásuvného modulu a jeho funkcionality (kapitola 6). Poslední dvě kapitoly jsou věnovány řešení problémů a nápadům na další vývoj a vylepšení vzniklé aplikace.

2 Conflation

Cílem této kapitoly je přiblížit čtenáři pojem sloučení vektorových map (*conflation*). Kromě definice samotného pojmu kapitola popisuje i klasifikaci sjednocování map a obecný postup při této úloze. Hned zpočátku bych však ráda upozornila na to, že v angličtině používaný pojem *conflation* nemá žádný oficiální český ekvivalent, proto je v některých případech uveden v originálním znění, aby nemohlo dojít k špatnému výkladu textu.

2.1 Definice pojmu *conflation*

Pojem *conflation* byl poprvé v souvislosti s kartografií a GIS použit manželkou Jamese Corbetta, matematikou působícího v US Census Bureau [8]. Termín pochází z latinského *con flare*, což znamená „rozdmýchávat“ nebo doslovněji „foukat dohromady“. Ještě před tím, než se toto slovo objevilo v digitální kartografii, používalo se pro popis spojování dvou rukopisů do třetí verze, která je kombinací těchto předchozích.

Conflation lze do češtiny přeložit jako slučování či spojování map. Překlad tohoto pojmu však není vždy jasný, jelikož se jím označuje více různých úloh, procesů či činností. Někdy bývá tento výraz zaměňován s anglickými termíny *map matching*, *map merging*, *rubber sheeting* atd., které obecně označují činnosti spadající do této oblasti. Jelikož žádná oficiální česká definice tohoto pojmu neexistuje, jsou zde uvedeny překlady definic z různých zahraničních zdrojů.

- Sada činností, které vzájemně zarovnávají prvky dvou geografických datových vrstev a následně převádí atributy jedné vrstvy do druhé.

(Wiki.GIS.com - The GIS Encyclopedia Glossary, http://wiki.gis.com/wiki/index.php/GIS_Glossary/C, citováno 30. 4. 2013)

- *Feature conflation* je proces kombinování geografických informací z překrývajících se zdrojových dat, který zachovává přesnost dat, minimalizuje nadbytečná data a předchází konfliktům v datech. Potřeba *conflation* vyvstává s nutností aktualizace dat kvůli přesnosti či chybějícím prvkům/atributům pomocí novějších datových zdrojů zahrnujících překrývající se území. *Conflation of geospatial data* (geoprostorových dat) je spojení či sladění dvou různých prostorových datasetů zahrnujících stejně území.

(Shekhar, Xiong: Encyclopedia of GIS [14, s. 129])

- Proces sjednocení dvou rozdílných datasetů.

(Blasby, Davis, Kim, Ramsey: GIS Conflation Using Open Source Tools [1, s. 2])

- *Conflation* je proces, jehož cílem je geometricky upravit (posunem, transformací) digitální data zobrazující stejné území (obvykle pořízená v jiném čase) tak, aby byla vzájemně geograficky „korektní“ a vzájemně se překrývala.

(HUIC Data Services, <http://huic.com/HUIC/job/right/conflation.htm>, citováno 30. 4. 2013)

- Schopnost kombinovat data z různých zdrojů do jednoho společného datasetu je jedním ze základních problémů v GIS. Tato úloha je ve vědecké literatuře označována jako *conflation* prostorových dat.

(Freitas, Afonso: Distributed Vector Based Spatial Data Conflation Services [5, s. 23])

- Soubor funkcí a procedur, které zarovnávají prvky jednoho GIS souboru k prvkům jiného a následně převádí atributy mezi těmito vrstvami. Zarovnání předchází převodu atributů a je nejčastěji prováděno *rubber-sheeting* operacemi.

(Prince George's County - GIS Glossary, <http://www.princegeorgescountymd.gov/Government/AgencyIndex/OITC/GIS/glossary.asp#C>, citováno 30. 4. 2013)

- Proces sladění poloh odpovídajících si prvků v různých datových vrstvách. Funkce pro spojení map (*conflation*) provádějí toto sladění tak, aby se odpovídající si prvky přesně překrývaly.

(Geographic Information Nova Scotia - Standards Manual, <http://www.gov.ns.ca/snsmr/land/standards/post/manual/appedxa1.asp#C>, citováno 30. 4. 2013)

2.2 Historie

Až do 80. let 20. století bylo pořízení digitálních dat velmi drahé, a proto se často nestávalo, že by nějaká firma vlastnila více mapových souborů jediného území. S vývojem počítačů a digitálních technologií se však tato data stávala stále dostupnějšími a náhle vyvstala potřeba kombinovat data o jednom území z více zdrojů a provádět aktualizace těchto dat.

Jako první se k takovému množství dat dostaly pochopitelně různé vládní instituce [13]. Ačkoli první článek o problematice spojování geografických dat z více zdrojů vyšel už v roce 1981 (M. White: *The Theory of Geographical Data Conflation*), k opravdovému rozvoji došlo až po roce 1985. V té době totiž vznikl projekt, jehož cílem bylo propojení mapových souborů organizací US Census Bureau a US Geological Society. Vzhledem k množství dat bylo nutné celý proces co nejvíce automatizovat. Použitý algoritmus, jehož hlavním autorem je Alan Saalfeld, byl založen na nalezení odpovídajících si prvků a následné transformaci dat.

S narůstající dostupností dat a zveřejněním této myšlenky přibývalo i menších firem zabývajících se problémem kombinace mapových souborů z různých zdrojů.

Zatímco před dvaceti lety se datasety slučovaly pouze na základě geometrických a topologických vlastností [13], dnes stále větší roli hrají i atributy jednotlivých prvků [3], které mohou práci velmi usnadnit. Nejnovějším přístupem je pak zjišťování sémantické podobnosti, která je založena na porovnávání atributů, datových struktur a vzájemných vztahů [11].

2.3 Klasifikace *conflation*

2.3.1 Dle území zobrazovaného vstupními vrstvami

1. Horizontální

Za *horizontal conflation* se označují procesy, které zpracovávají data ze vzájemně sousedících území. Cílem je získat mapové soubory, jejichž hranice na sebe dokonale navazují, a to pokud možno bez ztráty přesnosti.

2. Vertikální

Při *vertical conflation* obsahují vstupní data překrývající se území. Jde tedy o dva či více souborů zobrazujících jediné území nebo alespoň jeho část. Může se jednat o dvě verze té samé mapy nebo o dva datasety s nějakými společnými prvky a vlastnostmi. Výsledkem celého procesu je jediný dataset, jehož přesnost není horší než přesnost původních dat a obsahuje informace z obou zdrojů. Jde tedy o vylepšenou, obsahově bohatší mapu s odpovídající přesností.

2.3.2 Dle typu vstupních vrstev

1. Imagery to imagery, Raster to raster

Jedná se o případ, kdy je úkolem na sebe napasovat dva rastrové mapové soubory. Nejčastěji jde o ortofoto a naskenovanou analogovou mapu daného území. Pomocí této úlohy můžeme například porovnat starou analogovou mapu se současným stavem reprezentovaným právě leteckým snímkem. Řešení tohoto problému vyžaduje poměrně složité techniky pro nalezení odpovídajících si objektů. Velmi důležitým faktorem je kvalita a rozlišení vstupních dat.

2. Vector to imagery, vector to raster

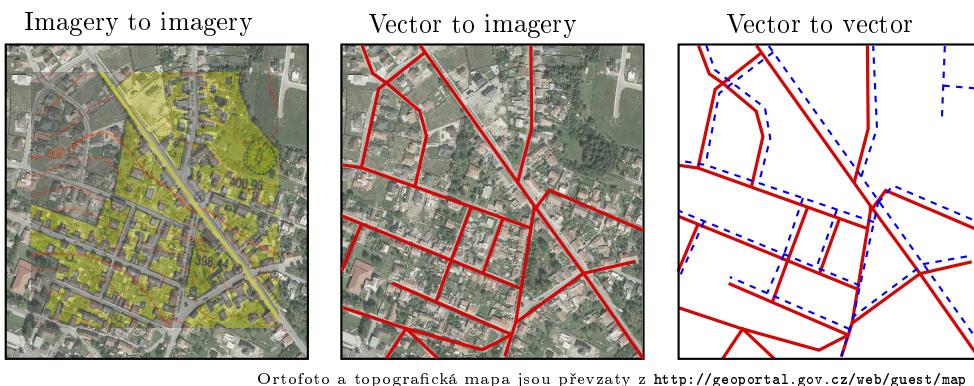
Kombinace vektorových a rastrových dat stejného území se využívá často ke zpřesnění vektorových dat jejich napasováním na ortofoto. Hlavní náplní této oblasti je vývoj algoritmů umožňujících následující [14]:

- (a) Detekce charakteristických hran rastrového obrazu a jejich porovnání s vektorovými daty.
- (b) Využití vektorových dat k identifikaci hran v rastru - tzv. *Snakes algorithm*.
- (c) Užití stereo obrazu, výškových dat a dalších znalostí pro porovnání vektorových a rastrových dat.
- (d) Využití prostorových informací a jiných vlastností mapy (jako např. barev) k rozpoznání odpovídajících si prvků.

3. Vector to vector

Případem sloučení dvou vektorových datasetů se zabývá tato práce. Jednou ze specifických a velmi častých aplikací je navázání silničních sítí, dále aktualizace digitálních map aj. Existuje mnoho různých algoritmů pro slučování vektorových dat, přičemž základní přístupy k řešení problému jsou následující [14]:

- (a) Sloučení dat za pomocí algoritmů, které pracují na základě porovnávání geometrických vlastností prvků.
- (b) Algoritmy, které berou v potaz podobnost tvarů prvků a zároveň podobnost jejich atributů.
- (c) Spojení vektorových dat s neznámým souřadnicovým systémem na základě rozmístění významných bodových prvků (např. křižovatky cest).



Obrázek 2.1: Klasifikace *conflation* dle vstupních vrstev

2.4 Obecný postup

Obecný postup při slučování vektorových map z více zdrojů se skládá z několika kroků, které jsou uvedeny níže [1, 17]. Jako u většiny podobných operací je třeba nejdříve provést přípravu dat a následně data zpracovat a upravit.

1. Předzpracování dat

Tento krok slouží k zajištění kompatibility vstupních dat tak, aby bylo možné je porovnávat. Obvykle spočívá v převedení vstupních datasetů do stejného souřadnicového systému, zajištění stejných základních jednotek a dalších menších úpravách.

2. Kontrola kvality dat a topologické správnosti vrstev

Zde se kontroluje vnitřní konzistence dat každé vstupní vrstvy. V tomto případě záleží především na požadavcích zvoleného algoritmu pro sloučení map. Může jít například o odstranění topologických chyb v dané vrstvě jako jsou nežádoucí drobné překryty či mezery mezi polygony.

3. Vyhledání odpovídajících si prvků

Následuje vyhledání prvků, které si v upravovaných datasetech odpovídají, tedy zobrazují stejný předmět ve skutečnosti. Tento krok je nezbytný proto, aby bylo možné rozhodnout, jak na sebe vstupní vrstvy navazují.

4. Sloučení geometrických prvků a/nebo atributů

Po rozpoznání odpovídajících si prvků je možné upravit geometrii¹ či atributy prvku z jedné vrstvy s přihlédnutím k vlastnostem prvku z druhé vrstvy. Při procesu slučování různých datasetů lze převádět atributové hodnoty mezi odpovídajícími si prvky nebo změnit geometrii prvku tak, aby odpovídala geometrii prvku z jiné vrstvy, která bývá označena za referenční. Ve složitějších úlohách už je možné převádět zároveň atributy i geometrii, a to nejen jednosměrně (tedy z vrstvy referenční do vrstvy upravované), ale výsledkem může být vrstva, jejíž geometrické vlastnosti vychází z kombinace obou vstupních datasetů.

5. Závěrečné úpravy

Po provedení automatického a někdy i manuálního sloučení datových vrstev je vhodné provést kontrolu výsledku. Často jsou potřeba ještě drobné úpravy, aby výsledná data odpovídala počátečním požadavkům. Ne vždy jsou totiž při automatickém procesu určeny správně všechny odpovídající si prvky a některé algoritmy mohou při úpravě geometrických vlastností narušit topologickou správnost dat.

¹Geometrií je v tomto případě myšlen tvar a poloha geometrického útvaru.

3 Existující nástroje

Tato kapitola se zabývá již existujícími nástroji pro spojování vektorových map z různých zdrojů (*conflation*).

3.1 Proprietární nástroje

Proprietárních nástrojů umožňujících řešení spojování vektorových map existuje celá řada. Některé programy jsou orientovány přímo na tento problém, jiné nabízejí funkce pro slučování map pouze jako vedlejší funkcionalitu a ne vždy je proto možné pomocí nich řešit složitější problémy. Následující výčet neobsahuje všechn komerční software zabývající se slučováním vektorových map, ale nejvýznamnější nástroje, které lze pro zpracování použít. Vzhledem k tomu, že se jedná o proprietární software, nebylo možné všechny níže popsané nástroje otestovat, proto jejich popis vychází především z informací dostupných na oficiálních internetových stránkách produktů.

3.1.1 ESRI ArcGIS

V softwaru ArcGIS 10² existuje několik nástrojů, které lze využít pro spojování geografických dat, přenos atributů a odstraňování geometrických rozdílů mezi datasety [19].

Spatial Adjustment

Soubor nástrojů *Spatial Adjustment* systému ArcGIS zahrnuje základní funkce týkající se této problematiky. Má sloužit především k úpravě dat z různých zdrojů a zajistit tak jejich celistvost. Poskytuje několik metod pro zarovnání jedné vrstvy či její části ke druhé. Lze provést transformaci, navázání hran (*edge matching*) nebo srovnání překrývajících se dat (*rubber sheeting*).

Pro použití metody transformace je nejdříve třeba označit data, která budou do procesu vstupovat. Poté se vyberou dvojice uzlových bodů, které by si měly odpovídat. Na výběr je několik typů transformací pro úpravu dat na základě vybraných dvojic identických bodů.

Do procesu horizontálního zarovnání (*horizontal conflation*) lze zařadit funkce nástroje *Edge Match Tool*, který umožňuje navázat na sebe dvě sousedící vrstvy. Zarovnání je poměrně snadné, stačí pouze zvolit toleranci (maximální vzdálenost pro navázání prvků) a označit tímto nástrojem hranici mezi vrstvami, kde by měly na sebe navazovat. Ve vybrané oblasti se zobrazí indikátory naznačující způsob zarovnání, které lze ještě ručně upravit. Po potvrzení se provede zarovnání tak, aby byla zachována topologie.

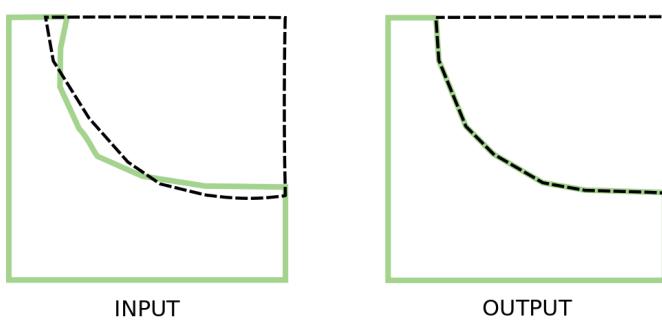
Dalším způsobem geometrického spojení vrstev, který *Spatial Adjustment* nabízí, je *Rubber Sheet*. V tomto případě je opět třeba označit odpovídající si body a navíc body,

²<http://www.esri.com/software/arcgis>

jejichž poloha by se neměla změnit. Při spuštění zarovnání se dočasně vytvoří triangulační síť mezi označenými body. Poloha ostatních neoznačených bodů je pak vypočtena interpolací v trojúhelnících sítě.

Poslední z nástrojů pro kombinaci map je *Attribute Transfer*. Pomocí něho je možné převést atributy mezi odpovídajícími si prvky, ale také upravit jejich geometrii. Po volbě jednoho či více atributů, které se mají převádět mezi zdrojovou a cílovou vrstvou lze interaktivně vybírat odpovídající si dvojice prvků v obou vrstvách. Mezi těmito prvky se převedou atributy a pokud je zaškrtnuta volba *Transfer Geometry* neboli „převést geometrii“, změní se geometrie cílového prvku dle zdrojového. Převod lze provést také na jednou pro více vybraných prvků.

Integrate



Obrázek 3.1: Integrate - princip (zdroj: <http://resources.arcgis.com/en/help/main/10.1/index.html#/00170000002s000000>)

Nástroj *Integrate* umožnuje sladění dvou datasetů. Na vstupu vyžaduje dvě či více vrstev, které chceme zarovnat, a dále maximální vzdálenost, při které lze považovat prvky za odpovídající si. U každého datasetu navíc lze zadat prioritu (*rank*). Prvky s nižší prioritou se pak budou zarovnávat k těm s prioritou vyšší. Tato funkce je velmi užitečná, pokud máme dvě překrývající se nebo sousedící vrstvy s malými rozdíly např. v důsledku různé přesnosti.

3.1.2 ConflleX

ConflleX³ je software pro automatické spojování vektorových GIS dat, který pro automatizaci využívá umělou inteligenci. ConflleX umožňuje zpracování i takových případů, kdy se zdrojová mapa s cílovou nepřekrývají nebo nejsou topologicky identické. Systém porovnává každé dva segmenty z obou map a jejich vztah k ostatním segmentům. Na základě tohoto postupu pak rozhodne, zda se jedná o stejné prvky či nikoli. Kromě automatického procesu umožňuje program i následnou ruční editaci.

³<http://www.citygategis.com/conflex.htm>

ConflteX je k dispozici jako samostatná aplikace ale také jako extenze programu ArcGIS Desktop 9/10.

3.1.3 Intergraph GeoMedia Fusion

Jedná se o doplněk desktopové aplikace GeoMedia⁴ firmy Intergraph, který je dostupný od roku 2005. Nástroj GeoMedia Fusion je navržen pro topologické opravy dat, validaci atributů a integraci dat. Cílem je umožnit snadnou údržbu dat v rozsáhlých geografických databázích, kde jsou data získávána z různých zdrojů. Nástroj porovnává dva datasety obsahující rozdílné reprezentace stejné skutečnosti. Nejdříve automaticky vytvoří *conflation links* indikující způsob spojení vrstev, které lze ještě ručně editovat. Následně geometrie i atributy těchto dvou reprezentací sjednotí. GeoMedia Fusion slouží k úpravě bodových, liniových i plošných dat včetně jejich atributů.

3.1.4 MapMerger

MapMerger⁵ je GIS nástroj firmy ESEA zaměřený na slučování geometrie a atributů vektorových map a kontrolu kvality dat. Umožnuje převod atributů mezi dvěma překrývajícími se mapami, navázání hranic dvou sousedících map, přidání prvků z jedné mapy do druhé, synchronizaci mapy s její aktualizovanou verzí, identifikaci geometrických a atributových rozdílů mezi dvěma verzemi té samé mapy. První verze programu vyšla již v roce 1998, od té doby již firma vyvinula devět verzí. Z uvedených nástrojů poskytuje asi nejvíce možností zpracování a díky specializaci přímo na spojování map se řadí mezi nejvýkonnější programy v této oblasti.

3.2 Open Source nástroje

V této sféře zatím neexistuje mnoho nástrojů, které by komplexně řešily problém spojování vektorových map (*conflation*). Většinou jde pouze o malé programy či zásuvné moduly k větším projektům, pomocí nichž se dá provést manuální nebo poloautomatické sloučení vektorových datasetů, případně jejich atributů. Ovšem většinou je cesta k dosažení cíle pomocí těchto nástrojů poměrně složitá a ne vždy jsou výsledky takové, jak bychom si představovali. Asi jediným ucelenějším nástrojem je knihovna Java Conflation Suite (JCS) implementovaná jako kolekce zásuvných modulů v programu OpenJUMP.

3.2.1 JCS - Java Conflation Suite

JCS⁶ je *open source* knihovna napsaná v jazyce Java, která zahrnuje API a soubor interaktivních nástrojů, které slouží ke slučování prostorových datasetů. Byla vyvinuta společností

⁴<http://geospatial.intergraph.com/products/GeoMedia/Details.aspx>

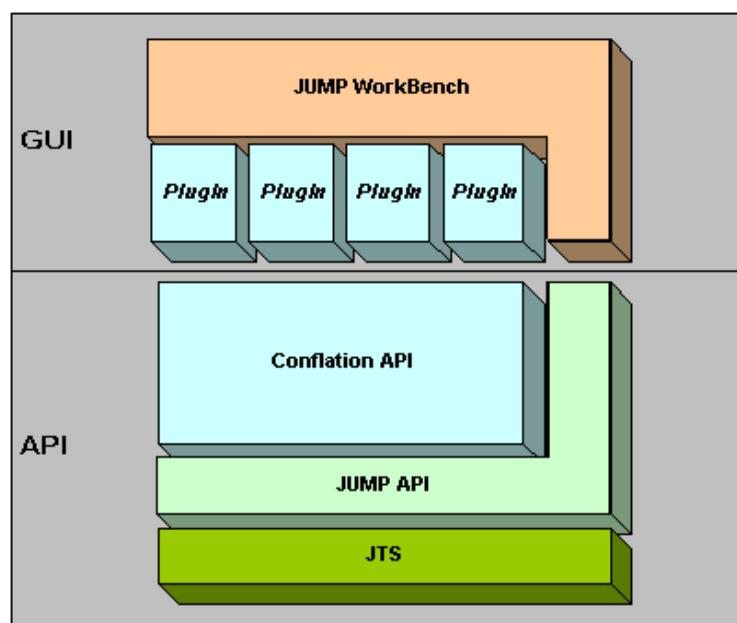
⁵<http://www.mapmerger.com>

⁶<http://www.vividsolutions.com/jcs/>

Vivid Solutions, Inc. Obsahuje funkce umožňující provádění různých procesů spojených se spojováním vektorových map, které jsou zaměřeny především na polygonové, případně liniové datasety. Co se týče bodových vrstev, je její funkcionality poměrně omezená. Knihovna JCS je závislá na knihovně Java Topology Suite (JTS)⁷, která poskytuje základní geometrické funkce pro práci s prostorovými daty. Obě knihovny jsou navrženy v souladu s OGC specifikací *Simple Features* pro SQL, která popisuje způsob uložení geografických digitálních dat, prostorové vztahy a funkce. Knihovna JCS vznikla v rámci projektu JUMP Unified Mapping Platform. Je poskytována pod licencí GNU Lesser General Public License (LGPL)⁸.

Architektura

Knihovna JCS používá pro vizualizaci dat a interakci JUMP WorkBench a API. Poskytování základní geometrické funkcionality je pak zajištěno knihovnou JTS. JUMP API umožňuje vstup a výstup prostorových dat a další funkcionality s nimi spojenou. Jádro JCS tvoří Conflation API obsahující algoritmy pro kontrolu a slučování prostorových dat (*conflation*). Funkce JCS jsou v projektu OpenJUMP implementovány v podobě kolekce zásuvných modulů - *QA*, *Conflate*, *RoadMatcher*.



Obrázek 3.2: Architektura JCS (zdroj: <http://www.vividsolutions.com/JCS/images/>)

⁷<http://www.vividsolutions.com/jts/>

⁸<http://www.gnu.org/licenses/lgpl.html>

OpenJUMP projekt

OpenJUMP je *open source* projekt, který vyvinula firma Vivid Solutions, Inc. Jedná se o GIS software, který umožňuje základní práci s prostorovými daty v rastrové či vektorové podobě a jejich atributy.

JTS - Java Topology Suite

Knihovna JTS je jedním ze základních prvků projektu OpenJUMP. Je napsána v jazyce Java a poskytována pod licencí LGPL. Obsahuje třídy pro reprezentaci geometrických objektů a základní funkce pro práci s prostorovými daty dle specifikace *Simple Features* pro SQL od OGC. Kromě tříd reprezentujících geometrické prvky dle zmíněné specifikace zahrnuje další podpůrné třídy pro reprezentaci seznamu souřadnic, aplikaci geometrického filtru (např. při transformaci), uchovávání informace o maximální a minimální souřadnici objektu a jiné. Dále jsou součástí této knihovny třídy umožňující geometrické výpočty jako je vzájemná poloha bodu a linie, výpočet průsečíku, prostorové analýzy, test polohy bodu a uzavřené oblasti atd.

Funkcionalita JCS

Projekt JCS neumožňuje změnu souřadnicového systému. Proto se automaticky přepokládá, že vrstvy vstupující do zpracování mají stejný prostorový referenční systém. Souřadnice bodů výstupních vrstev mají vždy desetinnou přesnost.

Při většině výpočtů v zásuvných modulech JCS je využívána tzv. Hausdorffova metrika. Na rozdíl od euklidovské metriky totiž nezkoumá jen nejkratší vzdálenost mezi prvky, ale i vzdálenost největší. Zohledňuje tedy do jisté míry i topologické vztahy.

Na výpočet je Hausdorffova vzdálenost poměrně složitá. Proto se v algoritmech použitých v JCS používá spíše vrcholová Hausdorffova vzdálenost (*Vertex Hausdorff Distance*), která není vztažena ke geometrickému prvku, ale pouze k jeho vrcholům. Tato varianta Hausdorffovy vzdálenosti ve většině případů vrací stejně dobré výsledky.

Postup spojování vektorových datasetů pomocí knihovny JCS je založen na nalezení geometrických rozdílů mezi oběma mapami a následném odstranění těchto rozdílů.

K detekci geometrických odlišností je použit algoritmus pro prostorové rozdíly. Tento algoritmus funguje tak, že postupně porovnává geometrické prvky z obou datasetů, případně pouze jejich jednotlivé části, a pokud se tyto prvky shodují, označí je jako odpovídající si. Výsledkem jsou pak ty prvky z obou datasetů, ke kterým nebyly nalezeny žádné odpovídající prvky. Nalezení odpovídajících si prvků probíhá následovně. Pokud je požadována přesná shoda, provede se testování, zda jsou prvky stejného geometrického typu

a zda se rovná jejich obsah. Porovnání obsahu se zakládá na porovnávání jednotlivých komponent a seznamu bodů daných geometrií. Nevždy je předpokládána přesná shoda, proto je možné určit i prvky, které splňují podmínu danou tolerancí. Zde se provádí porovnání Hausdorffovy vzdálenosti mezi prvky s touto tolerancí, nepočítá se přitom přímo tato vzdálenost. Rozhodující je, jestliže obalová zóna o velikosti vzdálenostní tolerance prvního prvku obsahuje prvek druhý a naopak. Tato podmínka je ekvivalentní k podmínce, že Hausdorffova vzdálenost musí být menší nebo rovna toleranci.

Pro odstraňování překryvů či mezer neexistuje exaktní algoritmus, ale jsou využívány různé heuristiky poskytující dobré topologické výsledky.

Následující výčet podává přehled nejčastějších úloh, které je možné s tímto nástrojem řešit.

- Jako *Coverage Cleaning* je označován proces hledání a odstraňování topologických chyb - mezer a překryvů v rámci jedné vektorové mapy tvořené polygony či multipolygony. Detekce nežádoucích mezer mezi polygony je založena na rozpoznání blízkých liniových segmentů, kde blízkost je určena zvolenou vzdálenostní tolerancí.
- Další často řešenou úlohou je *Boundary Alignment*, což by bylo možné přeložit do češtiny jako „zarovnání či navázání hranic“. Cílem je napojit k sobě dvě vektorové vrstvy, které zobrazují sousedící území, tak, aby mezi nimi nevyskytovaly nežádoucí mezery či překryvy. Výsledkem jsou tedy plynule navazující datasety, které tvoří bezešvou mapu. Při této úloze je nutné zvolit přesnější referenční vrstvu, jejíž geometrické vlastnosti se nezmění.
- U *Coverage Alignment* jsou na vstupu dvě vektorové mapy zobrazující to samé území nebo alespoň jeho část. Tyto mapy se tedy výrazně překrývají. Úkolem je upravit méně přesnou vrstvu tak, aby odpovídala vrstvě referenční, popřípadě její části, pokud se vrstvy úplně nepřekrývají. Proces spočívá v posunutí vrcholů polygonů upravované vrstvy do blízkých vrcholů vrstvy referenční.
- Poměrně specifickou, ale velmi častou úlohou je *Road Network Matching* neboli „spojovalní silničních sítí“. Na vstupu jsou dvě vektorové mapy té samé silniční sítě. Při této úloze se hledá podobnost mezi liniovými prvky obou datasetů, které jsou označeny za odpovídající si. Poté je vytvořena nová vrstva silniční sítě, která obsahuje odpovídající si prvky, přičemž při mírných odlišnostech použije liniové prvky z přesnější mapy. JCS bohužel zatím neumožňuje automatické provedení této úlohy. Pouze nalezne odpovídající si prvky a další kroky už je nutné provést manuálně.

- Úloha *Geometry Difference Detection*, v češtině „detekce geometrických rozdílů“, na rozdíl od předchozích nijak neupravuje vstupní vrstvy ani z nich netvoří jiné. Cílem je pouze nalézt rozdíly mezi vstupními datasety. Nejčastěji je používána pro rozpoznání změn mezi dvěma verzemi jedné vektorové mapy (např. po aktualizaci).

Popis zásuvných modulů

Zásuvné moduly ze skupiny *QA - Quality Assurance* umožňují najít geometrické rozdíly a nesrovnalosti mezi datasety, ale také v rámci jediného datasetu. Funkce zde obsažené neslouží k opravě či propojení vrstev, ale pouze k identifikaci geometrických rozdílů.

- *Find Misaligned Segments* - slouží k nalezení segmentů ze dvou datasetů, které by si v rámci dané tolerance měly odpovídat, ale je mezi nimi mezera či překryv.
- *Find Overlaps* - najde překrývající se prvky ze dvou datasetů.
- *Find Coverage Gaps* - umožňuje nalézt mezery mezi polygony jednoho datasetu, které jsou užší než zadaná vzdálenostní tolerance. Zároveň musí být mezi hranami polygonů, které tvoří tuto mezenu, úhel menší než daná úhlová tolerance.
- *Find Coverage Overlaps* - najde všechny překryvy mezi polygony v rámci jednoho datasetu, respektive všechny překrývající se polygony.
- *Find Close Vertices* - identifikuje body (samostatné body, vrcholy linií či polygonů) ze dvou různých datasetů, jejichž vzdálenost je menší než daná tolerance.
- *Find Offset Boundary Corners* - slouží k nalezení hranic polygonů ze dvou sousedících vektorových map, které by na sebe měly navazovat, ale je mezi nimi posun menší než zadaná tolerance.
- *DiffSegmentsPlugin* - identifikuje liniové segmenty, které jsou obsaženy pouze v jedné ze zadaných vrstev, nikoli v obou dvou.
- *DiffGeometryPlugin* - funguje stejně jako předchozí funkce s tím rozdílem, že hledá i samostatné geometrie (celé linie, polygony), nikoli jen liniové segmenty.

Další zásuvné moduly zařazené do skupiny s názvem *Conflate* slouží k samotnému spojování vektorových map a navázání dvou sousedních map.

- *Vertex Snapper* - identifikuje a napojí k sobě blízké uzlové body, vrcholy ze dvou překrývajících se datasetů. Při použití této funkce je nutné označit, která vrstva je referenční (s body z této vrstvy se nebude hýbat).

- *Coverage Alignment* - zarovná geometrii jednoho datasetu k jinému referenčnímu datasetu v místech, kde se překrývají nebo spolu sousedí. Na rozdíl od předchozí funkce nepracuje pouze s odpovídajícími si body, ale s celými geometriemi.
- *PolygonToolboxMatcherPlugin* - tento nástroj slouží k identifikaci podobných polygonů ve dvou různých datasech, přičemž umožňuje různá nastavení tak, aby bylo možné najít opravdu jen odpovídající si polygony, popřípadě více polygonů odpovídajících jednomu či naopak.
- *AlignmentToolboxPlugin* - slouží k zarovnání dvou vrstev k sobě. Bohužel zatím není plně funkční.

Skupina označena jako *Clean* obsahuje funkce k opravě nepřesností nalezených pomocí funkcí skupiny *QA* v rámci jednoho datasetu.

- *Remove Coverage Gaps* - odstraní mezery mezi polygony jedné vrstvy dle zadané tolerance.
- *Remove Short Segments* - tato funkce by měla dokázat odstranit liniové segmenty kratší než daná tolerance tak, aby co nejméně porušila topologii vrstvy. Zatím však umožňuje pouze odstranění krátkých izolovaných segmentů.
- *CoverageCleaningToolboxPlugin* - poskytuje stejnou funkcionalitu jako první nástroj z této skupiny, navíc umožňuje odhalit překryvy mezi polygony jedné vrstvy.

Poslední skupina zásuvných modulů je nazvana *Roads*. Zabývá se spojováním vektorových map silniční sítě.

- *RoadMatcherToolboxPlugin* - umožňuje vytvořit vrstvu s rozdíly mezi silnicemi ze dvou vrstev. Na základě těchto rozdílů a identifikovaných společných prvků jednu z těchto vrstev naváže na druhou referenční tak, aby si odpovídaly.

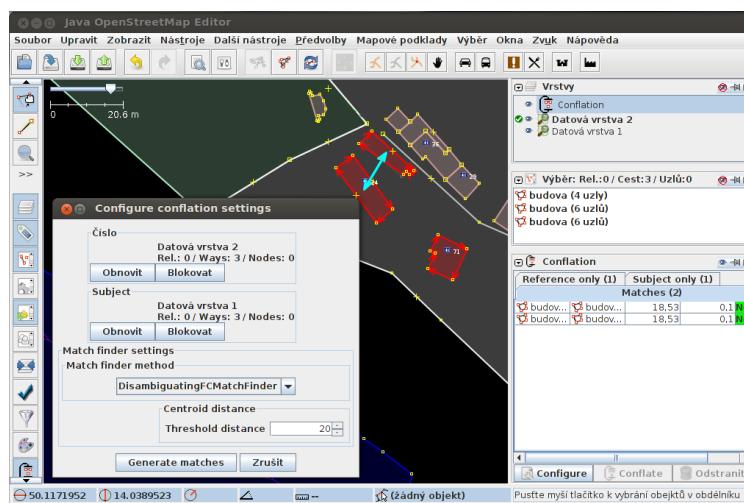
3.2.2 OpenStreetMap

OpenStreetMap je projekt sloužící k tvorbě a vizualizaci geografických dat. Jedná se o *open source* projekt, což znamená, že ho může kdokoli využívat a přispívat do něj. V OpenStreetMap existuje mnoho nástrojů pro editaci prostorových dat a některé z nich umožňují i manuální nebo poloautomatické spojování datasetů z různých zdrojů (*conflation*). Bohužel ani zde však neexistuje žádný komplexnější nástroj jako je výše popsaná JCS. Navíc většina těchto nástrojů je vytvořena pro nějaký konkrétní účel (např. pro spojování silničních sítí v USA) a ne vždy je proto jejich použití zcela obecné. Uživatel si tedy mnohdy

musí poradit sám a použít několik různých nástrojů, aby dosáhl požadovaného výsledku. Dále uvádím nástroje, které lze pro některé činnosti související se spojováním map použít [22].

JOSM conflation plugin

Java OpenStreetMap Editor (JOSM) je desktopová aplikace umožňující editaci dat projektu OpenStreetMap. Jedním ze zásuvných modulů pro tuto aplikaci je *Conflation*, který umožňuje spojování vektorových dat. Tento nástroj však je stále označen jako experimentální, což znamená, že ne vždy funguje zcela správně. Nástroj umožňuje zarovnat prvky jedné vrstvy tak, aby souhlasily s prvky druhé vrstvy, která je označena za referenční.



Obrázek 3.3: JOSM conflation

V prvním kroku je určena referenční a upravovaná vrstva. Poté je třeba v obou těchto vrstvách označit prvky, které by si měly odpovídat. V dalším kroku se provede automatická identifikace dvojic vybraných prvků z obou datasetů. Nakonec se vykoná samotný proces zarovnání prvků (*conflate*), kdy jsou prvky upravované vrstvy změněny tak, aby odpovídaly prvkům vrstvy referenční. Jedná se o proces poloautomatický, jelikož nejdříve je třeba ručně identifikovat odpovídající si prvky a teprve na základě tohoto kroku je provedena automatická úprava prvků.

Potlatch 2 merging tool

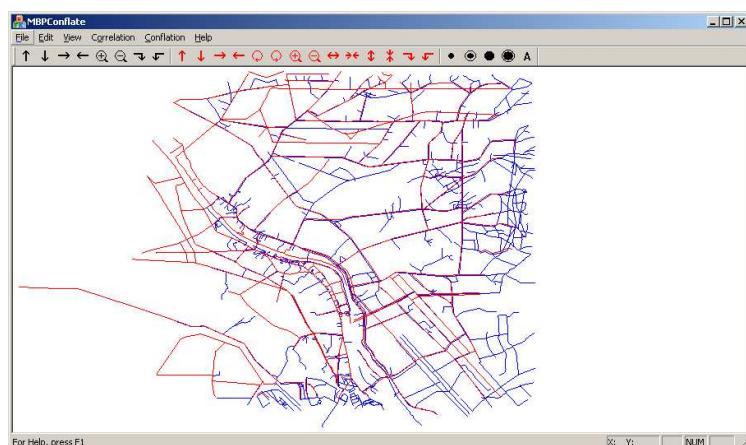
Potlatch 2 merging tool je nástroj původně navržený pro spojování vektorových dat cyklistických tras v rámci England Cycling Data Project⁹. V případě tohoto nástroje se jedná především o proces přenosu atributů. Pokud se v mapě nachází dva odpovídající si prvky,

⁹Projekt pod záštitou britského ministerstva dopravy, který si klade za cíl umožnit dostupnost informací o síti cyklostezek ve Velké Británii prostřednictvím OpenStreetMap.

pomocí tohoto nástroje je lze označit za odpovídající a jednoduše sloučit jejich atributy. Nutno však podotknout, že výběr odpovídajících si prvků se musí provést vždy ručně.

3.2.3 Univerzitní projekty

Mimo výše uvedené nástroje existuje také několik projektů různých světových univerzit zabývajících se buď kombinací geografických dat obecně nebo spojováním map silničních sítí. Možnosti využití těchto programů pro běžného uživatele se velmi různí. A ne vždy lze zcela volně tyto nástroje vyzkoušet. Jedním z nich je i Conflation System MBP, který byl vyvinut na katedře počítačů (*Computer Science Department*) na Central Washington University v USA. V rámci tohoto projektu vznikl *MBPConflate* software, který má přispět k výzkumu v oblasti slučování geografických dat. Program umožňuje automatické spojení map, poskytuje nástroje pro následnou kontrolu kvality výsledné mapy. Je navržen tak, aby bylo snadné implementovat nové techniky a algoritmy v této oblasti. Bohužel k tomuto nástroji nejsou k dispozici dostatečné informace týkající se licence a dostupnosti.



Obrázek 3.4: MBPConflate (zdroj: [2])

4 Použité technologie

Tato kapitola si klade za cíl seznámit čtenáře s technologiemi, které byly při tvorbě zásuvného modulu a externí knihovny využity. Uvedeny jsou pouze základní informace o jednotlivých projektech, více podrobností lze nalézt na uvedených oficiálních internetových stránkách projektů.

4.1 Quantum GIS



Obrázek 4.1: QGIS logo (zdroj: http://www.qgis.org/wiki/File:QGis_Logo.png)

Quantum GIS (QGIS) je *open source* multiplatformní geografický informační nástroj, který patří mezi oficiální projekty OSGeo. QGIS je napsán v jazyce C++, je publikován pod licencí GNU General Public License (GPL)¹⁰. Program slouží ke zpracování a analýze vektorových i rastrových dat. Základní funkcionalitu samotné aplikace rozšiřuje velké množství volně dostupných zásuvných modulů. Tyto moduly jsou psané v jazycích C++ nebo Python.

Vývoj projektu QGIS začal v roce 2002. Původní myšlenka vzešla z potřeby prohlížeče GIS dat pro operační systém GNU/Linux, který by podporoval většinu existujících formátů. To vedlo k vytvoření *Quantum GIS* projektu. Úplně první verze podporovala pouze *PostGIS*¹¹ vrstvy, poměrně rychle však byla doplněna podpora dalších datových formátů a QGIS se z pouhého prohlížecího nástroje stal plnohodnotným geografickým informačním systémem.

4.2 GEOS - Geometry Engine, Open Source

Geometry Engine, Open Source (GEOS)¹² je přepisem knihovny Java Topology Suite (viz kap. 3.2.1) do jazyka C++. Záměrem projektu je poskytovat kompletní funkcionalitu

¹⁰<http://www.gnu.org/licenses/gpl.html>

¹¹Prostorová nadstavba databázového systému *PostgreSQL*.

¹²<http://trac.osgeo.org/geos/>

JTS pro C++. Knihovna byla původně vyvíjena firmou Refractions Research of Victoria, Canada. Nyní je projektem Open Source Geospatial Foundation (OSGeo) a je dostupná pod licencí LGPL.

Knihovna GEOS implementuje třídy pro reprezentaci geometrických objektů (bod, linie, polygon, vícenásobný bod, geometrická kolekce - soubor prvků aj.). Umožňuje určovat vztahy mezi prostorovými objekty (test, zda se dva objekty protínají, dotýkají, překrývají, jeden obsahuje druhý atd.), provádět prostorové operace (výpočet vzdálenosti, plochy polygonu, konvexní obálky, obalové zóny, sjednocení atd.). Obsahuje také třídy, které dokáží číst a převádět Well Known Text (WKT)¹³ a Well Known Binary (WKB)¹⁴ formáty. Stejně jako knihovna JTS se řídí specifikací *Simple Features* pro SQL. Obsahuje C i C++ API.

4.3 Qt projekt



Obrázek 4.2: Qt logo (zdroj: <http://qt.digia.com/About-us/Logos-for-Download/>)

Qt je multiplatformní knihovna pro vývoj aplikací včetně grafického uživatelského rozhraní určená především pro vývojáře pracující v jazyce C++, existuje však i pro některé další jazyky.

Projekt Qt¹⁵ byl vyvinut v roce 1999 norskou společností Trolltech, v roce 2008 ho tato firma prodala firmě Nokia, až nakonec v roce 2012 skončil v rukou společnosti Digia. Přes všechny tyto změny však zůstal jedním z oblíbených nástrojů pro vytváření desktopových a mobilních aplikací s grafickým rozhraním. Qt je licencován pod *open source* licencemi LGPL verze 2.1, GPL verze 3.0, ale existuje i komerční licence pro vývoj proprietárních projektů.

¹³Textový značkovací jazyk pro popis vektorové geometrie geografických objektů apod.

¹⁴Binární forma pro popis vektorové geometrie geografických objektů apod.

¹⁵<http://qt.digia.com/>

V rámci projektu nevznikla pouze knihovna, ale i řada nástrojů usnadňujících vývoj aplikací. Jedním z nich je i program **Qt Creator**. Jedná se o vývojové prostředí (IDE) obsahující mimo jiné i nástroj pro návrh grafického rozhraní (UI designer) a nástroje pro *debugging* (testování, ladění počítačového programu). Tento nástroj byl použit pro tvorbu zásuvného modulu i knihovny, jež jsou výsledkem této práce.

5 Knihovna GEOC

Algoritmy týkající se slučování vektorových map (*conflation*) byly implementovány v externí knihovně GEOC bez závislosti na QGIS API. Zásuvný modul **Conflate** zprostředkovává funkcionality knihovny v prostředí Quantum GIS. Díky tomuto oddělení vznikla nezávislá knihovna, kterou bude možné případně použít i v jiných programech a projektech.

Tato kapitola popisuje jednotlivé algoritmy a jejich implementaci v knihovně GEOC, zároveň také stručně pojednává o možnostech jejich využití. Podrobný popis jednotlivých tříd a jejich metod lze najít v dokumentaci na přiloženém CD (viz příloha F).

5.1 Reprezentace geometrie

Pro reprezentaci geometrie a prostorové analýzy GEOC využívá knihovnu GEOS. Geometrie prvků je představována třídou `geos::geom::Geometry`. Kromě samotné geometrie je v algoritmech často potřeba pracovat se seznamem bodů jednoho či více prvků. V těchto případech je využita třída `geos::geom::CoordinateSequence` a `geos::geom::Coordinate`, která pak reprezentuje samostatný bod.

5.2 Vertex Snapper

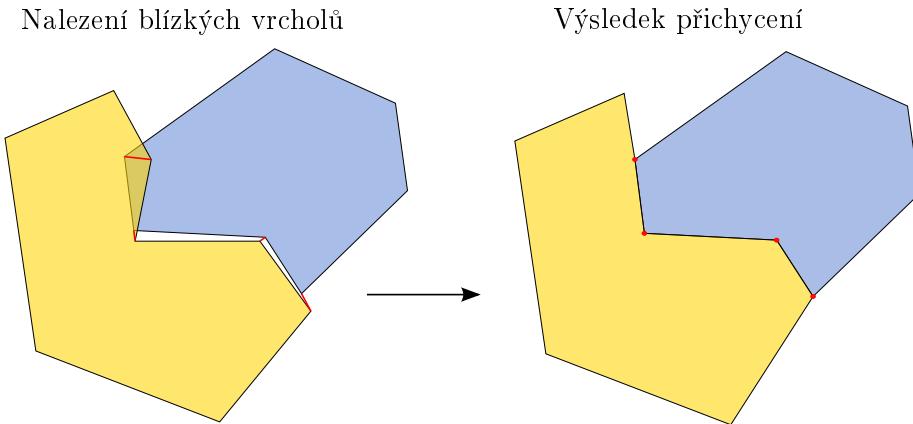
Při zpracování vrstev z více zdrojů někdy stačí pouze upřesnit polohu či tvar prvků z cílové vrstvy tak, aby se přiblížil prvkům z vrstvy referenční. Není-li podrobnost obou datasetů příliš rozdílná, lze využít jednoduchého postupu **přichycení blízkých vrcholů**. Tento algoritmus byl inspirován stejnojmenným zásuvným modulem knihovny JCS (viz kapitola 3.2.1).

5.2.1 Popis algoritmu

Nejjednodušším způsobem kombinace dvou vektorových vrstev je pouhé přichycení blízkých vrcholů cílové vrstvy k vrstvě referenční. Algoritmus lze obecně popsat následujícími několika body.

1. Na počátku je třeba určit vzdálenostní toleranci, tedy maximální vzdálenost mezi dvěma body, kdy ještě bude provedeno jejich přichycení.
2. Ke každému prvku ze zpracovávané vrstvy se najdou nejbližší prvky z vrstvy referenční. To jsou prvky, jejichž nejkratší vzdálenost od zpracovávaného prvku není větší než vzdálenostní tolerance.
3. Pro každý bod ze zpracovávaného prvku jsou vypočteny vzdálenosti ke všem bodům z blízkých referenčních prvků.

4. Pokud nejmenší z těchto délek je menší než vzdálenostní tolerance, pak je zpracovávaný bod posunut do odpovídajícího referenčního bodu s touto nejmenší vzdáleností.
5. Takto se prochází postupně všechny vrcholy všech prvků cílové vrstvy a vyhledávají se k nim blízké body z prvků vrstvy referenční.



Obrázek 5.1: Postup přichycení vrcholů

5.2.2 Implementace

Algoritmus pro přichycení vrcholů upravované vrstvy k referenční je implementován ve třídách `VertexSnapper` a `VertexGeometryEditorOperation`. Lepší představu o jednotlivých třídách lze získat z UML diagramu tříd v příloze B. Při použití v externí aplikaci stačí po předání nezbytných vstupních parametrů (k tomu slouží metody `setRefGeometry()`, `setSubGeometry()` a `setTolDistance()`) třídě `VertexSnapper` zavolat funkci `snap()`. Ta vyhledá blízké prvky s využitím prostorových indexů sestavených metodou `buildIndex()` třídy `SpatialIndexBuilder`.

Výsledky vyhledávání poté předá funkci `snapVertices()`. Uvnitř této metody se vytvoří instance třídy `VertexGeometryEditorOperation`, která edituje příslušnou geometrii přichycením blízkých vrcholů v metodě `edit()`. `VertexGeometryEditorOperation` je potomkem třídy `geos::operation::CoordinateOperation`, která je *interface*¹⁶ třídou pro editaci geometrie.

Pro kontrolu validity geometrií je volána metoda `isValid()` třídy `geos::geom::Geometry`.

Výsledné geometrie lze získat zavolením metody `getNewGeometry()`, která vrací vektor s interní reprezentací geometrií.

¹⁶Rozhraní třídy, kde jsou deklarovány pouze abstraktní metody bez implementace, není možné vytvořit instanci této třídy.

5.2.3 Využití

Přichycení bodů jedné vrstvy k vrstvě druhé má své výhody i nevýhody, které je před volbou tohoto způsobu zpracování třeba zvážit. Použití této metody je vhodné v takových případech, kdy jsou k dispozici dvě vrstvy o rozdílné přesnosti (tento rozdíl však nesmí být příliš veliký) a prvky vzájemně se překrývající. Cílem je upřesnit polohu a tvar prvků jednoho datasetu. Dopředu je třeba si uvědomit, že kromě polohy prvků je měněn i jejich tvar.

Využít by tento postup šel i pro přichycení dvou sousedních vrstev o stejně přesnosti, avšak to znamená, že by se změnil pouze tvar krajních prvků (nedošlo by k posunu celé vrstvy), a to nemusí být vždy žádoucí. Jako jediný z algoritmů GEOS Conflation (GEOC) má pak smysl pro bodové vrstvy.

Pro rozumné výsledky je důležité zvolit vhodnou toleranční vzdálenost. Tato hodnota by měla odpovídat maximální vzdálenosti, o kterou se vrchol prvku může posunout. Při volbě příliš krátké vzdálenosti se výsledná vrstva nemusí vůbec odlišovat od té vstupní. Naopak je-li zvolená vzdálenost delší než nejkratší úsek geometrie (linie, polygonu), může dojít k přichycení dvou bodů k jednomu bodu z referenční vrstvy. Zda je toto přípustné či nikoli, je už na rozhodnutí uživatele.

5.3 Coverage Alignment

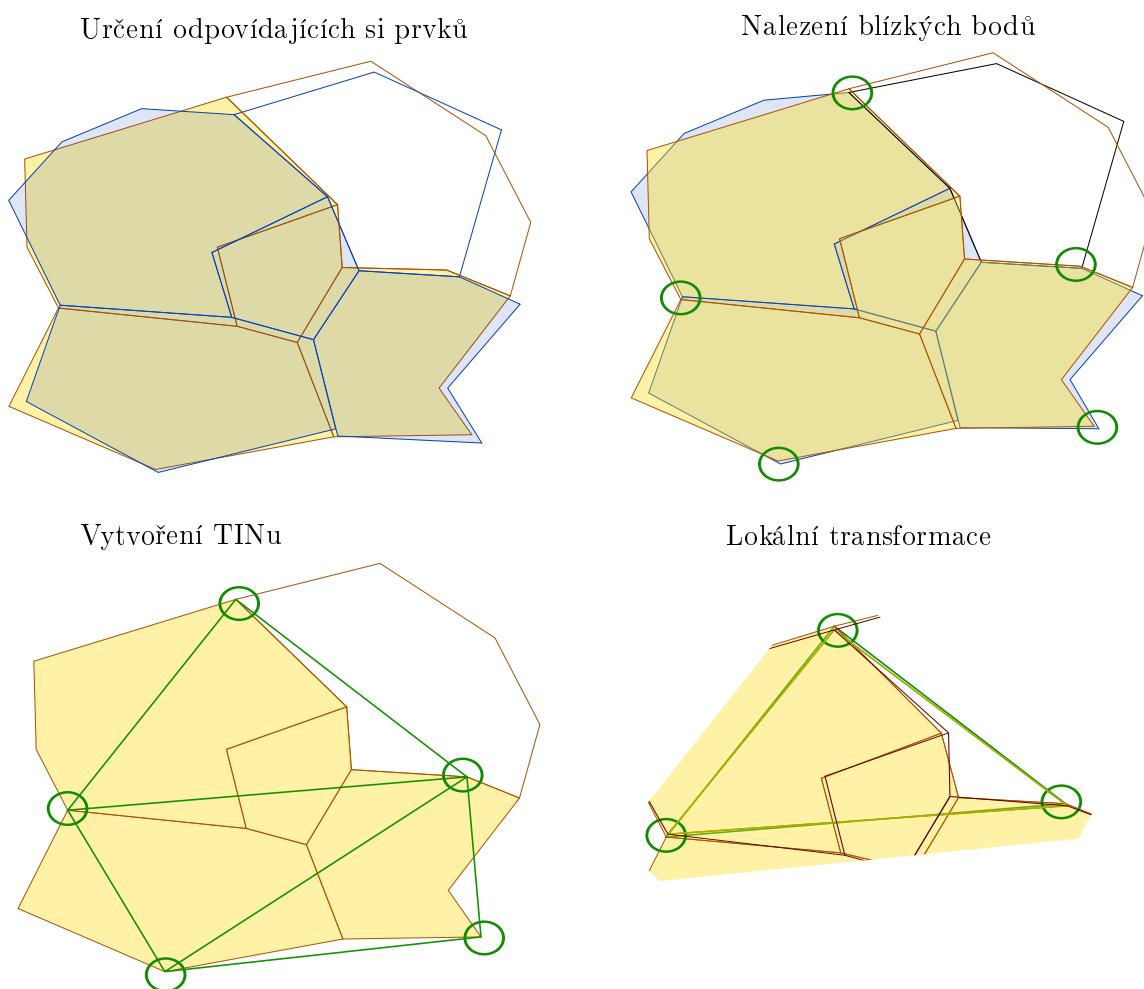
Coverage alignment lze vysvětlit jako **zarovnání jedné vrstvy k vrstvě druhé**. Tento způsob je složitější než výše uvedené přichytávání vrcholů. V knihovně GEOC je využit opět pro úpravu jedné vrstvy na základě vrstvy referenční. Do upravované vrstvy nejsou žádné prvky přidávány ani z ní mazány, jde pouze o jejich modifikaci. Velmi podobný algoritmus se dá použít i ke kombinaci dvou vrstev.

5.3.1 Popis algoritmu

Nejčastější používaný postup při spojování vektorových map je následující.

1. Nejprve je třeba nalézt odpovídající si prvky v obou překrývajících se vrstvách. Kritéria pro určení odpovídajících si prvků mohou být velmi odlišná. Existuje mnoho algoritmů řešících tuto problematiku, přičemž postupy se mohou různit podle toho, zda je úkolem vyhledání odpovídajících si bodů, polygonů či linií. Kritéria a postup použitý v knihovně GEOC je popsán níže.
2. Poté, co se určí odpovídající si prvky, musí se nalézt totožné vrcholy těchto dvojic prvků. Ty z vrcholů, které jsou určeny s dostatečnou přesností (ta může být určena například danou vzdálenostní tolerancí), jsou označeny jako body budoucí triangulační sítě.

3. Jak už bylo naznačeno, z nalezených bodů se vytvoří pomocí Delaunayho triangulace (*Delaunay Triangulation*)¹⁷ trojúhelníková síť.
4. Následně se provede lokální, nejčastěji affinní transformace v každém trojúhelníku sítě. Tak se přetrasformují body cílové vrstvy do systému vrstvy referenční.
5. Celý postup je možné iterativně opakovat, dokud není dosaženo požadovaného výsledku (ten může být dán např. podmínkou minimálního množství nově nalezených odpovídajících si vrcholů či prvků).



Obrázek 5.2: Postup zarovnání vrstev

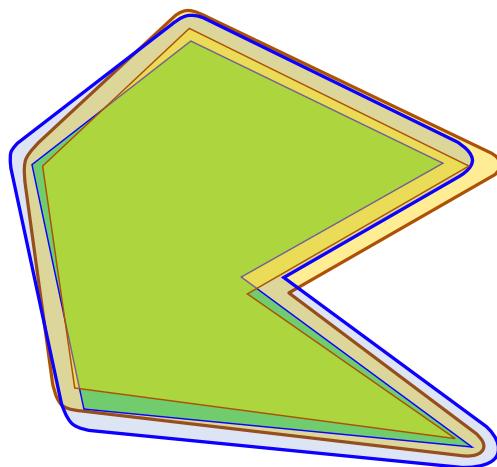
V knihovně GEOC je pro nalezení odpovídajících si prvků využito obdobného postupu jako ve výše zmiňované knihovně JCS. Využívá se vrcholová Hausdorffova vzdálenost,

¹⁷Delaunayho triangulace z množiny bodů v rovině vytvoří takovou trojúhelníkovou síť, pro kterou platí, že v kružnici opsané každému trojúhelníku, neleží žádný jiný bod. DT maximalizuje minimální úhly trojúhelníků.

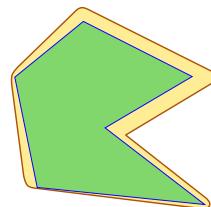
přičemž tato vzdálenost není počítána přímo. Splnění podmínky, že dané prvky nejsou od sebe dále, než je daná Hausdorffova vzdálenost, se testuje pomocí obalových zón jednotlivých prvků následujícím způsobem.

1. Mějme dva prvky A a B ze dvou různých překrývajících se vrstev.
2. Pokud prvek B leží v obalové zóně prvku A o velikosti vzdálenostní tolerance a A leží v obalové zóně prvku B o stejné velikosti, je možné, že si prvky odpovídají, a pokračuje se dalším krokem. Situace je naznačena na obrázku 5.3. V opačném případě si prvky neodpovídají.
3. Dále se testuje, zda hranice prvku B leží v obalové zóně hranice prvku A a naopak (viz obrázek 5.4). Je-li splněna i tato podmínka, pak jsou prvky označeny za odpovídající.

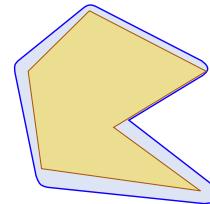
Test obalových zón prvků **A** a **B**



Prvek **A** leží v obalové zóně **B**

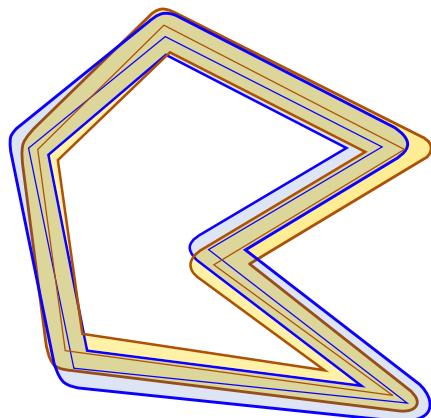


Prvek **B** leží v obalové zóně **A**

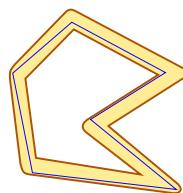


Obrázek 5.3: Porovnání prvků na základě jejich obalových zón

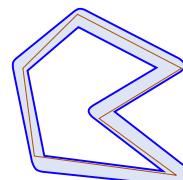
Test obalových zón hranic prvků **A** a **B**



Hranice **A** leží v obalové zóně hranice **B**



Hranice **B** leží v obalové zóně hranice **A**



Obrázek 5.4: Porovnání prvků na základě obalových zón jejich hranic

5.3.2 Implementace

Algoritmus je opět implementován pod jedinou metodou `align()` třídy `CoverageAlignment`, kterou je možné zavolat po nastavení základních parametrů. Ta postupně volá metody provádějící jednotlivé kroky algoritmu.

Nejdříve je třeba nalézt odpovídající si prvky. K tomu slouží třída `MatchingGeometry`, která k dané geometrii najde odpovídající (metoda `setMatch()`). Obdobně jako u `VertexSnapper` je i zde při hledání blízkých prvků využíváno prostorových indexů, které jsou vytvořeny metodou `SpatialIndexBuilder::buildIndex()`. Testy obalových zón jsou prováděny za pomoci funkcí `buffer()`, která vytvoří obalovou zónu, `getBoundary()`, která vrací hranici geometrie, a konečně `contains()`, která testuje, zda jedna geometrie obsahuje druhou. Všechny jsou přitom metodami třídy `geos::geom::Geometry`.

Určení blízkých bodů je provedeno prostřednictvím metody `chooseMatchingPoints()`, která dále využívá `findClosestPoints()` a `cleanMatchingPoints()` třídy `CoverageAlignment`.

Třetím krokem je vytvoření trojúhelníkové sítě metodou `createTIN()`, která k tomu využívá třídu `Triangulation`. Ta pouze aplikuje třídu `geos::triangulate::DelaunayTriangulationBuilder`, kde je vytvářen TIN ze zadaných bodů.

Konečně je provedena postupně transformace všech prvků. Funkce pro transformaci poskytuje třída `AffineTransformation`, která transformuje prvky na základě předané geometrie a triangulační sítě. Ta je volána prostřednictvím třídy pro editaci `AlignGeometryEditorOperation`.

Propojení jednotlivých tříd a nejdůležitější metody zobrazuje UML diagram tříd pro tento algoritmus v příloze B.

5.3.3 Využití

Na rozdíl od předchozího algoritmu je tento trochu šířeji využitelný. Je opět vhodný pro zpřesnění vrstvy dle vrstvy referenční, avšak tentokrát nejsou pouze přichytávány blízké vrcholy, ale jsou upravovány téměř všechny vrcholy. To zajišťuje reálnější výsledky i v situacích, kdy hustota vrcholů v obou datasetech je velmi rozdílná.

5.4 LineMatcher

Častým předmětem spojování vektorových dat jsou liniové mapy silnic, cest apod. Proto byl do knihovny doplněn i algoritmus *Line Matcher*, který je určen speciálně pro liniové vrstvy. Výrazný rozdíl oproti předchozím algoritmům je také fakt, že tento nezarovnává jednu vrstvu ke druhé, ale výsledkem je průměr z odpovídajících si prvků obou vrstev.

Nezáleží tedy na tom, která z vrstev je referenční. Prvky se zde nemyslí celé linie, ale pouze jejich segmenty.

5.4.1 Popis algoritmu

Jak je uvedeno výše, tento algoritmus pracuje pouze s úseků jednotlivých linií, a to z toho důvodu, že jedna dlouhá linie může být v jiném datasetu rozdělena na několik menších například přerušením na křížovatkách. Pro každou linii jednoho datasetu je pak provedeno následující.

1. Ke každému segmentu linie jsou nalezeny blízké linie z druhého datasetu. Stejně jako u předchozích, i zde je blízkost definována zvolenou toleranční vzdáleností.
2. Testovaný segment je dále porovnáván s každým úsekem z těchto blízkých linií na základě kritérií 5.1 uvedených níže.
3. Pokud je splněna předem zvolená hodnota podobnosti segmentů pro všechna kritéria S_1, S_2, S_3 , je výsledná podobnost určena hodnotou S .
4. Z odpovídajících segmentů je nalezen takový, jehož podobnost s testovaným je největší (hodnota S je co nejblíže jedné).
5. Výsledný segment je průměrem ze dvou nejlépe si odpovídajících segmentů.

Kritéria pro určení odpovídajících si liniových segmentů

První dvě kritéria (viz [9]) nejlépe určují podobnost dvou linií. Třetí kritérium bylo přidáno pro praktické použití tak, aby se neurčovaly pouze podobné prvky, ale i prvky blízké. Jednotlivá kritéria určující podobnost mohou nabývat hodnot od 0 do 1, kde 1 znamená naprostou shodu.

$$S_1 = \frac{\Delta l_{max} - \Delta l}{\Delta l_{max} - \Delta l_{min}}, \quad S_2 = \frac{\alpha_{max} - \alpha}{\alpha_{max} - \alpha_{min}}, \quad S_3 = \frac{d_{max} - d}{d_{max} - d_{min}} \\ S = \frac{S_1 + S_2 + S_3}{3} \quad (5.1)$$

Hodnoty Δl_{max} , Δl_{min} udávají maximální a minimální rozdíl délek segmentů v celém datasetu, Δl je pak rozdíl délek porovnávaných segmentů. α udává odchylku dvou segmentů. Hodnota α_{max} ve většině případů může být zvolena 90° , α_{min} pak 0° . Proměnná d ve třetím kritériu reprezentuje vzdálenost segmentů, která je spočtena jako průměr vzdáleností jejich počátečních a koncových bodů. Jelikož se porovnávají pouze blízké segmenty, d_{max} je rovno toleranční vzdálenosti, d_{min} by pak mělo odpovídat nejmenší vzdálenosti krajních bodů dvojice segmentů z obou datasetů.

5.4.2 Implementace

Veškeré funkce pro tento algoritmus jsou obsaženy ve třídě `LineMatcher` (blíže viz příloha B, obrázek B.3).

Funkce `match()` hledá postupně ke každé linii blízké linie z druhého datasetu. Následně předá výsledky tohoto hledání metodě `matchLine()`, která ke každému úseku linie nalezne prostřednictvím metody `matchingSegment()` odpovídající segment mezi předanými blízkými prvky. Pokud k nějakému úseku linie není určen žádný dostatečně podobný, je zde linie přerušena. Z původního jednoduchého liniového prvku tak může vzniknout multipek. Pro vytváření nových geometrií jsou využívány metody `createLineString()` a `createMultiLineString()` třídy `geos::geom::GeometryFactory` knihovny GEOS.

Výpočet kritérií podobnosti segmentů se nachází v metodě `similarity()`.

5.4.3 Využití

Line Matcher lze využít pro spojování liniových vrstev. Map různých cest, silnic, tras atd. existuje velké množství i díky rozvoji technologií GNSS¹⁸. Právě k jejich slučování slouží *Line Matcher*. Hodí se i pro vyhledání společných prvků obsažených ve dvou datasetech. Příkladem takové aplikace je určení cyklotras vedoucích po silnici z vrstvy silnic a cyklotras.

Vzhledem k požadavku co nejobecnějšího využití je výsledkem tohoto algoritmu průměr z odpovídajících si prvků. Někdy by však bylo vhodnější nechat prvkům původní polohu, pouze vymazat ty nepasující. Jindy může být požadavkem také ponechání i neodpovídajících si prvků apod. Různými drobnými modifikacemi by se tedy dalo dosáhnout lepšího využití pro nějaký konkrétní případ.

¹⁸Data tras z GNSS je nutné před použitím upravit, nejlépe zjednodušením linií, aby délka úseků přibližně odpovídala druhé vrstvě.

6 Zásuvný modul pro QGIS

Tato kapitola popisuje tvorbu zásuvného modulu **Conflate**. Uvedeny jsou nejdříve obecné informace o zásuvných modulech v programu Quantum GIS a poté už konkrétní popis tvorby modulu pro slučování map.

6.1 Tvorba QGIS zásuvného modulu

Do projektu Quantum GIS je možné psát zásuvné moduly v jazycích C++ nebo Python. Dnes už jsou častější zásuvné moduly psané v Pythonu, přesto se však objevují i ty v C++, a to z těchto důvodů. QGIS jako takový je napsán v tomto jazyce. Jedná se o objektově orientovaný jazyk, hodí se proto pro větší projekty. Nevýhodou C++ pluginů je nutnost komplikace, kterou však vyváží výsledná vyšší rychlosť běhu aplikace. V případě zásuvného modulu **Conflate** byl jazyk C++ zvolen i z důvodu použití knihovny GEOS, která je psána rovněž v tomto jazyce.

Vzhledem k tomu, že C++ zásuvné moduly většinou využívají Quantum GIS knihovny *libqgis*.so* publikováné pod licencí GPL, musí být tyto moduly pod stejnou licencí.

C++ zásuvné moduly QGISu jsou dynamické knihovny (*.so* nebo *.dll*¹⁹). Načtení modulu v QGISu probíhá za běhu programu. Při startu aplikace *Správce zásuvných modulů (Plugin Manager)* vyhledá všechny soubory s danou příponou ve složce se zásuvnými moduly, popřípadě v dalších složkách, jejichž umístění je definováno v uživatelském nastavení, a načte je. Pro správné načtení musí zásuvný modul obsahovat následující.

1. Funkce `classFactory()`, která vytvoří instanci daného pluginu.
2. Metoda `initGui()`, jejímž prostřednictvím jsou zobrazeny prvky uživatelského rozhraní (ikona apod.) v nabídce *Zásuvné moduly (Plugins)* a v liště nástrojů.
3. Metoda `unload()`, která odstraní alokované elementy a samotnou instanci třídy zásuvného modulu (pomocí destruktoru) při ukončení programu.
4. Další externí C funkce (`name()`, `description()`, `version()`, `category()` apod.) pro správné zobrazení ve *Správci zásuvný modulů*.

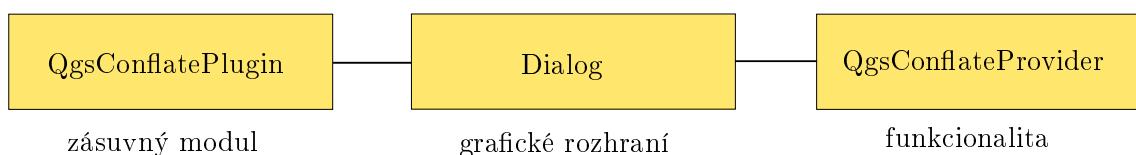
V modulu **Conflate** jsou všechny tyto funkce obsaženy ve třídě `QgsConflatePlugin`. Podrobnější popis tvorby C++ pluginu lze najít v *QGIS Coding and Compilation Guide* [10], odkud byly čerpány výše uvedené informace.

¹⁹ Přípona *.so* se používá v Linuxu, přípona *.dll* pod systémem Windows.

6.2 Zásuvný modul Conflate

Zásuvný modul **Conflate** pro QGIS, který je jedním z výstupů této práce, umožňuje spojení vektorových datasetů. **Conflate** provádí sjednocení dvou vrstev zvolených uživatelem vybranou metodou a parametry. Pro tento proces úpravy vrstev využívá tříd a metod výše popsané knihovny GEOF. Výstupem je nová vrstva a také textový protokol o zpracování.

Modul se skládá celkem ze tří tříd, a to `QgsConflateProvider`, `QgsConflatePlugin` a `QgsDialog`. Třída `QgsConflatePlugin` obsahuje funkce pro vytvoření a načtení zásuvného modulu do QGIS tak, jak bylo uvedeno v kapitole 6.1. Zobrazování a interakci s uživatelem zajišťuje třída `QgsDialog`. Vzhled grafického rozhraní byl vytvořen pomocí nástroje Qt Designer. Konečně třída `QgsConflateProvider` se stará o funkcionalitu aplikace a obsahuje tak funkce využívající knihovnu GEOF a další metody pro kopírování vrstvy, změnu geometrie, výpis protokolu apod.



Obrázek 6.1: Architektura zásuvného modulu *Conflate*

Následuje výčet činností probíhajících při spuštění a použití **Conflate**. Tento výčet má za úkol pouze zprostředkovat čtenáři představu, jak modul funguje. Podrobný návod pro jeho použití z hlediska uživatele je pak rozepsán v příloze C v uživatelské příručce. Popis jednotlivých metod lze nalézt v anglické dokumentaci k zásuvnému modulu na CD (příloha F).

1. Při otevření dialogu se do rozbalovací nabídky načtu názvy vrstev otevřených v programu.
2. Dalším krokem je výběr vrstev a nastavení možností uživatelem.
3. Po spuštění zpracování se nejdříve zkopíruje upravovaná vrstva (*Subject layer*) do nové vektorové vrstvy pomocí funkce `copyLayer()`. Formát nové vrstvy je vždy *shapefile*²⁰. Tato vrstva je vytvořena v aktuální složce a její název, pokud není zadán uživatelem, odpovídá vzoru

²⁰ESRI Shapefile je datový formát pro ukládání vektorových prostorových dat vyvinutý firmou Esri. Jedná se o jeden z nejrozšířenějších formátů dat pro geografické informační systémy, přípona souborů v tomto formátu je *.shp*.

`nazev_puvodni_vrstvy(nejnizsi_nepouzite_cislo).shp,`

např. `subject(3).shp.`

4. Poté je převedena geometrie prvků zvolených vrstev (upravovaná a referenční) na reprezentaci geometrie knihovny GEOC, tedy z `QgsGeometry` na `GEOCGeometry`. Tyto geometrie jsou uloženy do vektorů, které se pak předají třídám GEOC.
5. Jak už bylo naznačeno, je vytvořena instance příslušné třídy GEOC a té jsou předány parametry - vektory s prvky vybraných vrstev, toleranční vzdálenost a případně i další.
6. Dále je volána funkce vytvořené třídy pro zarovnání datasetů. Při volbě přichycení vrcholů *Snap vertices* v dialogu je použita třída `VertexSnapper` a metoda `snap()`, při volbě zarovnání vrstev *Coverage Alignment* třída `CoverageAlignment` a metoda `align()`, poslední možností je *Match Lines*, která využívá třídu `LineMatcher` a její metodu `match()`. Výsledek je uložen do nového vektoru.
7. Na závěr je upravena geometrie nové vrstvy (kopie upravované vrstvy) dle výsledků úprav provedených algoritmy knihovny GEOC.
8. Nová vrstva se automaticky přidá do aktuálního projektu v QGIS a zobrazí se v panelu *Vrstvy (Layers)*.
9. Po zpracování vrstvy se do textového okna dialogu vypíše protokol o zpracování, který obsahuje název vstupních a výstupních dat, počet zpracovávaných prvků a počet nevalidních prvků včetně výpisu jejich identifikátorů (*id*).

7 Problémy a jejich řešení

7.1 Použití knihovny GEOS

Jako poměrně zásadní problém se ukázalo použití knihovny GEOS. Přestože tato knihovna je napsána v jazyce C++, obsahuje *wrapper* (třída, program, knihovna apod., který slouží ke zprostředkování přístupu k nějakému dalšímu programu, jedná se pouze o nové rozhraní k existujícímu programu, které má umožnit kompatibilitu s externím programem apod.) v podobě C API. Hlavním důvodem toho je zajištění vyšší stability. Při použití C API totiž není nutné vlastní aplikaci, která ho používá, při každé změně v knihovně GEOS překompilovat. Proto většina programů využívajících GEOS, mezi nimi i Quantum GIS, používají právě C API. Nicméně to má i své nevýhody. V C API bohužel zatím není implementována veškerá funkcionality C++ API. Některé chybějící funkce lze obejít aplikací sice složitějšího postupu, nicméně výsledek je obdobný jako při použití C++ API. Avšak jiná funkcionality je pro uživatele C API zcela nedostupná. Po dlouhých úvahách²¹ bylo proto v knihovně GEOF přistoupeno k využití C++ API i s rizikem, že bude při každé změně nutné vše překompilovat.

S tím souvisí i další problém převodu geometrie. Jelikož v programu QGIS je používáno právě C API, nelze GEOS geometrii jednoduše převést na typ QGIS geometrie. Tento problém byl vyřešen konverzí mezi těmito dvěma typy přes WKT. Nevýhoda tohoto postupu je však zpomalení běhu programu obzvláště u většího objemu dat. Toto by mělo být v budoucnu vyřešeno vytvořením vlastního C API.

7.2 Rychlosť zpracováni

Vyhledání blízkých prvků

Základem použitých algoritmů je vždy nalezení blízkých či podobných prvků ze dvou vrstev. Pro nalezení takových dvojic prvků je obecně třeba porovnat každý prvek z jedné vrstvy s každým prvkem z vrstvy druhé. Je tedy třeba porovnat $n \cdot m$ dvojic, kde n je počet prvků vrstvy první a m vrstvy druhé. U rozsáhlějších datasetů je nutné provést poměrně velké množství operací a čas zpracování pak výrazně roste. Tento problém byl částečně vyřešen aplikací prostorových indexů, ty slouží k optimalizaci prostorových dotazů v prostorových databázích. Konkrétně byla využita třída `geos::index::STRtree` knihovny GEOS. Tato třída implementuje metodu indexování dat založenou na *R-Tree*²².

²¹O problému jsem napsala na *mailing-list* vývojářů GEOSu a odpověď bylo uvedení již zmíněných pro a proti. Problém jsem následně prodiskutovala s vedoucím práce, který mi doporučil toto řešení.

²²Česky R-strom, prostorová stromová datová struktura využívající minimální ohraňčující obdélníky.

Díky tomu se porovnávají pouze dvojice prvků, u nichž to má smysl, tedy takové, jejichž minimální ohraňující obdélníky (v některých algoritmech rozšířené o toleranční vzdálenost) se protínají.

Použití prostorových indexů rychlosť zpracování sice značně zvýšilo (podrobněji v kapitole 7.2.1), ale stále trvá zpracování velmi obsáhlých dat obzvláště u algoritmu **Coverage-Alignment** poměrně dlouhou dobu. Zvýšení rychlosti by bylo možné dosáhnout použitím prostorových indexů nejen při vyhledávání blízkých prvků, ale i při následném hledání blízkých bodů. V těchto případech nebyly prostorové indexy využity z toho důvodu, že pro reprezentaci bodů je v algoritmech použita třída `geos::geom::Coordinate`, pro níž však nejsou na rozdíl od `geos::geom::Geometry` prostorové indexy v knihovně GEOS implementovány.

Testování podobných prvků

Další částí programu, která působí výrazné zpomalení, je vyhledávání odpovídajících si geometrií ve třídě **MatchingGeometry**. Konkrétně se jedná o test, kdy jsou porovnávány obalové zóny prvků. Konstrukce obalových zón je výpočetně náročná, proto je žádoucí, aby se tento test prováděl u co nejmenšího počtu prvků. V algoritmu je tento počet snížen porovnáváním pouze blízkých prvků, přesto je takových prvků ještě mnoho.

7.2.1 Porovnání rychlosti s a bez použití prostorových indexů

Porovnání vlivu prostorových indexů na rychlosť zpracování bylo provedeno nad dvojicí polygonových vrstev `obce_ref` a `obce_sub`, které obsahují polygony obcí části ČR, a nad dvojicí liniových vrstev `zel_ref` a `zel_sub` obsahující výběr železnic ve Středočeském kraji. Použity byly pouze tyto výběry, nikoliv data z celé České republiky, jelikož na tak velkých datasetech zpracování bez prostorových indexů může trvat poměrně dlouho.

Testovány byly vždy uvedené dvojice datasetů pro různé toleranční vzdálenosti, a to při zpracování algoritmy **VertexSnapper** a **CoverageAlignment** s použitím prostorových indexů a bez jejich použití. Kompletní výsledky testování jsou uvedeny v tabulkách v příloze A, parametry použitého počítače pak tamtéž v tabulce A.1. Zde je uveden jen přehled výsledků testování.

Tab. 7.1: Výsledky testování - zrychlení algoritmů při použití prostorových indexů

toleranční vzdálenost	VertexSnapper		CoverageAlignment	
	100 m	1000 m	100/1000 m	10 000 m
polygonová data	20.9 → 0.8 s	22.6 → 0.7 s	4.7 → 0.5 s	4.9 → 0.5 s
liniová data	38.4 → 1.1 s	39.0 → 1.8 s	10.5 → 1.3 s	19.4 → 7.5 s

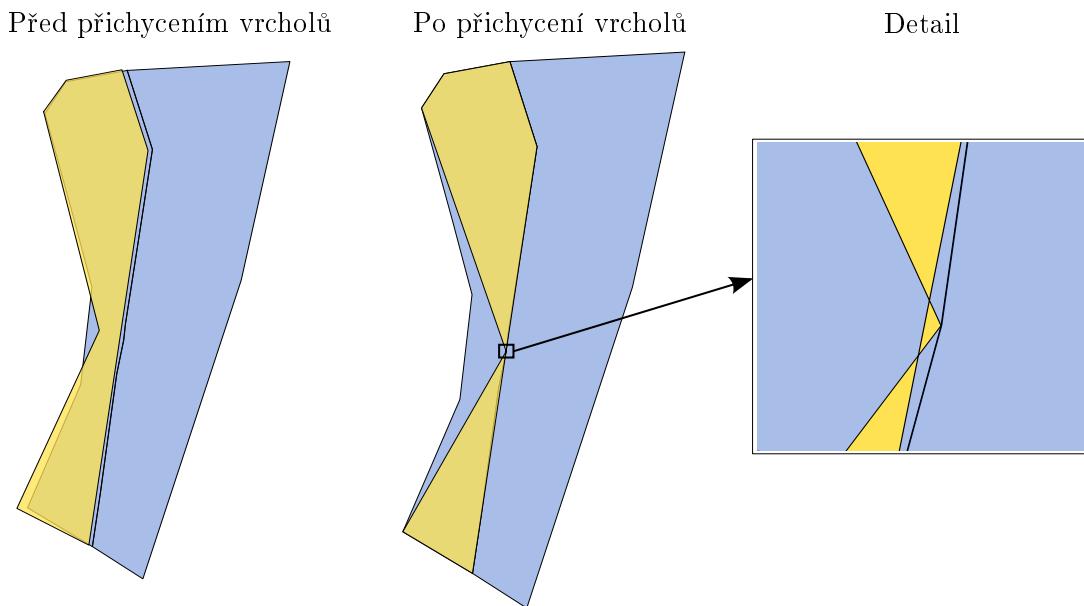
Z dat uvedených v tabulce 7.1 je patrné, že doplnění prostorových indexů oba algoritmy výrazně urychlilo (řádově více než desetkrát). Pro podrobnější rozbor a určení míry zrychlení by samozřejmě bylo nutné provést mnohem rozsáhlější testování. Zde bylo účelem pouze ukázat, že prostorové indexy algoritmy opravdu zrychlily.

Pro účely testování byla vytvořena jednoduchá konzolová aplikace, která je společně s testovanými daty také obsahem přiloženého CD (viz příloha F). Příklad použití této aplikace pro testování je pak popsán v příloze A.

7.3 Vertex Snapper

Nevalidní geometrie

Vzhledem k principu tohoto algoritmu mohou při jeho použití vznikat nevalidní geometrie, to je takové, jejichž segmenty se vzájemně protínají apod. To se nejčastěji stává u protáhlých úzkých prvků a jiných speciálních tvarů. Příkladem může být situace uvedená na obrázku 7.1, kde žlutý polygon je přichycen k referenční modré vrstvě, ale z důvodu nedostatečné hustoty vrcholů a nevhodného tvaru vzniká nežádoucí křížení.



Obrázek 7.1: Vznik nevalidní geometrie při přichycení vrcholů

Tyto chyby je samozřejmě možné opravit ručně po zpracování. Avšak v rozsáhlých datasetech může být nevalidních prvků poměrně velké množství. Pokud se toto nestalo z důvodu nevhodného nastavení zpracování, bylo by dobré, aby samotný algoritmus tyto chyby minimalizoval. To by bylo možné například přidáním nových vrcholů na vhodná místa (dlouhé segmenty bez vrcholu apod.). Další možností je zrušit přichycení vrcholu,

pokud při něm vznikne nevalidní geometrie. Nevýhoda takového postupu je horší zarovnání výsledné vrstvy.

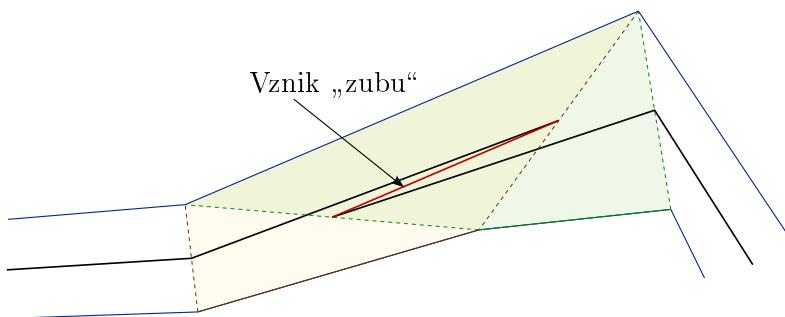
Přichycení do jednoho bodu

Speciálním případem vzniku nevalidní geometrie je situace, kdy se všechny vrcholy prvku přichytí pouze k jednomu bodu. Tím vzniká z linie či polygonu bodový prvek. K této situaci dochází v případě volby velké toleranční vzdálenosti v poměru k velikosti prvku. U datasetů obsahujících prvky o výrazně rozdílných velikostech se tomu nelze vždy vyhnout. V knihovně GEOF není tento problém zatím vyřešen. Nejjednodušší řešení, které by bylo možné implementovat, je nabídnout uživateli, co s takovými prvky provést. Na výběr by byly tyto možnosti: odstranění takových prvků, export těchto prvků do nové vrstvy, kde by bylo možné je dále upravovat, nebo ponechání původní geometrie.

7.4 Line Matcher

Navázání linií

Při určování podobných segmentů v rámci jedné linie nastává problém jejich navázání. V závislosti na nastavení funkce mohou při napojení výsledných segmentů vznikat na linii nežádoucí „zuby“ (viz obrázek 7.2). Tento problém v algoritmu nebyl vyřešen, jelikož nelze jednoduše rozhodnout, kdy se jedná o nesprávný tvar linie a kdy je výsledek takto korektní. Aby nevznikaly pouze samostatné nenavazující segmenty, byly i přes toto riziko jednotlivé úseky napojovány.



Obrázek 7.2: Příklad situace, kdy vzniká „zub“

7.5 Matching Geometry

Algoritmus *Matching Geometry*, který využívá třída `CoverageAlignment`, vyhledává odpovídající si geometrie tak, že nejdříve naleze blízké prvky k dané geometrii a pak testuje postupně všechny tyto blízké prvky. Po nalezení prvního odpovídajícího prvku se už další prvky netestují. To může způsobit, že je nalezen sice odpovídající prvek, avšak nemusí to

být ten, který je danému prvku podobný nejvíce. U polygonových vrstev při vhodné volbě toleranční vzdálenosti tento problém většinou nenastává, ale u vrstev liniových se může objevit poměrně často.

Řešením by bylo seřadit před testováním blízké prvky dle nějakého kritéria, které by určovalo pravděpodobnost, že si budou odpovídat. Takové kritérium je však obtížné určit. Dalším možným řešením je po nalezení všech prvků, které splňují podmínku podobnosti, porovnat plochy překrývajících se obalových zón.

Výše uvedeným problémům se u liniových vrstev lze vyhnout použitím jiného algoritmu *RoadMatcher/LineMatcher*, který porovnává pouze jednotlivé úseky linií, nikoli celé prvky.

7.6 Coverage Alignment

Vytvoření trojúhelníkové sítě

Po nalezení odpovídajících si prvků je nutné určit vrcholy triangulační sítě. V použitém algoritmu jsou zvoleny dvojice blízkých vrcholů dvou odpovídajících si prvků. Důsledkem tohoto výběru může být nepravidelné rozložení vrcholů sítě a také vytvoření trojúhelníkové sítě pouze nad malou částí území, a tudíž nezahrnující všechny odpovídající si prvy.

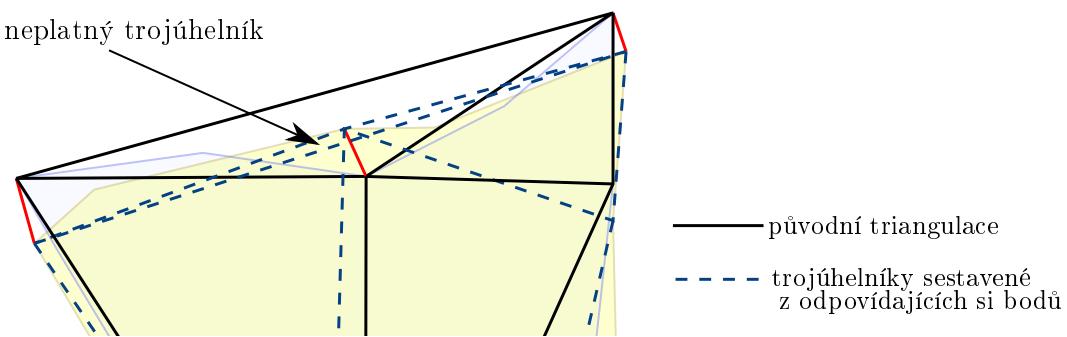
Pro pokrytí větší části území byly proto zahrnuty do vrcholů rohové body, jejichž souřadnice odpovídají extrémním souřadnicím ze všech vrcholů prvků zpracovávané vrstvy, ke kterým byl nalezen ekvivalent ve vrstvě referenční. Tyto body mají v obou systémech pro transformaci totožné souřadnice. Tímto postupem se nejen zajistí zpracování všech odpovídajících si prvků, ale také zarovnání prvků i v případech, kdy se jako odpovídající určí pouze jeden nebo dva body, z nichž by bez této úpravy nebylo možné TIN sestrojit. Nevýhodou zvoleného řešení jsou nevhodné tvary trojúhelníků na okrajích sítě a tím způsobené méně přesné zarovnání v těchto oblastech. Avšak i přes tyto nevýhody jsou výsledky zarovnání datasetů s použitím rohových bodů přesnější. Vylepšením tohoto řešení by mohlo být použití bodů konvexní obálky pasujících prvků namísto pouze čtyř rohových bodů.

Velký vliv na tvar TIN má způsob volby blízkých bodů. V některých programech (např. ArcGIS) je proto tento krok přenechán uživateli. Počet dvojic bodů je sice obecně nižší, avšak jsou tak zvoleny pouze skutečně si odpovídající body a jejich rozložení je většinou pravidelné. Výsledek je pak blíže skutečnosti než automatické zpracování. Navíc bylo experimentálně zjištěno, že větší počet trojúhelníků nezajistí větší přesnost, a obecně tak postačí jedna dvojice bodů pro padesát prvků (viz [7]).

Nevalidní geometrie

V případě nevhodné konfigurace odpovídajících si trojúhelníků může v důsledku lokální transformace dojít ke vzniku nevalidních geometrií. To je způsobeno faktem, že odpovídající trojúhelníky jsou sestrojovány z odpovídajících si bodů, nikoli triangulací (tím by totiž mohly vzniknout trojúhelníky jiného tvaru). Tudíž druhá trojúhelníková síť nemusí být platná. Nejsnazší řešení je vyřadit takové body z množiny vrcholů TIN a vypočítat triangulaci znovu.

Tento problém úzce souvisí s volbou toleranční vzdálenosti. Při dodržení pravidla o nejkratším segmentu se mu lze téměř vyhnout (stále mohou vznikat nevalidní geometrie u příliš úzkých, protáhlých tvarů apod.).



Obrázek 7.3: Vznik neplatné trojúhelníkové sítě

Volba počtu iterací

V popisu algoritmu zmiňuji, že proces zarovnání může probíhat iterativně. Otázkou je volba počtu iterací. Nabízí se tři možnosti.

1. konstantní počet iterací
2. ponechání volby počtu iterací na uživateli
3. opakování procesu tak dlouho, dokud není splněna určitá podmínka

Pevně daný počet není příliš vhodný, jelikož vůbec nezohledňuje výsledky předchozí iterace. Vzhledem k tomu, že běžný uživatel by pravděpodobně obtížně určil správný počet iterací, zůstává třetí možnost. Tím vyvstává problém zvolení vhodné podmínky. Může to být počet nových odpovídajících si prvků, počet nově nalezených dvojic bodů, počet zpracovaných prvků aj. Dle [13] obvykle postačí dvě až tři opakování, přičemž mezi jednotlivými cykly je vhodná kontrola operátora, který by měl odstranit případné hrubé chyby. Zároveň navrhujeme jako podmínu počet nově nalezených odpovídajících si prvků. Cyklus se tedy

opakuje, dokud jsou nacházeny nové dvojice pasujících prvků. Tento postup je v knihovně GEOC implementován bez možnosti kontroly mezi iteracemi.

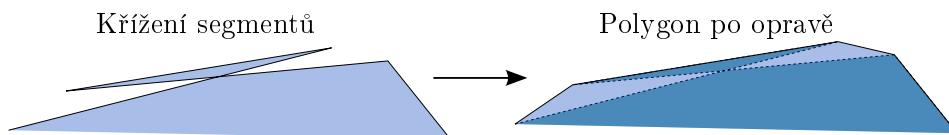
7.7 Automatická oprava nevalidních geometrií

Jak již bylo uvedeno výše, častým problémem u všech algoritmů je vznik nevalidních prvků. Proto byla do knihovny GEOC přidána třída `GeometryCorrectionOperation`, kterou lze pro dodatečnou opravu nově vzniklých geometrií použít. Ne vždy je však jednoduché rozhodnout, jak geometrii změnit, aby výsledek odpovídal skutečnosti. V programu proto byla aplikována jedna možnost způsobu opravy a na uživateli ponecháno rozhodnutí, zda automatickou korekci využije.

Automaticky je možné opravit tyto dvě chyby v geometrii:

- Křížení segmentů

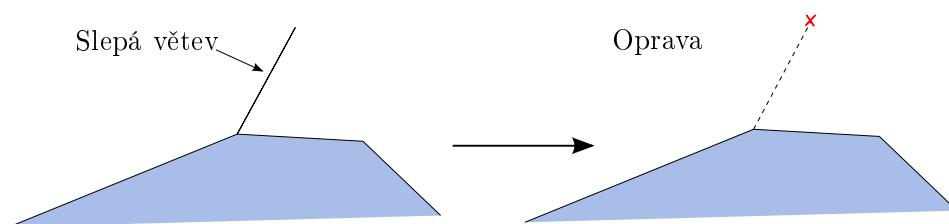
Změněna orientace hrany či hran mezi křížícími se úsekami.



Obrázek 7.4: Oprava křížících se segmentů

- Slepé větve

Vyřešeno odstraněním bodu na konci větve.



Obrázek 7.5: Oprava slepých větví

Samozřejmě existuje více možností, jak výše uvedené chyby opravit. U křížících se polygonů se často používá také pouhé přidání vrcholů do místa průsečíku segmentů. Tím vznikne validní geometrie a nezmění se přitom její tvar. Vzhledem k tomu, že u úprav prováděných s polygony při slučování map, vznikají křížením většinou spíše drobné „výstupky“, je použité řešení přijatelné. Navíc funkci pro opravu geometrií přidáním bodů obsahuje většina GIS programů, takže není pro uživatele složité ji po spojení dat v rámci

dodatečných úprav aplikovat. Stejně tak se ručně musí opravit další chyby jako je otvor v polygonu dotýkající se vnějšího okraje atd.

8 Další vývoj

V rámci dalšího vývoje zásuvného modulu i knihovny GEOC by bylo vhodné vylepšit některé algoritmy a také funkcionality. Některé možné úpravy algoritmů jsou už zmíněny v části 7, proto je zde již neopakuji. Samozřejmě vždy lze optimalizovat kód a zvýšit efektivitu aplikace, tato kapitola však obsahuje spíše konkrétní návrhy, co by se dalo do programu doplnit, aby měl širší uplatnění.

Nejbližším krokem by mělo být vytvoření *Makefile*²³ a konfiguračního souboru, který by uživatelům umožnil snadné zprovoznění knihovny a zásuvného modulu s využitím standardních nástrojů.

8.1 Knihovna GEOC

Zpracování po dlaždicích

Větší efektivita při zpracování rozsáhlých dat je v GIS aplikacích zajištěna většinou jejich zpracováním po částech a paralelizací algoritmu (části se zpracovávají souběžně). Je možné si data rozdělit na pravidelné dlaždice a každou tuto část upravovat samostatně. Ještě vhodnější způsob u této aplikace je dle [5] rozčlenění dat pomocí shlukových algoritmů a následné rozdělení na jednotlivá pole Voronoi teselaci²⁴.

Bez tohoto postupu je využití programu omezeno pouze na menší objem dat. Uživatel by si sice mohl data na části rozdělit sám a zpracovávat je postupně, to je však nepohodlné a časově náročné, nehledě na to, že by musel následně řešit problém úpravy prvků na styku jednotlivých částí.

Další algoritmy

Zatím jsou v knihovně GEOC implementovány pouze některé základní algoritmy pro slučování map. V rámci dalšího vývoje by bylo možné postupně doplňovat další pro její širší využití.

V současné době je v oblasti slučování vektorových map (*conflation*) snaha o hledání společných elementů dvou datasetů nejen na základě geometrie, ale i s přihlédnutím k atributům prvků. Zakomponování tohoto přístupu do algoritmu pro určení odpovídajících si prvků je proto logicky vhodným dalším krokem jeho vývoje.

²³Soubor určující postup nástroje *make* při překladu zdrojových souborů do binárních souborů.

²⁴Rozdělení roviny na nepravidelné Voronoi polygony, které jsou duální k trojúhelníkům DT.

8.2 Zásuvný modul Conflate

Protokol a výsledek zpracování

U zásuvného modulu by bylo vhodné usnadnit uživateli vyhledání a opravu výsledných nevalidních geometrií. Nyní si musí sám vyhledat prvky dle identifikátorů v protokolu. Ideálně by mohl být po kliknutí na daný identifikátor v okně projektu zobrazen a označen příslušný prvek.

Co se týče protokolu, lze ještě doplnit možnost jeho uložení do textového souboru.

Volba toleranční vzdálenosti

K lepšímu a pohodlnějšímu spojení map by zajisté vedlo doporučení toleranční vzdálenosti, jelikož právě její volba může vést k nereálným výsledkům. Nezkušený uživatel by tak nemusel metodou „pokus-omyl“ zjišťovat, jak tento parametr nastavit, pouze by ho mohl upravit dle svých představ. Jak už bylo několikrát řečeno, vyhovujícím návrhem by byl průměr z několika nejkratších segmentů prvků dané vrstvy.

9 Závěr

Cílem této bakalářské práce bylo nastudování algoritmů pro slučování vektorových map (*vector to vector conflation*) a následné vytvoření zásuvného modulu pro Quantum GIS, který pomocí těchto algoritmů umožní zpracování dat. Kromě toho je výstupem práce i přehled některých existujících nástrojů zabývajících se slučováním map.

Výsledný zásuvný modul **Conflate** z daných vstupních vrstev na základě parametrů zvolených uživatelem vytvoří novou vrstvu, která je kombinací původních. Samotné algoritmy týkající se slučování datasetů jsou implementovány v externí C++ knihovně GEOC využívající knihovnu GEOS.

Knihovna GEOC je navržena nezávisle na QGIS API tak, aby ji bylo možné případně později využít i jinými programy. Zatím obsahuje tři základní algoritmy pro sloučení vektorových dat. K problému sjednocení vektorových map existuje mnoho přístupů. Aplikace všech by však byla příliš náročná, proto byl zvolen jediný přístup zabývající se pouze geometrickými vlastnostmi dat.

Jelikož se jedná o poměrně novou oblast v GIS, bylo obtížné najít literaturu, která by popisovala nějaké obecné algoritmy. Většina dosavadních prací či článků se totiž zabývá spojováním map pro nějaký specifický případ. Proto bylo při implementaci nutné některé věci řešit na základě vlastního uvážení. Přesto však výsledky zpracování pomocí použitých algoritmů odpovídají původní představě o funkcionalitě zásuvného modulu.

Přestože knihovnu i zásuvný modul je možné ještě v různých směrech mírně vylepšit, lze říci, že stanovené cíle práce byly splněny. Zásuvný modul už nyní lze využít pro některé praktické úlohy. Algoritmus *Vertex Snapper* se hodí například pro napojení dvou blízkých vrstev, aby výsledkem byla topologicky správná data, nebo pro napasování hranic parcel k hranicím katastrálního území atp. *Coverage Alignment* lze využít pro úpravu méně přesné, avšak obsáhlější mapy dle přesnějšího zdroje. Konkrétně například pro zarovnání map z OpenStreetMap, které jsou sice velmi podrobné, ale nedosahují takové geometrické přesnosti. Poslední algoritmus *LineMatcher* je díky svému zaměření na liniová data vhodný pro spojování map silničních sítí, zpřesnění map tras vytvořených pomocí GNSS technologií apod.

Nejbližším krokem by nyní mělo být uvedení projektu do takového stavu, aby mohl být poskytnut uživatelům programu Quantum GIS.

Seznam zkratek

API	Rozhraní pro programování aplikací (<i>Application Programming Interface</i>)
ČÚZK	Český úřad zeměměřický a katastrální
DT	Delaunayho triangulace (<i>Delaunay Triangulation</i>)
GEOS	Geometry Engine, Open Source
GEOC	GEOS Conflation
GIS	Geografický informační systém (<i>Geographic Information System</i>)
GNSS	Globální navaigacní satelitní systémy
GPL	GNU General Public License
IDE	Vývojové prostředí (<i>Integrated Development Environment</i>)
JCS	Java Conflation Suite
JOSM	Java OpenStreetMap Editor
JTS	Java Topology Suite
LGPL	GNU Lesser General Public License
OGC	Open Geospatial Consortium
OSGeo	Open Source Geospatial Foundation
QGIS	Quantum GIS
SQL	Structured Query Language
TIN	Nepravidelná trojúhelníková síť (<i>Triangulated Irregular Network</i>)
UI	Uživatelské rozhraní (<i>User Interface</i>)
UML	Unified Modeling Language
WKT	Well Known Text
WKB	Well Known Binary

Literatura

- [1] BLASBY, Dave et al. *GIS Conflation using Open Source Tools* [online]. 2004. Dostupné z: <http://130.203.133.150/viewdoc/summary;jsessionid=8DAC68DCD24ADA3B38CFEE24108FD070?doi=10.1.1.195.9292>.
- [2] BOYCE, Richard D. a CHASE, Richard. Conflation System MBP. In: *Proceedings of CISST 2003, Las Vegas, Nevada*, [online]. CSREA Press, 2003. s. 246–252. Dostupné z: <http://www.cwu.edu/~Imaglab/docs/1022CT.pdf>.
- [3] COBB, Maria A. et al. A Rule-based Approach for the Conflation of Attributed Vector Data. *Geoinformatika*. 1998, roč. 2, č. 1, s. 7–35. Dostupné z: <http://80.link.springer.com.dialog.cvut.cz/article/10.1023/A%3A1009788905049>.
- [4] DAVIS, M. a AQUINO, J. *Java Conflation Suite: Technical Report*. Vivid Solutions, Inc. Dostupné z: <http://www.vividsolutions.com/jcs/bin/JCS%20Technical%20Report.pdf>.
- [5] FRÉITAS, Sergio a ALFONSO, Anna P. Distributed Vector based Spatial Data Conflation Services. In: NAMIKAWA, Laércio M. a BOGORNY, Vania (Ed.) *Proceedings XIII GEOINFO, listopad 2012, Campos do Jordao, Brazil*, [online]. 2012. s. 23–29. Dostupné z: <http://www.geoinfo.info/geoinfo2012/papers/freitas.pdf>.
- [6] LI, Linna a GOODCHILD, Michael F. Optimized Feature Matching in Conflation. In: *GIScience 2012, Zurich, Switzerland*, [online]. 2010. Dostupné z: http://www.giscience2010.org/pdfs/paper_111.pdf.
- [7] LUPIEN, Anthony E. a MORELAND, William H. A General Approach to Map Conflation. In: CHRISMAN, Nicholas R. (Ed.) *Proceedings of the International Symposium on Computer-Assisted Cartography, březen–duben 1987, Baltimore, Maryland*, [online]. Falls Church, Virginia: American Society for Photogrammetry and Remote Sensing, 1987. s. 630–639, [cit. 2013-02-10]. Dostupné z: <http://mapcontext.com/autocarto/proceedings/auto-carto-8/pdf/a-general-approach-to-map-conflation.pdf>.
- [8] LYNCH, Maureen P. a SAALFELD, Alan J. Conflation: Automated map compilation - A video game approach. In: *Proceedings of the Digital Representations of Spatial Knowledge, březen 1985, Washington DC, USA*, [online]. Falls Church, Virginia:

- American Society for Photogrammetry and Remote Sensing, 1985. s. 342–353. Dostupné z: <http://mapcontext.com/autocarto/proceedings/auto-carto-7/pdf/conflation-automated-map-compilation-a-video-game-approach.pdf>.
- [9] MOOSAVI, Azam a ALESHEIKH, Ali A. Developing of Vector Matching Algorithm Considering Topologic Relations. In: *Map Middle East 2008, Dubai, UAE*, [online]. 2008. Dostupné z: http://www.gisdevelopment.net/proceedings/mapmiddleeast/2008/mme08_40.pdf.
- [10] Quantum GIS Development Team. *QGIS Coding and Compilation Guide: Version 1.6 'Copiapó'*. OSGeo, 2004. Dostupné z: http://download.osgeo.org/qgis/doc/manual/qgis-1.6.0_coding-compilation_guide_en.pdf.
- [11] RESSLER, James; FREESE, Eric a BOATEN, Veleria. Semantic Method of Conflation. In: KOLAS, D.; WIEGAND, N. a BERG-CROSS, G. (Ed.) *Proceedings of the Terra Cognita Workshop, 2009, Washington DC, USA*, [online]. Washington: CEUR-WS, 2009. 518. Dostupné z: http://ceur-ws.org/Vol-518/terra09_submission_3.pdf.
- [12] ROSEN, Barbara a SAALFELD, Alan. Match Criteria for Automatic Alignment. In: *Proceedings of the Digital Representations of Spatial Knowledge, březen 1985, Washington DC, USA*, [online]. Falls Church, Virginia: American Society for Photogrammetry and Remote Sensing, 1985. s. 456–462. Dostupné z: <http://mapcontext.com/autocarto/proceedings/auto-carto-7/pdf/match-criteria-for-automatic-alignment.pdf>.
- [13] SAALFELD, Alan. Conflation: Automated map compilation. *International Journal of Geographical Information Systems*. [online]. 1988, roč. 2, č. 3, s. 217–228. ISSN 0269-3798. Dostupné z: <http://www.census.gov/srd/papers/pdf/rr87-24.pdf>.
- [14] SHEKHAR, Shashi a XIONG, Hui (Ed.). *Encyclopedia of GIS*. Springer Reference. New York: Springer, 2008. ISBN 978-0-387-35975-5. Dostupné z: <http://www.springer.com/computer/database+management+%26+information+retrieval/book/978-0-387-30858-6>.
- [15] SUTTON, Tim; SHERMAN, Gary a HUGENTOBLER, Marco. *Developers guide for QGIS* [online]. [cit. 2013-02-10]. Dostupné z: qgis.org/api/CODING.html.
- [16] WARE, J. Mark a JONES, Christopher B. Matching and Aligning Features in Overlaid Coverages. In: *Proceedings of 6th International Symposium on Advances in*

- Geographical Information Systems (ACM-GIS'98)*, [online]. 1998. s. 28–33. Dostupné z: <http://users.cs.cf.ac.uk/C.B.Jones/WareACMGIS98.pdf>.
- [17] WIEMANN, Stefan a BERNARD, Lars. Conflation Services within Spatial Data Infrastructures. In: *Proceedings of Agile 2010, Guimaraes, Portugal*, [online]. 2010. Dostupné z: http://agile.gis.geo.tu-dresden.de/web/Conference_Paper/CDs/AGILE%202010/ShortPapers_PDF/134_DOC.pdf.
- [18] ZHAOA, Dongbao; SHENG, Yehua a GUO, Hengliang. An algorithm for automatically matching corresponding points on homonymous map features. In: LIU, Lin; LI, Xia a KAI LIU, Xinchang Zhang (Ed.) *Geoinformatics 2008 and Joint Conference on GIS and Built Environment: Advanced Spatial Data Models and Analyses 2009*, [online]. 2009.
- [19] *ArcGIS Resource Center - Desktop 10 Help* [online]. Esri. Naposledy editováno 27. 6. 2012. [cit. 2013-03-01]. Dostupné z: <http://help.arcgis.com/en/arcgisdesktop/10.0/help/>.
- [20] *GEOS - Geometry Engine, Opens Source* [online]. [cit. 2013-04-30]. Dostupné z: <http://trac.osgeo.org/geos/>.
- [21] *JCS Conflation Suite* [online]. Vivid Solutions, Inc. [cit. 2013-03-10]. Dostupné z: <http://www.vividsolutions.com/jcs/>.
- [22] *OSM Conflation* [online]. Naposledy editováno 25. 4. 2012. [cit. 2013-02-20]. Dostupné z: <http://wiki.openstreetmap.org/wiki/Conflation>.
- [23] *PyQGIS Developer Cookbook* [online]. Martin Dobias. Dostupné z: <http://www.qgis.org/pyqgis-cookbook/>.
- [24] *Quantum GIS* [online]. OSGeo. [cit. 2013-03-10]. Dostupné z: <http://www.qgis.org/>.
- [25] *Qt* [online]. Digia. [cit. 2013-04-10]. Dostupné z: <http://qt.digia.com/>.
- [26] *Qt Project 4.8 Documentation* [online]. Qt Project Hosting. [cit. 2013-04-10]. Dostupné z: <http://qt-project.org/doc/qt-4.8/>.
- [27] *Wikipedie, anglická a česká verze* [online]. Wikipedia Foundation. [cit. 2013-04-30]. Dostupné z: <http://wikipedia.org/>.

Seznam příloh

A Porovnání rychlosti algoritmů	73
B UML diagramy tříd	77
C Uživatelská příručka	81
D Screenshoty zásuvného modulu	85
E Ukázky zpracování dat	87
F Obsah CD	93

A Porovnání rychlosti algoritmů po použití prostorových indexů

Knihovnu GEOC lze pro účely testování přeložit s definováním makra `WITHOUT_SPIDX`. Tím lze získat verzi knihovny bez prostorových indexů. Bez nastavení této možnosti knihovna automaticky prostorové indexy používá. Pro testování byla vytvořena aplikace `geoc_testing`, která byla volána pomocí jednoduchého skriptu pro `bash` (aplikaci, skript, i data lze nalézt na CD).

Porovnání bylo provedeno měřením času výpočtu pro daný algoritmus pro různá vstupní data a dvě různě zvolené toleranční vzdálenosti. Pro každou konfiguraci (algoritmus, vstupní data, toleranční vzdálenost) byl měřen čas výpočtu celkem desetkrát bez použití a desetkrát s použitím prostorových indexů. Cílem bylo určení změny rychlosti algoritmů po zavedení prostorových indexů. Testování bylo provedeno na přenosném počítači s parametry uvedenými v tabulce A.1.

Tab. A.1: Parametry počítače využitého pro testování

typ počítače	<i>HP ProBook 6450b</i>
paměť	<i>4 GiB</i>
procesor	<i>Intel® Core™ i5 CPU M 450 @ 2.40GHz × 4</i>
operační systém	<i>Ubuntu 12.04 LTS, 64 bit</i>

V tabulkách A.3 - A.6 jsou uvedeny časy zpracování pro danou upravovanou a referenční vrstvu (tabulka A.2) a zvolenou toleranční vzdálenost. Časy jsou uváděny v sekundách.

Tab. A.2: Vstupní vrstvy

typ dat	referenční vrstva	upravovaná vrstva
polygony	obce_ref	obce_sub
linie	zel_ref	zel_sub

Tab. A.3: VertexSnapper - čas zpracování bez prostorových indexů

id	polygony		linie	
	110 m	10 000 m	110 m	10 000 m
1	19.89	23.52	4.41	5.16
2	19.83	23.00	4.41	4.44
3	21.98	23.63	4.43	4.66
4	22.09	21.40	4.89	5.18
5	21.99	21.50	4.90	5.20
6	19.89	22.59	4.89	4.67
7	19.76	22.59	4.91	5.17
8	19.77	22.21	4.89	5.15
9	21.95	23.61	4.45	4.92
10	22.00	21.97	4.48	4.68
průměr	20.92	22.60	4.67	4.92

Tab. A.4: VertexSnapper - čas zpracování s prostorovými indexy

id	polygony		linie	
	110 m	10 000 m	110 m	10 000 m
1	0.78	0.69	0.45	0.47
2	0.75	0.72	0.51	0.49
3	0.69	0.76	0.48	0.50
4	0.72	0.71	0.50	0.45
5	0.77	0.78	0.48	0.48
6	0.82	0.76	0.49	0.49
7	0.90	0.70	0.43	0.50
8	0.69	0.78	0.47	0.44
9	0.68	0.69	0.43	0.51
10	0.76	0.84	0.44	0.49
průměr	0.76	0.74	0.47	0.48

Tab. A.5: CoverageAlignment - čas zpracování bez prostorových indexů

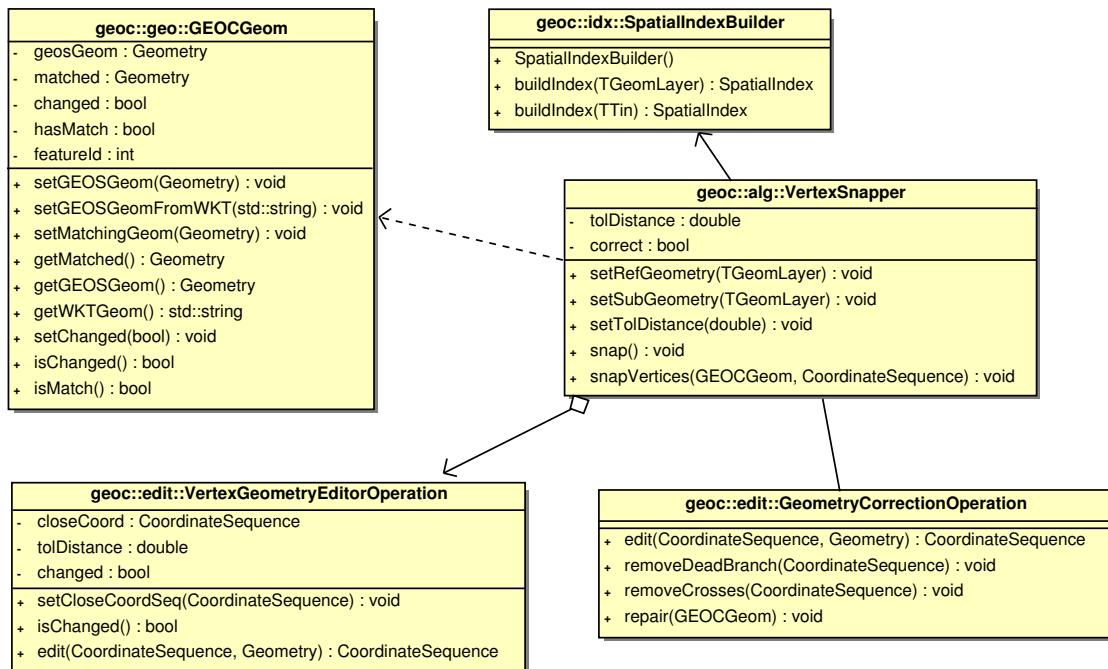
id	polygony		linie	
	110 m	1 000 m	1 000 m	10 000 m
1	38.75	37.68	10.02	18.85
2	38.55	37.93	10.36	20.94
3	38.51	37.44	10.24	18.77
4	41.6	36.95	11.22	18.74
5	34.02	41.81	10.04	18.83
6	40.31	38.55	10.44	18.73
7	38.77	41.39	11.21	20.91
8	38.54	36.79	10.09	18.98
9	37.69	39.69	9.67	18.81
10	36.72	41.6	11.67	20.87
průměr	38.35	38.98	10.51	19.44

Tab. A.6: CoverageAlignment - čas zpracování s prostorovými indexy

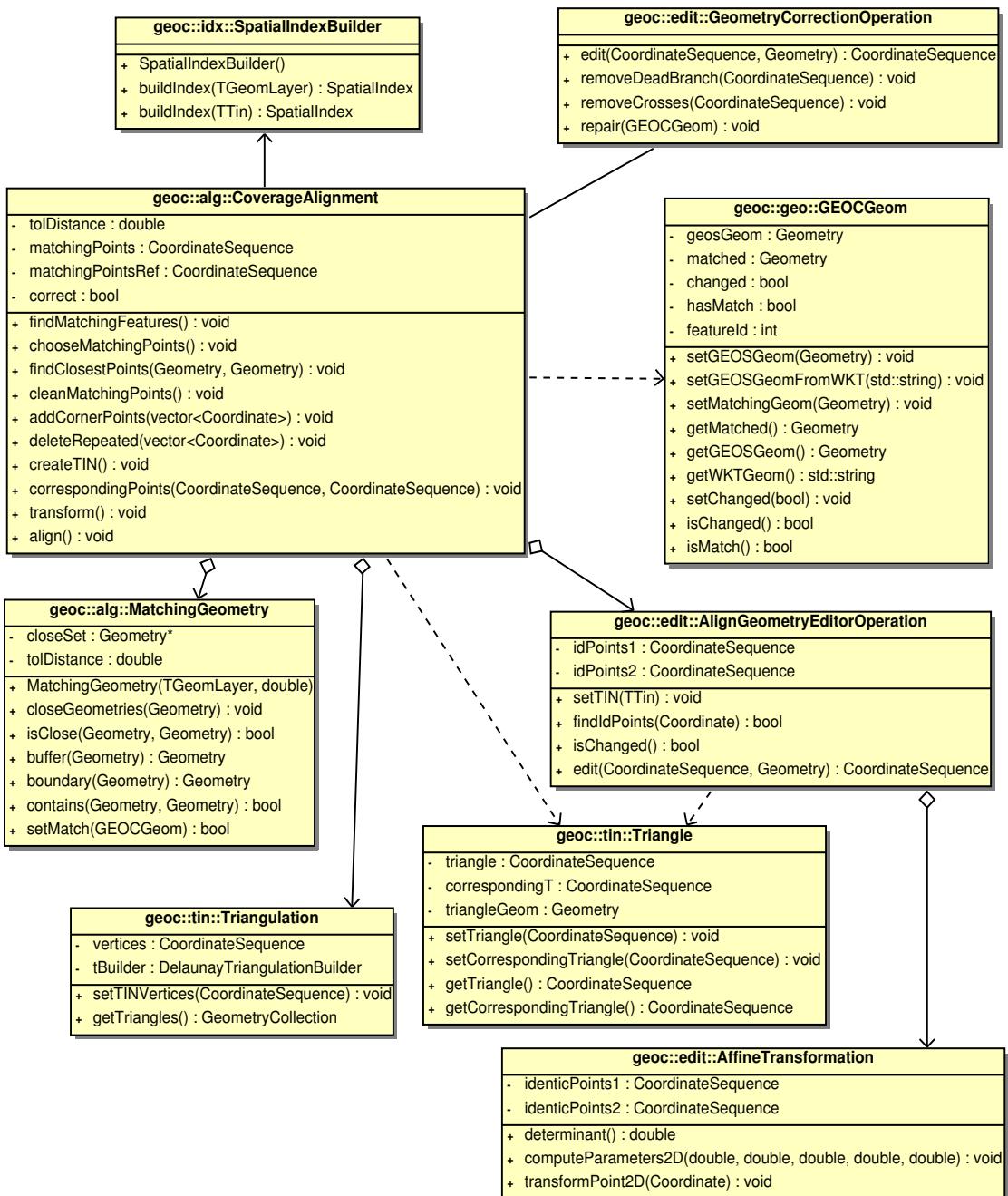
id	polygony		linie	
	110 m	1 000 m	1 000 m	10 000 m
1	1.05	1.82	1.35	6.99
2	1.13	1.78	1.35	6.98
3	1.14	1.66	1.20	7.72
4	1.04	1.65	1.36	7.83
5	1.04	1.79	1.24	7.04
6	1.03	1.79	1.21	7.77
7	1.06	1.87	1.32	7.73
8	1.12	1.84	1.21	6.98
9	1.02	1.8	1.22	7.75
10	1.14	1.76	1.33	7.75
průměr	1.08	1.78	1.28	7.45

B UML diagramy tříd

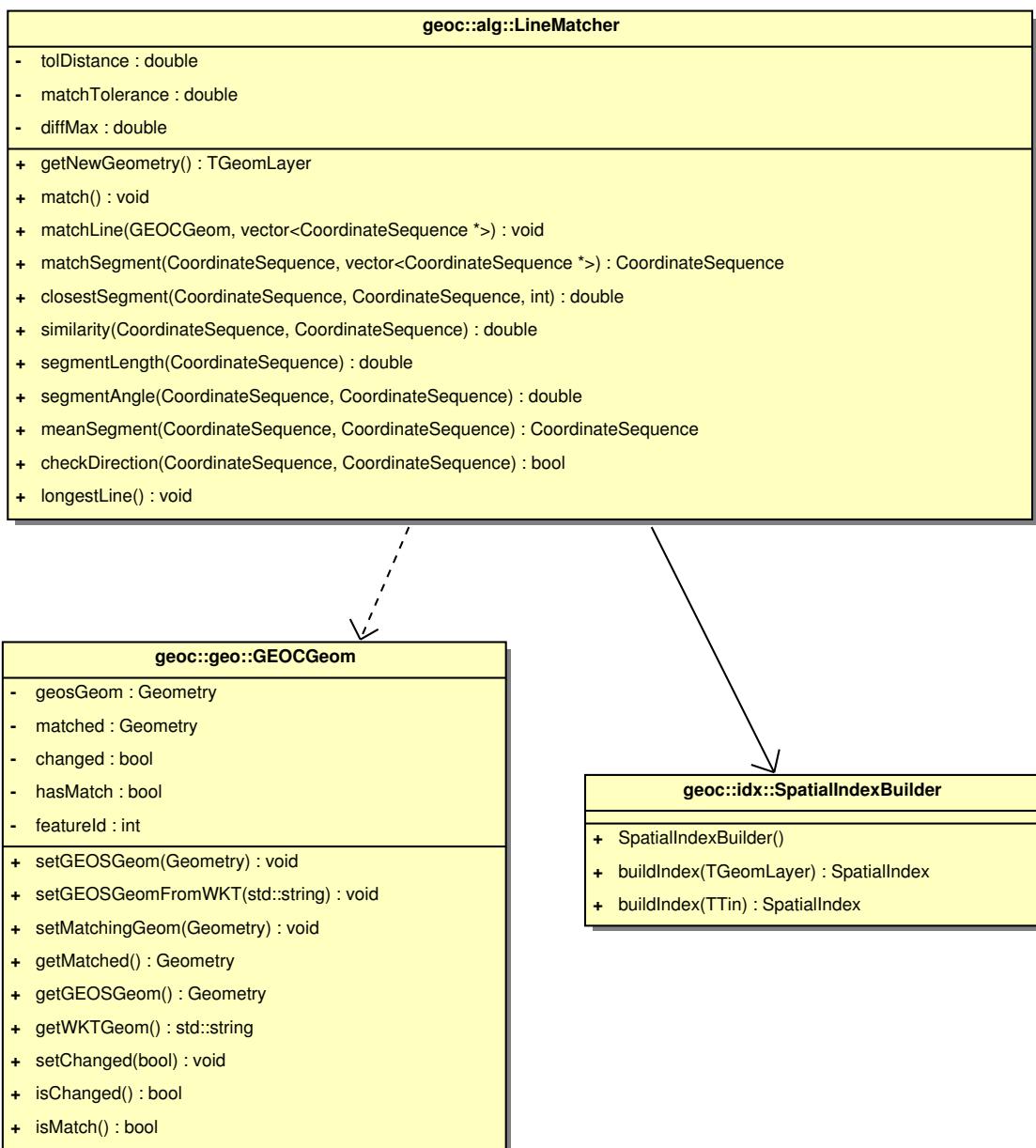
Následující UML diagramy zobrazují třídy spojené s jednotlivými algoritmy a jejich nejdůležitější atributy a metody.



Obrázek B.1: UML diagram tříd pro VertexSnapper



Obrázek B.2: UML diagram tříd pro CoverageAlignment



Obrázek B.3: UML diagram tříd pro LineMatcher

C Uživatelská příručka

Tato příručka je psána pro použití modulu **Conflate** v Quantum GIS 1.9.0 , v jiných verzích se může způsob načtení modulu, popř. i jiné činnosti mírně lišit.

C.1 Načtení zásuvného modulu

Načtení zásuvného modulu lze provést ve Správci zásuvných modulů.

Zásuvné moduly → Spravovat zásuvné moduly... (Plugins → Plugin Manager...)

Zde je třeba zaškrtnout *Conflate Plugin*. Po provedení tohoto kroku by se měla objevit ikonka modulu v nástrojové liště a také v menu *Zásuvné moduly*.



Obrázek C.1: Ikonka zásuvného modulu

Pokud se plugin nezobrazuje ve Správci zásuvných modulů, je třeba nastavit cestu k souboru *.so* v *Nastavení*.

*Nastavení → Volby → Systém → Cesty k zásuvným modulům
(Settings → Options → System → Plugin paths)*

C.2 Spuštění a nastavení dialogu

C.2.1 Výběr vstupních vrstev

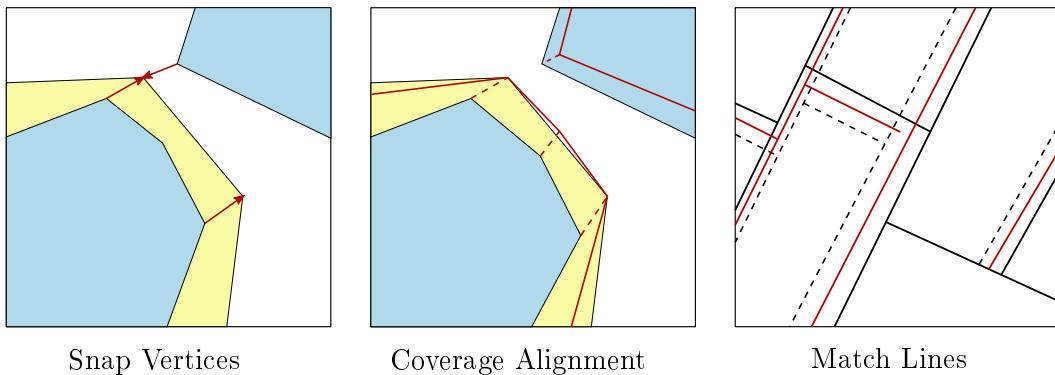
Před spuštěním dialogu *Conflate* je třeba mít v aktuálním projektu načteny vrstvy, které chceme zpracovávat. Pokud přidáme vrstvu až po otevření dialogu, lze ji načíst do výběru vrstev tlačítkem *Refresh*. Po otevření dialogu je třeba provést výběr **referenční vrstvy** (*Reference Layer*) a **upravované vrstvy** (*Subject Layer*). Referenční vrstva je obvykle ta s vyšší přesností, která se nebude měnit. Upravovanou vrstvu naopak chceme zarovnat k vrstvě referenční. Obě vrstvy by měly být stejného geometrického typu (polygon - polygon, linie - linie, bod - bod).

C.2.2 Metoda zpracování

Dalším krokem je volba způsobu zpracování (*Select the way of conflation*). Na výběr jsou tyto metody, jejichž princip je naznačen na obrázku C.2.

- **Přichycení vrcholů** (*Snap Vertices*) - tato metoda vyhledá blízké vrcholy z obou datasetů a změní polohu bodů upravované vrstvy tak, aby odpovídala poloze blízkých bodů vrstvy referenční.

- **Zarovnání vrstev** (*Coverage Alignment*) - princip této metody je složitější. Na rozdíl od předchozí metody nepracuje s jednotlivými body, ale s celými prvky. Upravuje i některé prvky, k nimž neexistují žádné odpovídající ve vrstvě referenční, a to na základě změny okolních prvků. Je proto vhodnější zejména pro datasety, které mají rozdílný počet prvků. Obecně je možné s touto metodou dosáhnout přesnějších a často reálnějších výsledků, avšak na úkor času zpracování.
- **Napasování linií** (*Match lines*) - tato metoda vyhledá odpovídající si úseky linií ze dvou různých vrstev. Takto nalezené páry zprůměruje a vytvoří z nich nové linie. Při volbě tohoto způsobu zpracování nezáleží na tom, která vrstva je referenční a která upravovaná. Napasování linií lze použít pouze na dvojice liniových vrstev.



Obrázek C.2: Princip jednotlivých algoritmů

C.2.3 Další nastavení

Poté je třeba nastavit **toleranční vzdálenost** (*Distance Tolerance*) v jednotkách projektu. Tato vzdálenost udává, v jaké maximální vzdálenosti mohou být odpovídající si prvky z obou vrstev, a jak moc se tedy může cílová vrstva měnit. V ideálním případě by tato vzdálenost neměla přesahovat nejkratší segment geometrie všech prvků upravované vrstvy (tzn. nejkratší úsek linie nebo nejkratší stranu polygonu). Nevhodná volba této vzdálenosti může vést ke vzniku nevalidních geometrií či nereálným výsledkům.

U poslední metody (*Match Lines*) se zviditelní ještě možnost nastavení **kritéria podobnosti** (*Matching criterium*). Jedná se o minimální podobnost dvou segmentů, která musí být dodržena pro jejich spárování. Tato hodnota se udává v procentech. Při zadání 100% jsou výsledkem pouze naprostotožné segmenty. Kritérium podobnosti se počítá na základě úhlu mezi segmenty, vzdálenosti jejich koncových bodů a rozdílu jejich délek.

Volba toleranční vzdálenosti a metody zarovnání může výrazně ovlivnit rychlosť zpracování, je proto doporučeno volit raději vždy menší vzdálenost a jednodušší metodu a teprve po zobrazení výsledků případně parametry změnit.

Poslední nastavovanou možností je **automatická oprava geometrie** zaškrtnutím *Try to repair invalid geometries*. Při této volbě se program pokusí opravit nově vzniklé nevalidní geometrie. Opravováno je pouze křížení úseků v polygonu a tzv. „slepé větve“. Automatická oprava může ovlivnit přesnost výsledku a někdy i vytvořit topologické chyby, proto je třeba ji používat opatrně. Při volbě *Match Lines* není tato možnost dostupná, jelikož uvedené situace nejsou u liniových geometrií považovány za chybné.

C.2.4 Výstupní vrstva

Před spuštěním zpracování lze ještě nastavit název a cestu, kam se má uložit výsledný soubor. Upravená vrstva se vždy uloží jako nová vrstva ve formátu *shapefile*. Do pole *Output shapefile* lze zadat **cestu k výstupnímu souboru**, popřípadě ji vybrat pomocí tlačítka *Browse (Procházet)* vpravo.

Pokud zadáte pouze název výstupní vrstvy nebo toto pole necháte prázdné, uloží se nová vrstva do aktuálního adresáře pod daným názvem nebo pod názvem upravované vrstvy s příslušným pořadovým číslem.

Po nastavení všech parametrů se stisknutím tlačítka *Process* spustí zpracování.

C.3 Výsledek

Upravená vrstva se automaticky přidá do otevřeného projektu v QGISu.

V dialogu zásuvného modulu se do textového pole vlevo vypíše protokol o zpracování. Formát protokolu je popsán na obrázku C.3.

Výsledkem zpracování mohou být i nevalidní geometrie, jejichž identifikátory lze nalézt v protokolu. Proto jsou často nezbytné ještě závěrečné úpravy s využitím editačních nástrojů QGISu.

```

Conflation protocol

reference layer:          vyber_obce
subject layer:            vyber_CR
new layer:                vyber_CR(4)
distance tolerance:       500
conflation method:        Snap Vertices

number of processed features: 94
number of invalid features: 11

ids of invalid features (needs to be repaired manually):
6
18
22
45
66
68
69
73
75
76
89

NOTE: The high number of invalid features can be caused by too
large tolerance distance or by combining two layers with very
different precision. Please try conflation again with different
settings or check if data are topologically correct.

end of protocol

```

reference layer: vyber_obce
 subject layer: vyber_CR
 new layer: vyber_CR(4)
 distance tolerance: 500
 conflation method: Snap Vertices

 number of processed features: 94
 number of invalid features: 11

 ids of invalid features (needs to be repaired manually):
 6
 18
 22
 45
 66
 68
 69
 73
 75
 76
 89

 NOTE: The high number of invalid features can be caused by too
 large tolerance distance or by combining two layers with very
 different precision. Please try conflation again with different
 settings or check if data are topologically correct.

 end of protocol

reference layer: vyber_obce
 subject layer: vyber_CR
 new layer: vyber_CR(4)
 distance tolerance: 500
 conflation method: Snap Vertices

 number of processed features: 94
 number of invalid features: 11

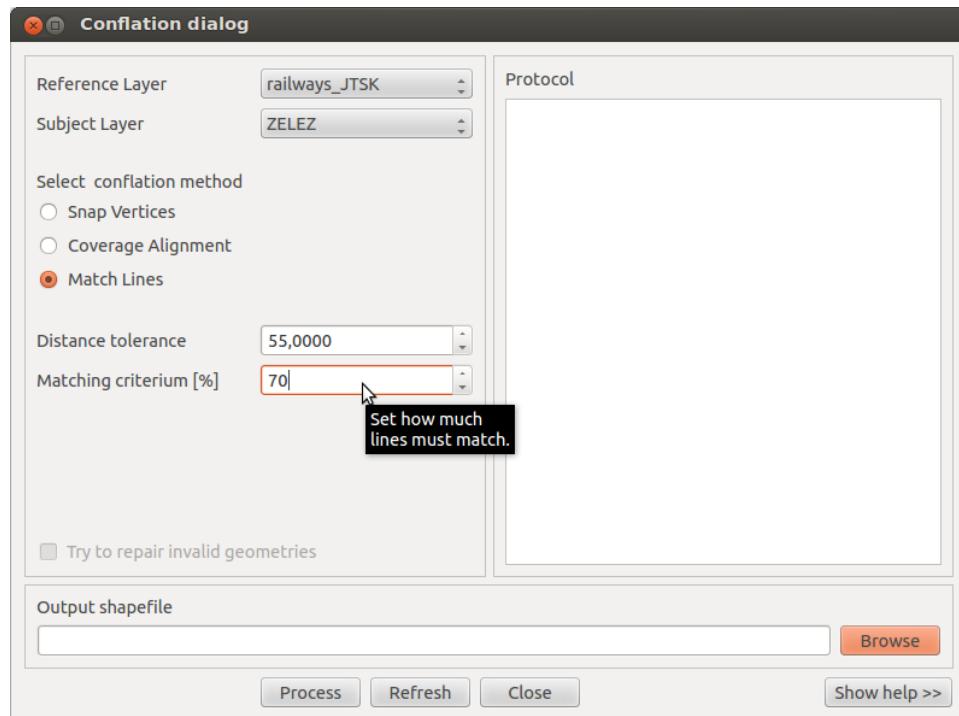
 ids of invalid features (needs to be repaired manually):
 6
 18
 22
 45
 66
 68
 69
 73
 75
 76
 89

 NOTE: The high number of invalid features can be caused by too
 large tolerance distance or by combining two layers with very
 different precision. Please try conflation again with different
 settings or check if data are topologically correct.

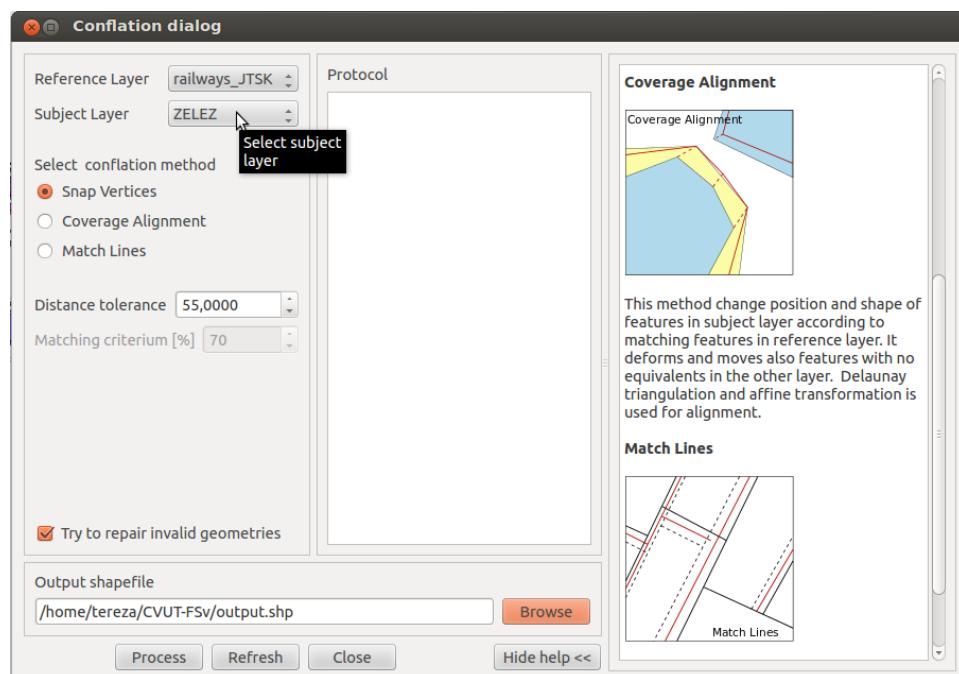
 end of protocol

Obrázek C.3: Formát protokolu

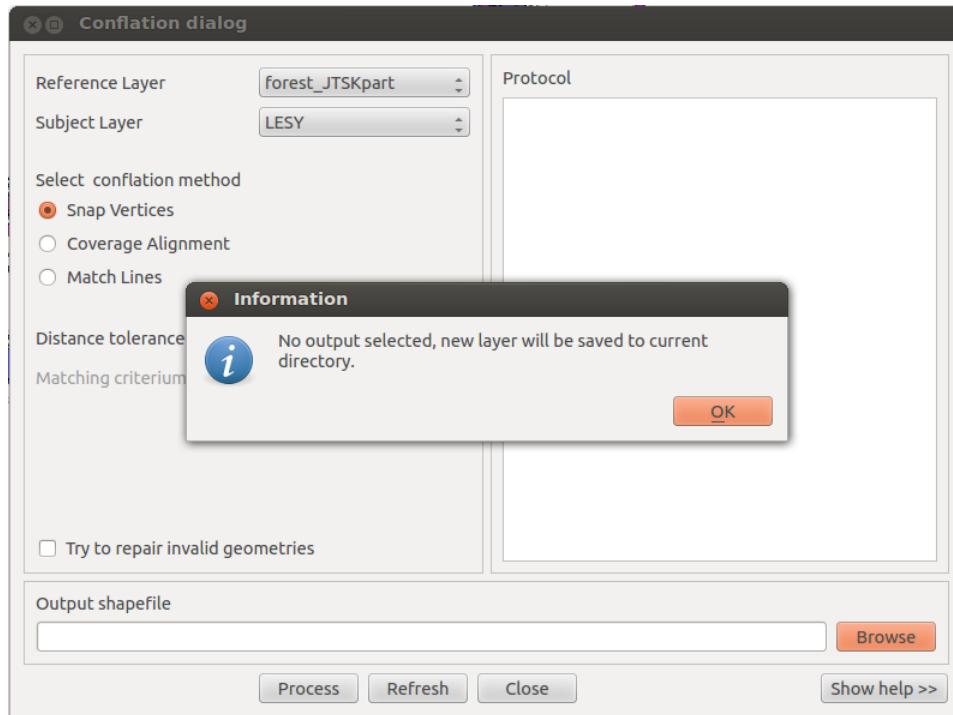
D Screenshoty zásuvného modulu



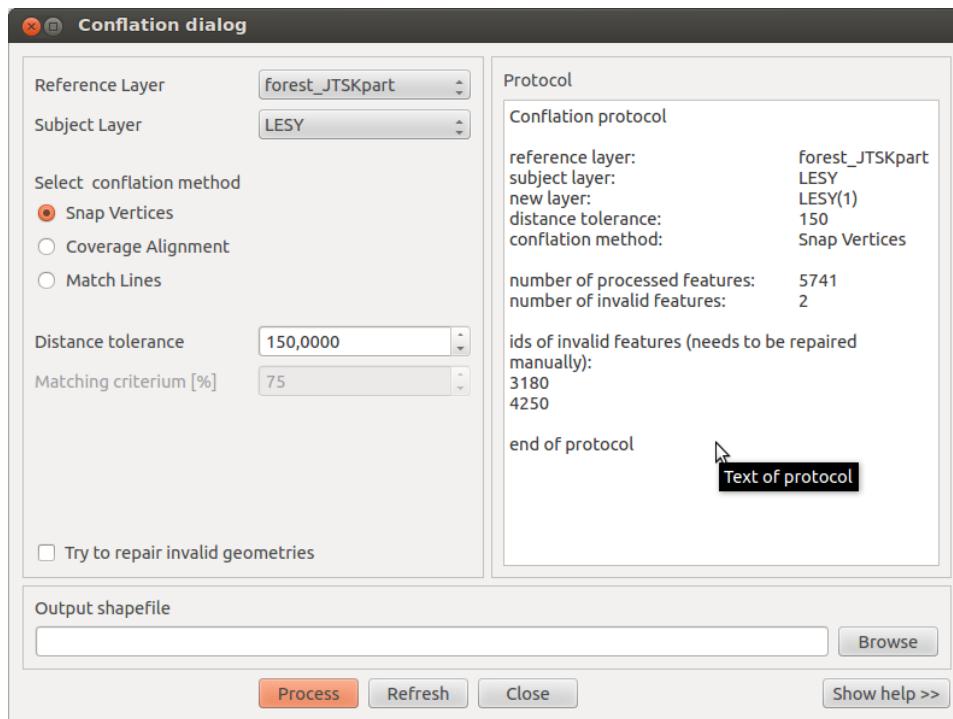
Obrázek D.1: Dialog zásuvného modulu - nastavení



Obrázek D.2: Dialog zásuvného modulu - návod



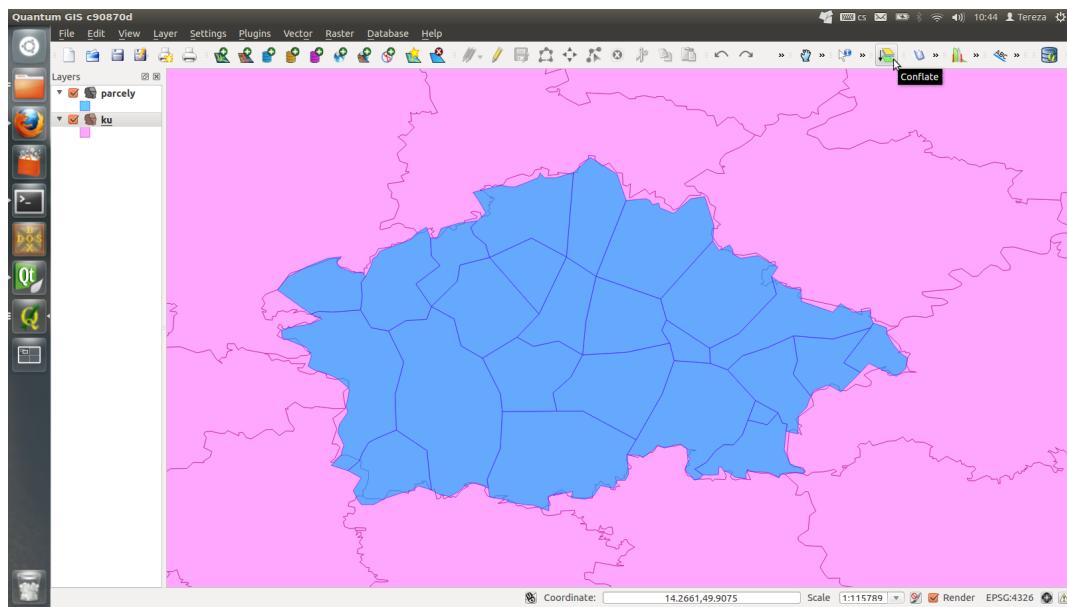
Obrázek D.3: Dialog zásuvného modulu - upozornění



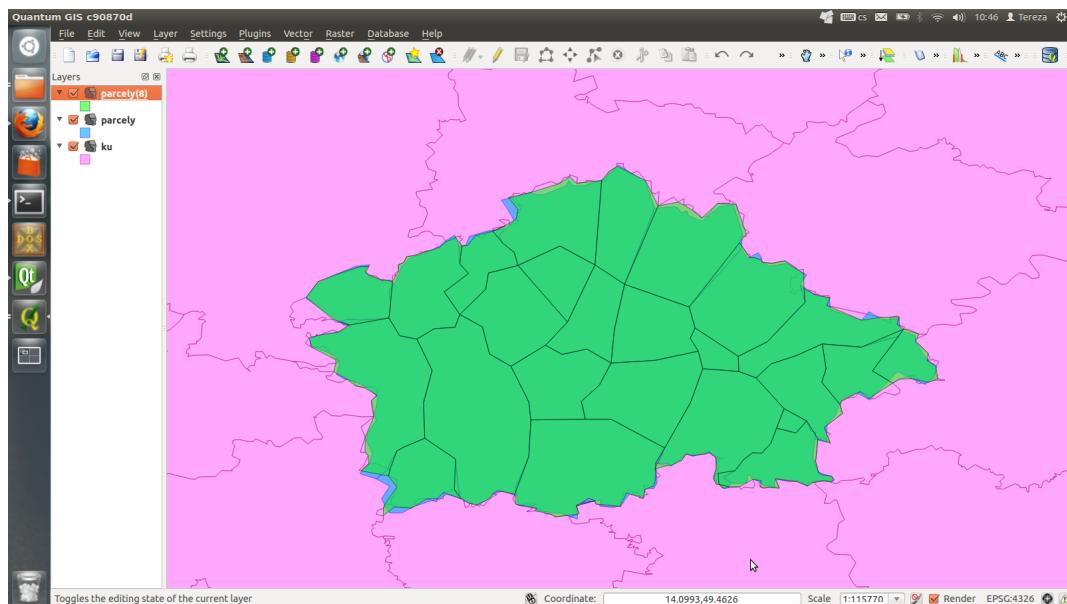
Obrázek D.4: Dialog zásuvného modulu - protokol

E Ukázky zpracování dat

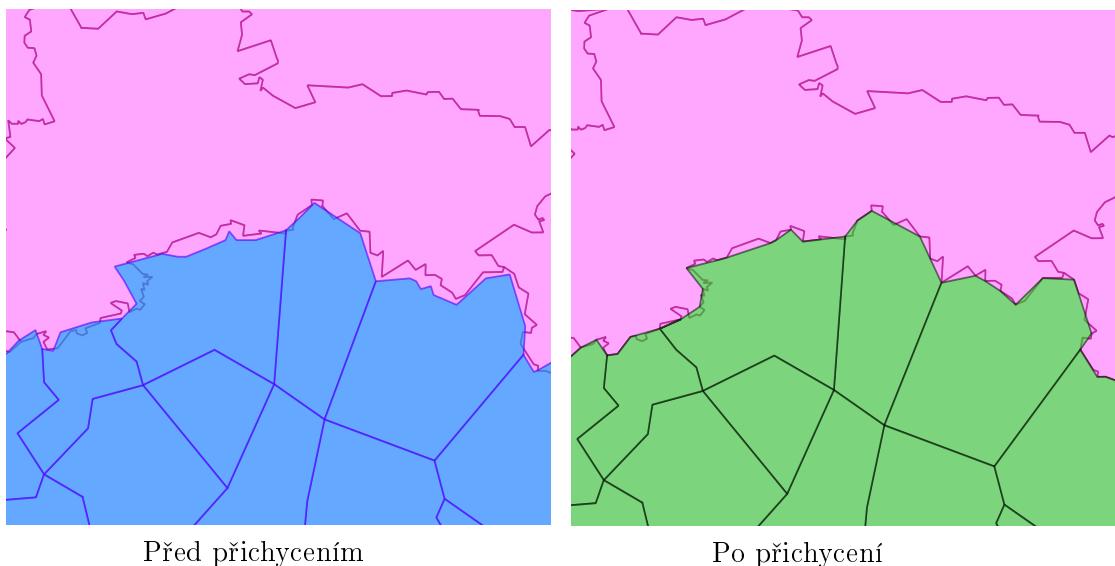
E.1 VertexSnapper



Obrázek E.1: VertexSnapper - vstupní data

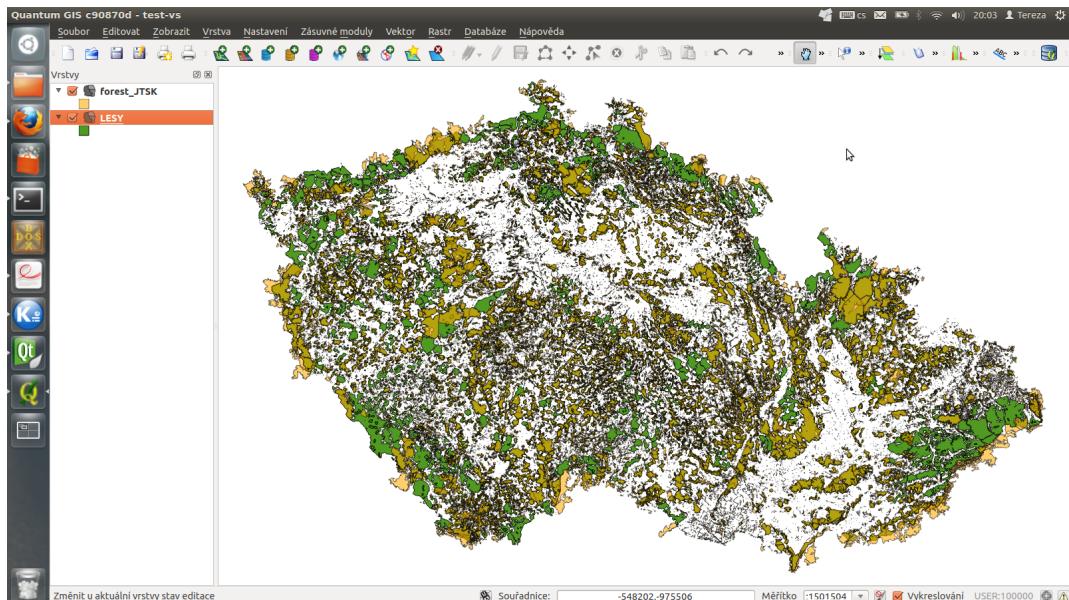


Obrázek E.2: VertexSnapper - výsledek zpracování

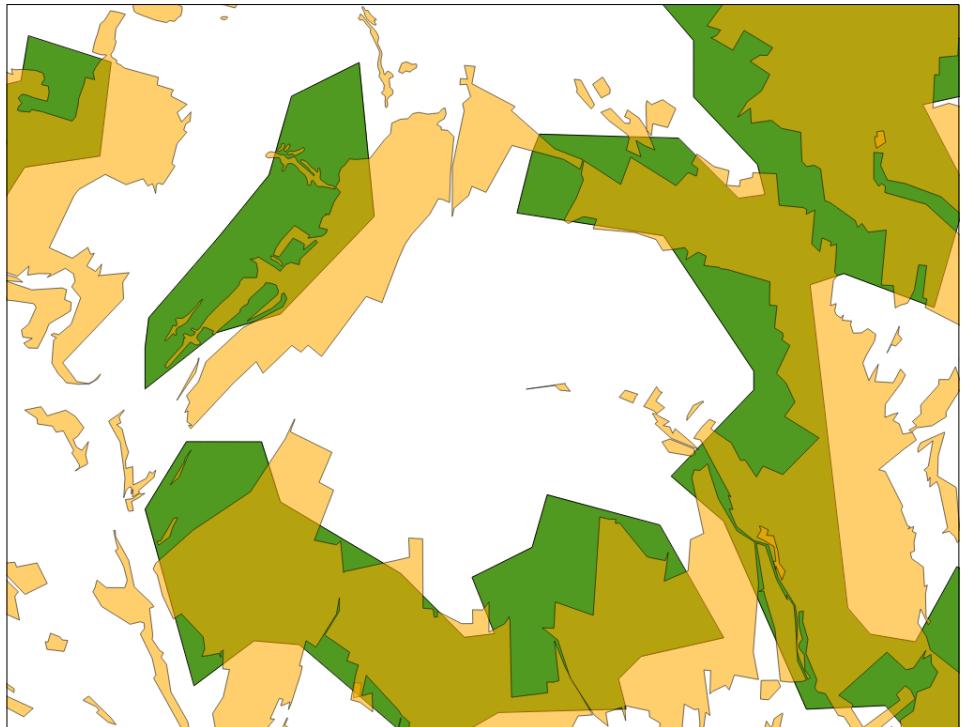


Obrázek E.3: VertexSnapper - porovnání

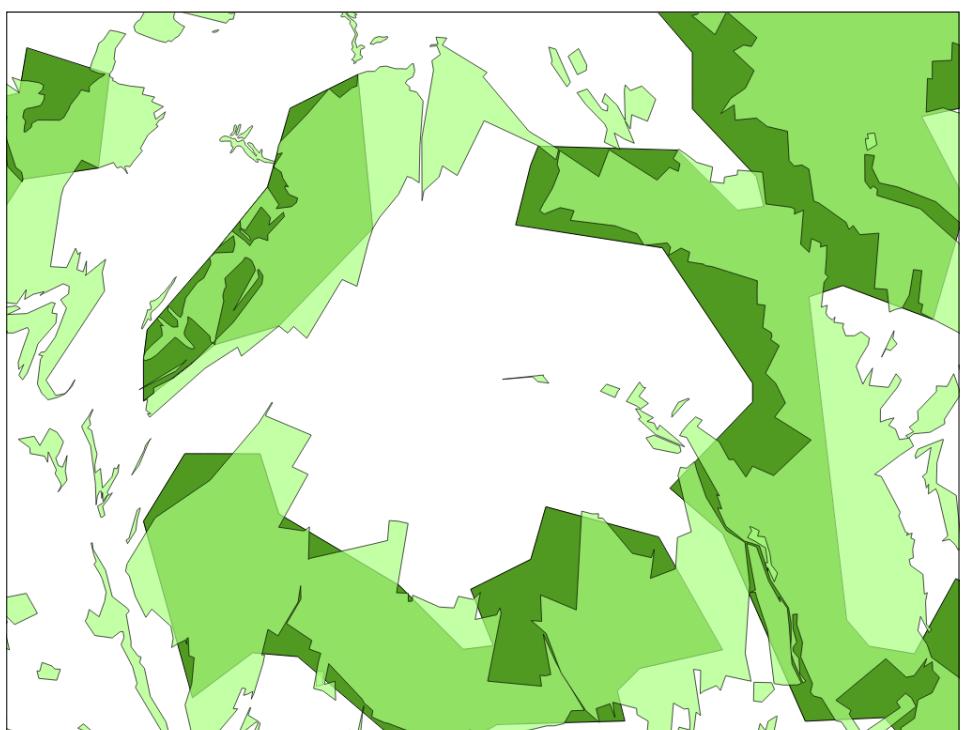
E.2 CoverageAlignment



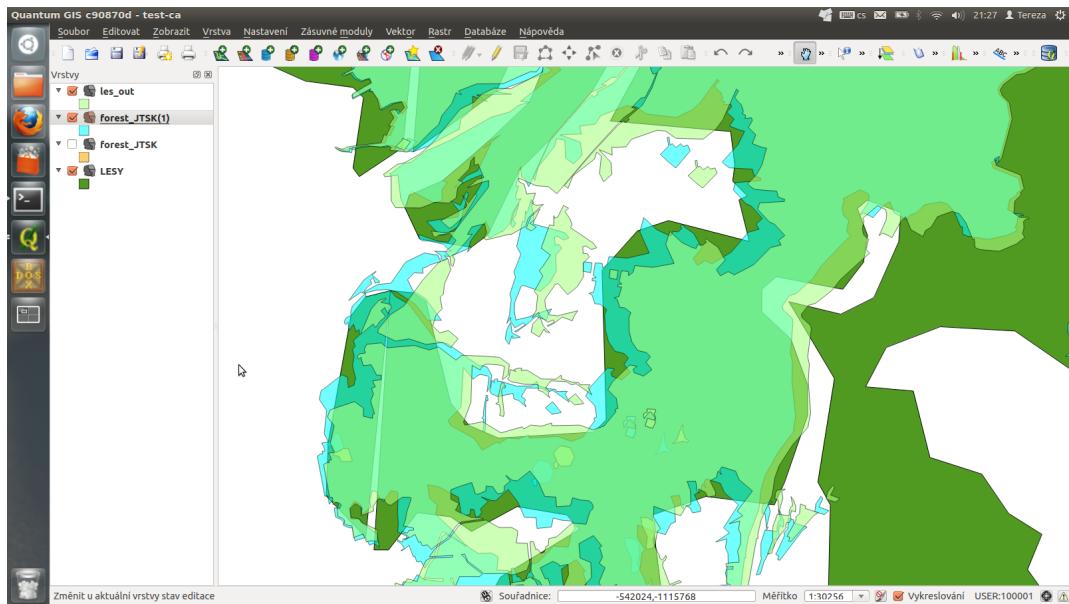
Obrázek E.4: CoverageAlignment - vstupní data



Obrázek E.5: CoverageAlignment - před zarovnáním

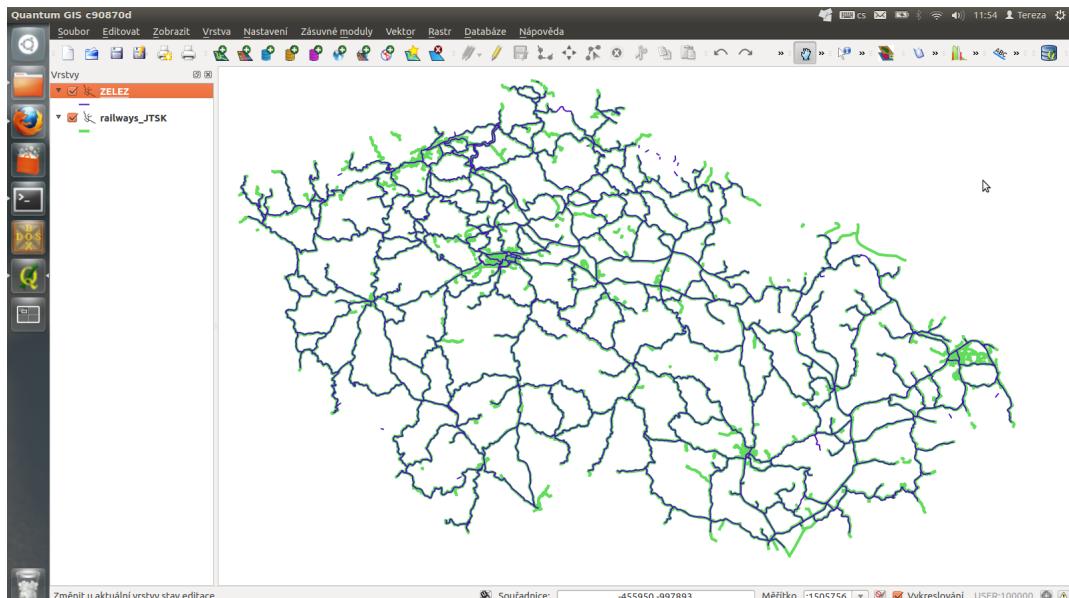


Obrázek E.6: CoverageAlignment - po zarovnání

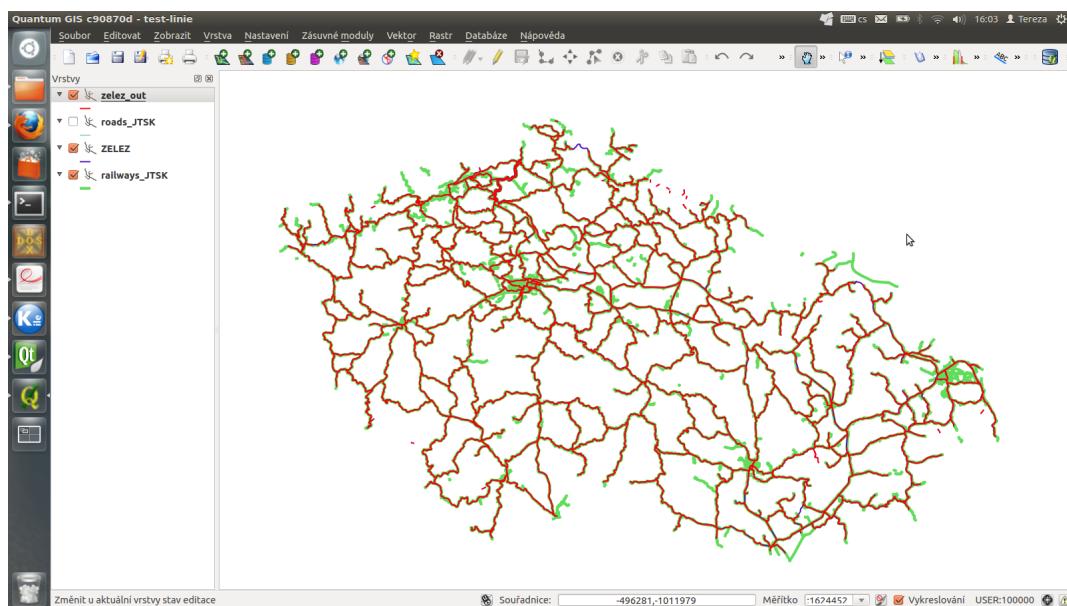


Obrázek E.7: CoverageAlignment - porovnání zarovnání s toleranční vzdáleností 600 m (zelená) a 1100 m (modrá)

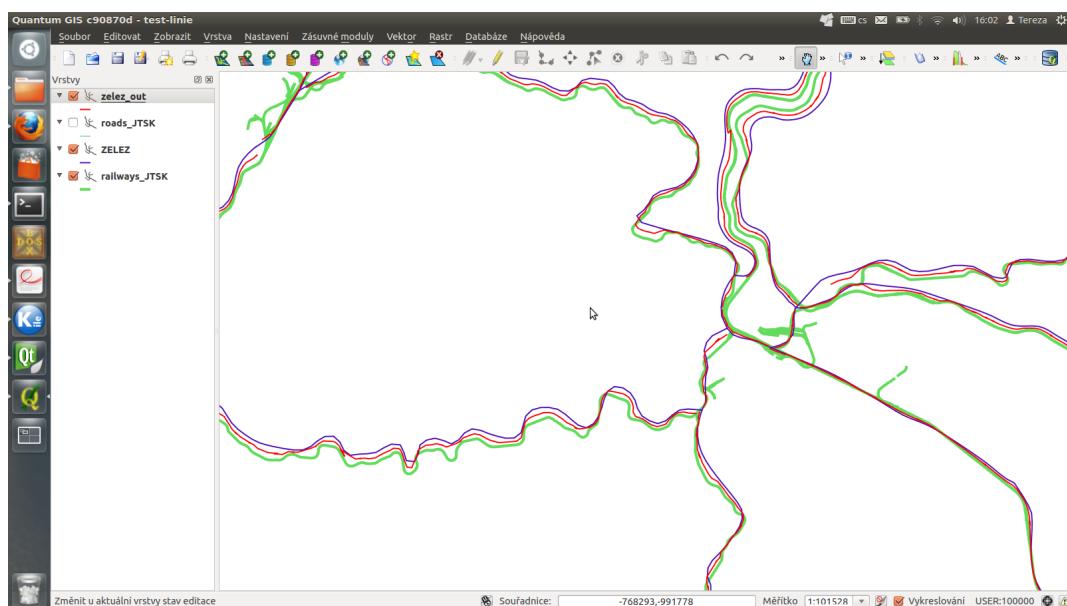
E.3 LineMatcher



Obrázek E.8: LineMatcher - vstupní data



Obrázek E.9: LineMatcher - výsledek



Obrázek E.10: LineMatcher - detail

F Obsah CD

.	
└── conflate	
├── src	zdrojové soubory pluginu
├── doc	dokumentace vygenerovaná pomocí doxygen
└── readme.txt	návod a základní informace
└── libqggsconflate.so.1.0.0	přeložená aplikace
└── geoc	
├── src	zdrojové kódy knihovny
├── doc	dokumentace vygenerovaná pomocí doxygen
└── readme.txt	návod a základní informace
└── libgeoc.so.1.0.0	přeložená knihovna
└── testing	
├── src	zdrojové soubory testovací aplikace
├── sample_data	testovací data
└── readme.txt	návod a základní informace
└── geoc_testing	testovací aplikace
└── testing.sh	bash skript pro testování
└── text	
└── latex	zdrojové soubory textu této práce
└── tereza-fiedlerova-bp-2013.xls	anotace práce
└── tereza-fiedlerova-bp-2013.pdf	tento text
└── zadani-oficialni.jpeg	naskenované oficiální zadání práce