Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Generování struktury silniční sítě na základě rozpoznávání obrazu (map)

Plzeň 2015 Jan Masopust

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 12. května 2015

Jan Masopust

Abstract

This thesis deals with generation of road network structure based on recognition of online available road maps. The first part of this paper describes the methods of image processing and pattern recognition. Selected algorithms are used to design and implement application. The outcome of the application is then the structure of the road network stored in XML format. Finally, the application is tested and the results are evaluated.

Obsah

1 Úvod

V dnešní době existuje velké množství poskytovatelů mapových podkladů, avšak naprostá většina z nich je poskytuje pouze ve formě bitmapových obrázků. Výjimkou je projekt OpenStreetMap, který poskytuje mapy i ve formě XML, kde je uložena struktura silniční sítě. Bohužel přesnost těchto materiálů je z důvodu jejich otevřenosti nedostatečná. Ostatní mapové podklady mají mnohem větší přesnost.

Proto cílem této práce je navrhnout a implementovat aplikaci, která dokáže z bitmapového obrázku z běžně dostupných on-line mapových podkladů automaticky vygenerovat strukturu silniční sítě. Je tedy nejdříve nutné seznámit se způsoby rozpoznávání obrazu a poté se způsoby převedení rozpoznaných cest do strukturované silniční sítě.

Práce je rozdělena do několika částí. Nejprve se ve kapitole 2 prozkoumají dostupné mapové podklady a se v kapitole 3 rozeberou jejich jednotlivé prvky. Poté budou následovat kapitoly 4, 5, 6 zaměřené na algoritmy pro zpracování obrazu a rozpoznávání, kde se bude uvažovat použití těchto algoritmů pro detekci cest v mapách. Následně budou probrány algoritmy pro převedení rozpoznaných cest do strukturované podoby (kapitola 7) a algoritmy pro nalezení cest, které se nepovedlo rozpoznat(kapitola 8).

Ještě před samotnou implementací bude v kapitole 9 provedena analýza jednotlivých částí programu, kde budou navrženy způsoby jejich fungování, a budou navrženy struktury vstupních a výstupních souborů. Pak se přejde k samotné implementaci aplikace (kapitola 10), kde budou nejdříve popsány použité technologie a poté bude popsán způsob implementace jednotlivých částí programu a jeho struktura. Poslední část práce se bude věnovat testování vzniklé aplikace (kapitola 11) a z výsledků bude vyvozen závěr.

2 Zdroje obrazových map

V této kapitole budou popsány jednotlivé aspekty, výhody a nevýhody nejpoužívanějších on-line dostupných mapových podkladů. Mezi ně patří Mapy.cz, Google mapy, Bing mapy a OpenStreetMap. Jedná se o tzv. obecné mapy, ty obsahují zjednodušené schématické zobrazení silniční sítě, základní rozlišení zástavby a porostu a mohou obsahovat různé body zájmu.

Jedním z důležitých aspektů map je jejich přesnost, jestli cesty někde nechybí nebo naopak nepřebývají oproti skutečnosti nebo jestli na sebe správně navazují. Dalším aspektem je potom čitelnost map (strojová čitelnost), tím je myšleno množství informací navíc, které nějak narušují rozpoznávání cest. Typickým příkladem je název ulice, který se zpravidla nachází přímo na cestě. Do čitelnosti se dá také zahrnout barevné rozlišení jednotlivcích komponent, kdy dvě různé komponenty mohou mít stejné barvy.

Porovnávání map probíhalo převážně na území České republiky a na různě osídlených oblastech, protože v malých městech se nacházejí spíše klikaté cesty s různými typy křižovatek a je v nich zpravidla málo zajímavých míst (banka, autobusová zastávka), která by se v mapách mohla také zobrazovat. Naopak ve větších městech jsou spíše rovné cesty s kolmými křižovatkami a větším množstvím zajímavých míst.

Přesnost se určovala pomocí porovnání se satelitními mapami a znalostí těchto míst autora této práce. Čitelnost map se pak určovala až v průběhu vývoje aplikace při aplikování algoritmů na předzpracování obrazu. To z toho důvodu, že žádný jednoznačný způsob určení neexistuje a lidským okem nejsou některé problémy zachytitelné, protože mapy jsou tvořené tak, aby byly lidmi snadno čitelné.

2.1 Mapy.cz

Mapy.cz je služba poskytovaná společností Seznam.cz a.s. Data pro Českou republiku a Slovensko pochází z kartografické redakce Mapy.cz a zbytek světa je generován ze zdrojů OpenStreetMap. Mapy jsou poskytovány pouze ve formě bitmapových obrázků, které je možné získat dostupným API nebo zachycením obrazovky (print screen) [1].

Velkou výhodou map je jejich přesnost, ta je velmi vysoká, alespoň pro Českou republiku, a chyby se v nich hledají jen velmi těžko.

Co se týče strojové čitelnosti, na tom jsou mnohem hůře. Vyskytuje se zde velké množství neduhů, které značně komplikují rozpoznávání cest:

Špatná barevná rozlišitelnost – přestože na první pohled mapa vypadá velmi dobře rozlišitelná viz Obr. 2.1, opak je pravdou. Na první pohled je možné vidět, že většina cest je rozlišena bílou barvou, což by bylo ideální. Bohužel bílou barvou jsou ohraničeny všechny ikony zobrazené v mapě a také všechny texty zobrazené v mapě. Navíc ve větších městech se pak stejnou barvou zobrazují i chodníky. To by nebyl takový problém, protože chodník je ve skutečnosti cesta, nicméně se často vyskytují ve velké blízkosti silnic, které jsou důležitější, a to komplikuje jejich rozpoznávání.

Velké množství zajímavých bodů – především ve velkých městech se nachází příliš velké množství ikon značících umístění zajímavých míst jako jsou autobusové zastávky, banky, muzea, kostely a mnoho dalších. Protože se tato místa zpravidla nacházejí u silnic, jejich ikony často zasahují přímo do zobrazených cest.

Parkoviště a náměstí – ty mají stejnou barvu jako většina cest, ale nejsou to cesty. Jsou většinou zobrazovány jako velké bíle polygony různých tvarů a je složité rozlišit je od obyčejných cest.

Obrázek 2.1: Ukázka mapového podkladu Mapy.cz [1].

2.2 Google mapy

Google mapy (nebo také Google maps) je internetová mapová aplikace od společnosti Google. Mapy, ve formě bitmapových obrázků, je možné stahovat pomocí API nebo zachycením obrazovky. Mapy jsou také velmi přesné, i když na rozdíl od Mapy.cz se v nich nachází o něco více chyb, ale ty jsou většinou zanedbatelné [2].

Co se týče čitelnosti v té výrazně předčí Mapy.cz. Mapy jsou velmi jednoduché a není v nich příliš mnoho dalších informací viz Obr. 2.2. Barevné rozlišitelnost je také velice dobrá, i když ikony mají stejnou barvu jako cesty, většinou se nacházejí mimo ně a je tedy snazší je odlišit. Ikon celkově je zde méně než v předchozích mapových podkladech. Další velkou výhodou je, že všechny silnice stejné barvy mají stejnou šířku, to je možné uplatnit při odstraňování chyb vzniklých při předzpracování.

Na druhou stranu v mapách se občas vyskytují cesty, které mají na první pohled stejnou barvu jako ostatní cesty, ale mají odlišný odstín. Tím pak výrazně narůstá seznam barev používaných pro vykreslování cest.

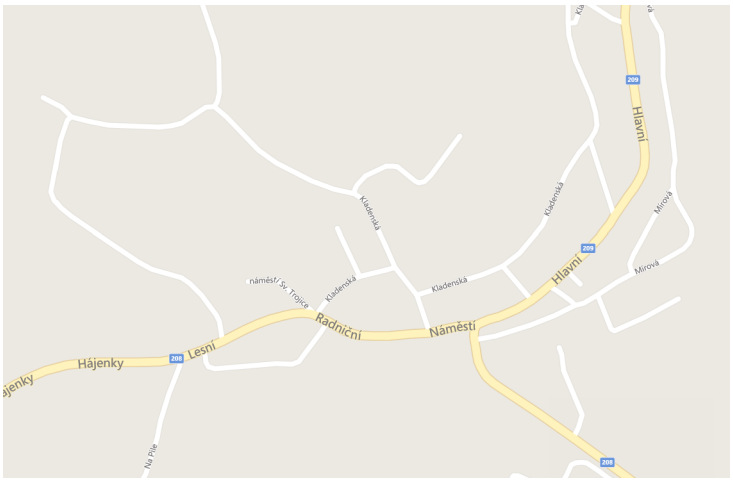
Obrázek 2.2: Ukázka mapového podkladu Google mapy [2].

2.3 Mapy Bing

Mapy Bing je webová mapová služba poskytovaná jako součást vyhledávacího nástroje Bing od společnosti Microsoft. Pro Bing mapy také existuje API, pomocí kterého je možné mapy stahovat ve formě bitmapových obrázků, ale také je tu možnost focení obrazovky [3].

Přesnost map je nižší než u předchozích map, hledání chyb už není tak obtížné. Některé vedlejší ulice, především v menších městech chybí úplně.

Na druhou stranu čitelnost map je zde nejlepší z testovaných. V mapách se vyskytuje jen velmi malé množství ikon (toto platí pro Českou Republiku, v zahraničích městech, především Amerických, je množství ikon srovnatelné s Google mapami). Barevná rozlišitelnost je také velice dobrá, jen těžko se hledají dvě různé komponenty s totožnými barvami viz Obr. 2.3.

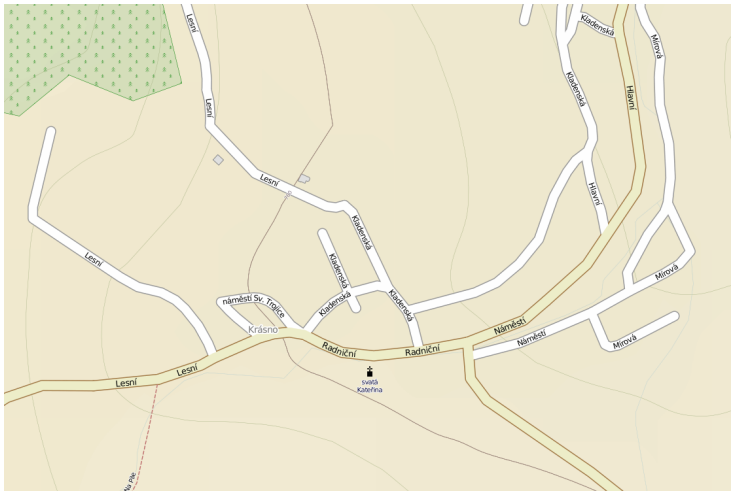


Obrázek 2.3: Ukázka mapového podkladu Bing mapy [3].

2.4 OpenStreetMap

OpenStreetMap (dále OSM) je projekt s cílem vytvořit globální volně dostupná geografická data, která budou následně vizualizována do podoby topografických map. Mapy je možné získat ve formě bitmapových obrázků stažitelných pomocí API, vyfocením obrazovky nebo je stáhnou ve formě XML [4].

Přesnost těchto map je však velice nízká, často chybí i celé ulice a často také dochází k špatnému propojování silnic. Čitelnost těchto map se velmi špatně hodnotí a to z toho důvodu, že data jsou veřejně přístupná a existuje tak větší množství vizualizací jako jsou obyčejné mapy [5] viz Obr. 2.4, cyklistické mapy [6], mapy pro lyžaře [7] a mnoho dalších. Některé z nich mají velmi dobrou rozlišitelnost barev a málo ikon a některé zase velmi špatnou rozlišitelnost a velké množství ikon.



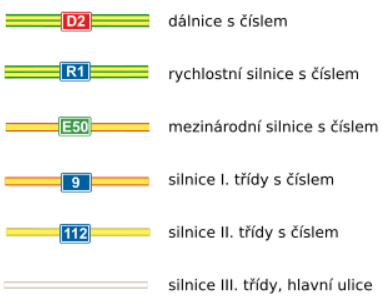
Obrázek 2.4: Ukázka mapového podkladu OSM [5].

3 Obecný popis map

Tato kapitola se bude krátce věnovat základním vlastnostem používaných map. Nejedná se o popis konkrétních vybraných mapových podkladů, nýbrž o obecný popis map tohoto typu. Obecné mapy se skládají ze zjednodušeného schématického zobrazení silniční sítě, základního rozlišení zástavby a porostu a mohou obsahovat různé body zájmu.

3.1 Silniční síť

Jelikož silniční síť je hlavním prvkem těchto map je na první pohled lidským okem detekovatelná. Cesty jsou zobrazovány jako křivky různých šířek a různých barev. Barvy potom rozlišují typ silnic. Například u mapových podkladů Mapy.cz bílá barva označuje ulice a žlutou barvou se zobrazují silnice vyšších tříd viz Obr. 3.1.



Obrázek 3.1: Část legendy mapových podkladů Mapy.cz [1].

Do silniční sítě je možné zahrnout i parkoviště a náměstí, která jsou v některých mapových podkladech také vykreslovány. Jsou většinou zobrazovány jako velké bíle polygony různých tvarů a mají stejnou barvu jako cesty viz Obr. 3.2a.

3.2 Značky

Značky mohou být také různého typy. Jedním typem značek jsou ikony naznačující polohu nějakého zajímavého místa, příkladem může být banka, autobusová zastávka nebo třeba parkoviště. Jelikož se tato zajímavá místa často nacházejí v blízkosti cest, tak jejich ikony často zasahují přímo do zobrazení silniční sítě, jak je zobrazeno na Obr. 3.2b. Vyskytují se v různých barvách a v některých případech mohou obsahovat i barvy použité pro vykreslení cest.

Dalším typem značek jsou potom jména, může se jednat o jména ulic, náměstí, zajímavých míst nebo třeba řek. V případě jmen náměstí a ulic nastává problém, protože ty jsou většinou umístěné přímo na cestách a komplikují tak jejich rozpoznávání. Některá jména, většinou jména náměstí, jsou také velmi dlouhá a překrývají velké oblasti cest (například „náměstí Milady Horákové“). Problém také je, že jména náměstí často nemají stejný sklon jako cesta, kterou popisují, jako na Obr. 3.2c.

(a) Náměstí a parkoviště (b) Ikony (c) Nápisy

Obrázek 3.2: Ukázka náměstí a typů značek. Zdroje [1] [2]

3.3 Pozadí

Pozadím je myšleno vše, co není značka nebo cesta. Jsou to velké oblasti značící výskyt porostu nebo naopak výskyt zástavby. Zpravidla nějak nezasahují do silniční sítě a není tedy obtížné je detekovat a odstranit.

4 Předzpracování

V této kapitole budou popsány techniky, které bylo možné použít pro prvotní oddělení cest, značek a pozadí. Protože Google mapy, jakožto primární zdroj dat (viz dále), mají velmi dobrou barevnou rozlišitelnost (různé komponenty mají různé barvy), pracují všechny následující techniky pouze s barvami. Ne všechny z následujících technik byly použity ve finální aplikaci, ale všechny byly vyzkoušeny, aby mohla být nalezena ta nejvhodnější.

Důvodem, proč byly nejdříve testovány složitější metody jako je K-means je, že tyto metody vyžadují menší interakci s uživatelem a jsou tak „více automatické“.

4.1 K-means

K-means je algoritmus provádějící shlukování (clustering). Shlukování je metoda rozdělující množinu dat do daného počtu skupin tak, že data s podobnými vlastnostmi jsou ve stejné skupině. Tato technika patří do skupiny algoritmů strojového učení a v tomto případě se jedná o učení bez učitele, to znamená, že podoba výsledných skupin není před spuštěním algoritmu známá [8].

K-means používá iterativní procedury pro rozdělení dat do shluků. K tomu potřebuje na vstupu zadat počet skupin, do kterých se má rozdělit, a počáteční střední hodnoty (tím je myšleno střední hodnoty skupin). Pokud je tedy požadavek rozdělit data do K skupin je nutné na vstupu zadat K středních hodnot [8].

Výsledkem algoritmu je poté K středních hodnot, odtud pochází název, které značí střední hodnoty rozpoznaných skupin. Každá položka dat poté paří do jedné skupiny, kterou lze určit prohledáváním středních hodnot a nalezením skupiny s nejbližší střední hodnotou. Za položky se v tomto případě považují jednotlivé pixely obrázku [8].

4.1.1 Měření vzdálenosti

Pro měření vzdáleností mezi jednotlivými složkami dat a středními hodnotami se používají různé techniky. Nejpoužívanější metriky, pro měření této vzdálenosti jsou Manhattanská metrika a Euklidovská vzdálenost [8].

V tomto případě byla použitá Euklidovská vzdálenost, to je odmocnina ze sumy čtverců rozdílu stejných dimenzí položky a střední hodnoty. Barvy je možné rozdělit do tří dimenzí: červená, zelená a modrá. Výpočet vzdálenosti dvou barev je zobrazen v rovnici 4.1, kde R je hodnota červené, G hodnota zelené a B hodnota modré [8].

$$d = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \tag{4.1}$$

4.1.2 Počáteční střední hodnoty

Existuje několik metod, používaných pro určení počátečních středních hodnot. Někdy se preferuje výběr náhodných prvků z množiny dat a někdy zase vytvoření nových náhodných prvků, v tomto případě byl použitý výběr náhodných prvků z množiny dat [8].

Velkým problémem je však určit správný počet skupin. V případě map je to nemožné, protože i když bude použit jeden typ map jako zdrojová data, nikdy není zaručeno, že se v něm budou vyskytovat všechny známé barvy, to by poté vedlo k rozpadu jiných skupin. Například pokud by mapa byla složená ze tří barev: bílá pro cesty, černá pro pozadí a modrá pro označení autobusových zastávek. Potom v případě, že by se v dané oblasti nenacházela žádná zastávka, rozpadla by se jedna ze dvou zbývajících skupin (to je jedním z důvodů proč nebyla metoda nakonec použita).

4.1.3 Algoritmus

Jak bylo zmíněno výše K-means je iterativní metoda, ta každou iteraci modifikuje umístění středních hodnot tak, aby byly blíže k finálním středním hodnotám.

Počáteční střední hodnoty jsou použity k přiřazení položek do jednotlivých počátečních skupin. Z počátku jsou tedy položky rozmístěny do skupin

s nejbližší střední hodnotou. To je první iterace algoritmu. Když jsou všechny položky přiřazeny ke skupinám, provede se přepočítání všech středních hodnot všech skupin. Tyto nové střední hodnoty by měly být blíže k finálním středním hodnotám. Dále se pak všechny položky znovu přiřadí do skupin s nově vypočítanými středními hodnotami. S velkou pravděpodobností se některé položky přesunou do jiných skupin [8].

Jednotlivé kroky algoritmu:

- Určení počátečních středních hodnot.
- Přiřazení položek do skupin s použitím počátečních středních hodnot.
- Prováděj dokud se položky přemisťují do jiných skupin:
  - Přepočítání středních hodnot skupin podle jejich položek.
  - Přiřazení položek do skupin s použitím nově spočítaných středních hodnot.
- Konec cyklu.

Existují speciální případy, ve kterých se cyklus nemusí zastavit, tyto případy se musí zvlášť ošetřit [8].

4.2 Jednoduchý výběr barev

Jednoduchý výběr barev je metoda, pracující, jak už název napovídá, na jednoduchém principu. U každého pixelu v obrázku rozhodne, zda se jedná o pixel znázorňující cestu. Pokud je pixel vyhodnocen jako součást cesty, je označen jako popředí (později označen číslem „1“), v opačném případě je pixel označen jako pozadí (později označen číslem „0“).

Vstupním parametrem této metody musí být seznam barev, které reprezentují právě cesty, na jehož základě se rozhoduje, zda bude pixel identifikován jako cesta nebo jako pozadí. Protože mapové podklady bývají často různě stínované, vyhlazované a plné přechodů, cesta nemusí být složena přímo z jedné barvy, ale z několika velmi podobných barev. Tyto barvy nejsou často na první pohled lidským okem viditelné. Aby nebylo nutné je všechny hledat

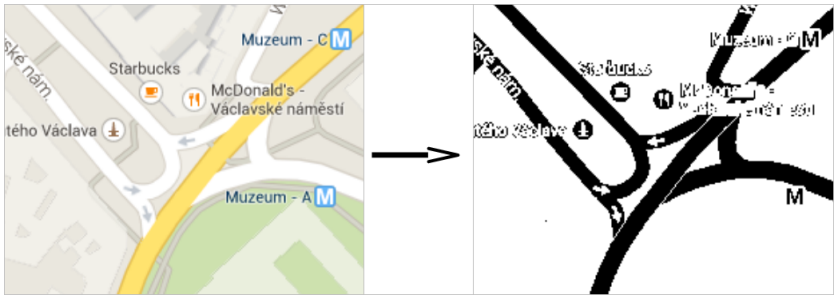
a poté zadávat na vstupu, je zavedena hodnota rozsah barev, kdy se u každé zadané barvy určí její maximální vzdálenost.

Následně se při rozhodování, zda je pixel cesta nebo pozadí, použije vzoreček 4.1, kterým se určí vzdálenost mezi barvou aktuálního pixelu a barvou na vstupu. Pokud je tato vzdálenost menší než rozsah barvy na vstupu, je pixel označen jako cesta. Jednotlivé kroky algoritmu jsou následující:

- Pro všechny pixely v obrázku proveď:
  - Nastav pixel v jako pozadí.
  - Pokud měl původní pixel barvu podobnou některé z uvedených barev cest, nastav ho jako cestu.

To, že metoda potřebuje na vstupu seznam barev cest, odebírá programu na automatizovanosti, je však důležité se uvědomit, že tento seznam je nutné vytvořit pouze jednou, pro jeden typ mapových podkladů. Pokud se tedy jednou vytvoří seznam barev, pro dané mapové podklady, při příštím rozpoznávání stejných mapových podkladů, stačí jen tento seznam použít.

Na Obr. 4.1 je zobrazeno použití této metody. Je vidět, že mimo cest byly jako cesty rozpoznány i některé značky.



Obrázek 4.1: Ukázka jednoduchého výběru barev. Zdroj [2]

4.3 Složitý výběr barev

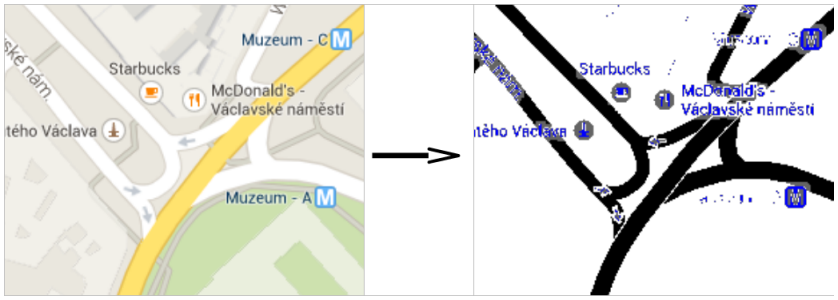
Složitý výběr barev pracuje na podobném principu jako předchozí metoda. U každého pixelu v obrázku rozhodne, zda se jedná o pixel znázorňující cestu, pozadí nebo značku. Pokud je pixel vyhodnocen jako součást cesty, je označen

jako popředí typu cesta, pokud je vyhodnocen jako značka nebo okolí značky, je označen jako popředí typu značka (později označen číslem „2“), jinak je pixel označen jako pozadí.

Navíc je zde přidán další vstupní parametr a tím je seznam barev značek a velikost jejich okolí. Barva značky má stejný význam jako u barev cest a velikost okolí značí, do jaké vzdálenosti od pixelu budou pixely s barvou cesty označeny jako pixely značky. Snaha je tedy o to, odlišit značky a jejich nejbližší okolí od cest. Jednotlivé kroky algoritmu jsou následující:

- Pro všechny pixely v obrázku proveď:
  - Nastav pixel v jako pozadí.
  - Pokud měl původní pixel barvu podobnou některé z uvedených barev cest, nastav ho jako cestu.
  - Pokud byl pixel v předchozím kroku obarven jako cesta proveď:
    - Pokud má některý pixel v jeho okolí barvu podobnou některé z uvedených barev značek, nastav ho jako okolí.
  - Jinak:
    - Pokud má pixel barvu podobnou některé z uvedených barev značek, nastav ho jako značku.

Na Obr. 4.2 je vidět, že všechny značky jsou již barevně odlišeny od cest. Modrou barvou jsou zobrazeny značky a šedou jejich okolí.



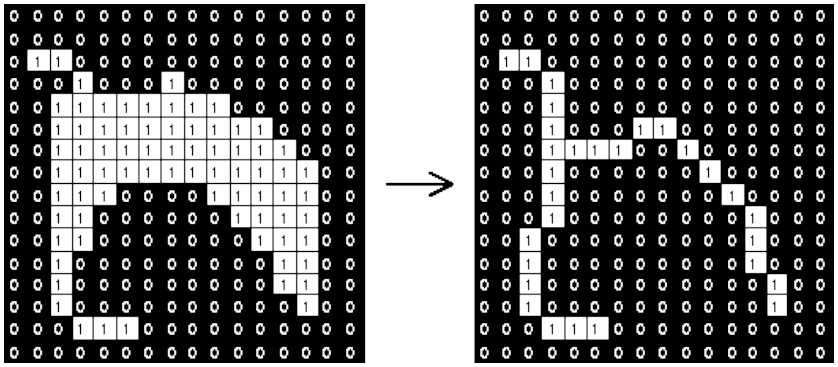
Obrázek 4.2: Ukázka složitého výběru barev. Zdroj [2]

5 Ztenčování

V této kapitole budou nejdříve popsány prostředky pro pochopení ztenčování a poté bude popsána jedna z metod ztenčování. Přestože bylo testováno více algoritmů ztenčování, je popsán pouze jeden, protože jejich popisy jsou si často podobné a jejich popis není v této práci klíčový. Prostředky pro ztenčování se pak dále využijí při popisu metod k odstraňování šumu.

5.1 Úvod do ztenčování

Ztenčování (anglicky thinning) je morfologická operace používaná k odstraňování vybraných pixelů z binárního obrázku podobně jako dilatace a eroze. Může být použito pro různé účely, ale nejčastěji se používá pro skeletonizaci. Skeletonizace je proces pro redukci oblastí označených jako popředí na jejich kostru, při zachování rozsahu a propojení původních oblastí. Ztenčování se většinou aplikováno na binární obrázek a výstupem je pak také binární obrázek. Obr. 5.1 ukazuje výsledek ztenčování na jednoduchém binárním obrázku [9].



Obrázek 5.1: Příklad ztenčování. Zdroj [9]

Jako jiné morfologické operace i ztenčování potřebuje dvě části vstupních dat. Jednou z nich je binární obrázek, ve kterém se má operace provést, a druhou je strukturní element, který určuje efekt metody na zadaný obrázek [9].

Algoritmy ztenčování mohou být rozděleny do dvou základních skupin a

těmi jsou iterativní a neiterativní metody. Neiterativní metody často bývají rychlejší, ale neposkytují tak dobré výsledky jako metod iterativní. Zde budou popsány pouze metody iterativní, jelikož je vyžadována co možná nejlepší přesnost.

5.1.1 Matematická morfologie

Obor matematické morfologie v sobě ukrývá širokou škálu metod používaných při zpracování obrazu. Byla vytvořena v roce 1964 a zasloužili se o to francouzští matematici Georges Matheron a Jean Serra. Jádro tohoto oboru vychází z pojmů teorie množin. Často se její součásti aplikují na binární obrázky a používá se například v následujících oborech:

- detekce hran,
- odstraňování šumu,
- skeletonizace,
- vylepšování obrazu,
- segmentace obrazu.

Základními součástmi matematické morfologie jsou dilatace a eroze viz kapitoly 6.2 a 6.3. Vstupem matematické morfologie jsou často dvě množiny. Jenda z nich představuje obrázek a její počáteční hodnota je zpravidla v levém horním rohu. Druhou množinou je potom strukturní element [9].

5.1.2 Strukturní element a okolí bodu

Strukturní element je množina souřadnic bodů, která je často reprezentována binárním obrazem. Od souřadnic vstupního obrázku se liší především tím, že je zpravidla menší a její počáteční hodnota se vetšinou nachází uprostřed, jedná-li se o elementy velikosti 3x3. Proto mohou některé jeho souřadnice nabývat i záporných hodnot [9].

Nejčastěji používaní zástupci jsou vyobrazeni na Obr. 5.2, kde vlevo je vyobrazen zástupce ve tvaru čtverce o velikosti 3x3 a vpravo je zástupce ve

tvaru kříže o velikosti také 3x3. Strukturální elementy však mohou mít různé tvary a rozměry a nemusejí být vůbec symetrické [9].

1	1	1
1	1	1
1	1	1

(a) Tvar čtverec

0	1	0
1	1	1
0	1	0

(b) Tvar kříž

Obrázek 5.2: Příklady strukturních elementů

Při aplikaci některé z morfonologický operací je počátek strukturního elementu posouván po jednotlivých pixelech vstupního obrázku a poté jsou všechny jeho pixely porovnávány s pixely původního obrázku ležícími pod nimi. Výsledný efekt pak záleží na jednotlivých metodách matematické morfologie [9].

Někdy je možné místo strukturálního elementu použít pojem sousednost. Bod (i, j) je sousedem bodu (k, l), existuje-li takové okolí bodu (i, j), že (k, l) patří do tohoto okolí. Vetšinou se používají tři základní typy okolí a těmi jsou čtyř-okolí (přímé okolí), osmi-okolí (přímé a nepřímé okolí) a nepřímé okolí. Jejich zobrazení je možné vidět v Obr. 5.3 [9].

	X	
X	(i, j)	X
	X	

(a) čtyř-okolí

X	X	X
X	(i, j)	X
X	X	X

(b) osmi-okolí

X		X
	(i, j)	
X		X

(c) nepřímé okolí

Obrázek 5.3: Okolí bodu.

5.2 Zhang-Suen

Tato metoda, někdy označovaná „Fast Parallel Thinning Algorithm“ (rychlý paralelní algoritmus ztenčování), byla navržena roku 1984 a jejími autory jsou T. Y. Zhang a C. Y. Suen [10]. Je to iterativní metoda a každá iterace obsahuje dvě pod-iterace (proto paralelní). Jedna z nich slouží k odstranění jihovýchodních hraničních bodů a severozápadních rohů. Druhá k odstranění severozápadních hraničních bodů a jihovýchodních rohů. Všechny koncové body a původní spojitost jsou ponechány. Každý objekt je ztenčen na kostru s šířkou jednoho pixelu.

Binární obrázek IM je definován maticí, kde každý pixel IM(i,j) má hodnotu 0 nebo 1. Pixely s hodnotou 1 jsou součástí popředí, neboli objektů, které se mají ztenčit. Následuje iterativní procházení matice IM bod po bodu a kontrola jejich nejbližších sousedů. Za sousedy bodu (i, j) jsou považovány body (i – 1, j), (i – 1, j + 1), (i, j + 1), (i + 1, j + 1), (i + 1, j), (i + 1, j – 1), (i, j – 1), (i – 1, j – 1), jak je zobrazeno na Obr. 5.4. Nová hodnota bodu (i, j) se určuje na základě jeho hodnoty a na základě hodnot jeho osmi sousedů [10].

P_9 (i – 1, j – 1)	P_2 (i – 1, j)	P_3 (i – 1, j + 1)
P_8 (i, j – 1)	P_1 (i, j)	P_4 (i , j + 1)
P_7 (i + 1, j – 1)	P_6 (i + 1, j)	P_5 (i + 1, j + 1)

Obrázek 5.4: Označení sousedních bodů. Zdroj [10]

Metoda spočívá v tom, že odstraňuje všechny kontury (obrysy) v obrázku kromě těch, které jsou součástí kostry. Pro zachování spojitosti kostry je každá iterace rozdělána na dvě pod-iterace. V první pod-iteraci je bod kontury P_1 odstraněn v případě, že splňuje následující podmínky:

a) $2 <= B(P_1) <= 6$

b) $A(P_1) = 1$

c) $P_2 * P_4 * P_6 = 0$

d) $P_4 * P_6 * P_8 = 0$

kde $A(P_1)$ je počet přechodů z 0 do 1 mezi okolními body P_1 v matici IM v pořadí P_2, P_3, P_4, ... P_8, P_9, P_2 viz Obr. 5.5. $B(P_1)$ je počet sousedů s hodnotou různou od 0, z toho vyplývá:

$$B(P_1) = P_2 + P_3 + ... + P_8 + P_9 \tag{5.1}$$

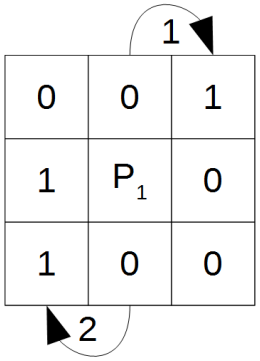
V druhé pod-iteraci jsou pouze pozměněné body c) a d):

c‘) $P_2 * P_4 * P_8 = 0$

d‘) $P_2 * P_6 * P_8 = 0$

a zbytek zůstává stejný. Z podmínek c) a d) z prví pod-iterace je vidět, že první pod-iterace odstraňuje pouze jihovýchodní hraniční body a severozápadní rohové body, které nepatří do kostry. Důkazem je řešení rovnic c) a d), které je: $P_4 = 0$ nebo $P_6 = 0$ nebo ($P_2 = 0$ a $P_8 = 0$). Z toho vyplývá, že P_1 může být východní nebo jižní hraniční bod nebo severozápadní rohový bod. Stejným způsobem se dá odvodit druhá pod-iterace a může být dokázáno, že P_1 je severní nebo západní hraniční bod nebo jihovýchodní rohový bod [10].

Podmínka a) slouží k zachování koncových bodů kostry a podmínka b) zabraňuje smazání bodů ležícími mezi koncovými body kostry. Celý algoritmus končí až v době, kdy není možné odstranit další body [10].



Obrázek 5.5: Ukázka počítání počtu přechodů z 0 do 1. Zdroj [10]

6 Odstranění šumu

Tato kapitola je věnována popisu metod sloužících k odstranění šumu. Tento šum mohl vzniknout při předzpracování nebo při ztenčování. Nejdříve zde však budou popsány jednotlivé typu šumu, které se zde vyskytují.

6.1 Typy šumu

Za šum jsou v tomto případě považovány pixely narušující konzistenci cest nebo pixely, které nejsou pro další rozpoznávání důležité. Některé typy šumu, jedná se především o ty vzniklé po ztenčování, je možné odhalit až po vektorizaci.

6.1.1 Díry v cestách

Tento typ šumu se vyskytuje po předzpracování a je často způsoben nedokonalým rozpoznáním značky přímo na silnici nebo zásahem stavby do cesty, jak je zobrazeno na Obr. 6.1a. Často to také bývají značky jednosměrných cest, jsou to malé šipky uprostřed cesty. Pokud by tento šum nebyl odstraněn, vznikali by pak při ztenčování v přímých ulicích cykly a další nepřirozené větve.

6.1.2 Náhodné pixely

Náhodné pixely vznikají také při předzpracovaní. V mapách se často nacházejí různé barevné přechody a efekty stínů, v nich se mohou vyskytovat pixely se stejnou barvou jakou mají cesty nebo značky. Zpravidla se jedná o malé oblasti s velikostí jednoho až dvou pixelů, to je vyobrazeno na Obr. 6.1b. Při jejich neodstranění by se na těchto místech mohly rozpoznat krátké cesty, které tam ve skutečnosti nejsou.

6.1.3 Značka v cestě

Zde se nejedná tak docela o šum, ale spíše o nedostatek, kterého je možné se vyvarovat. Také se objevuje po předzpracování. Jedná se o značky, které jsou z velké části zasazené do cesty a je u nich na první pohled jasné, že jsou přímo její součástí. Příkladem může být menší text nebo značka ve větší cestě viz Obr. 6.1c. Odstranění tohoto nedostatku výrazně usnadňuje rozpoznávání a snižuje počet případů, kde je nutné pokračování a tvar cesty odhadovat.

6.1.4 Pixely v okolí značek

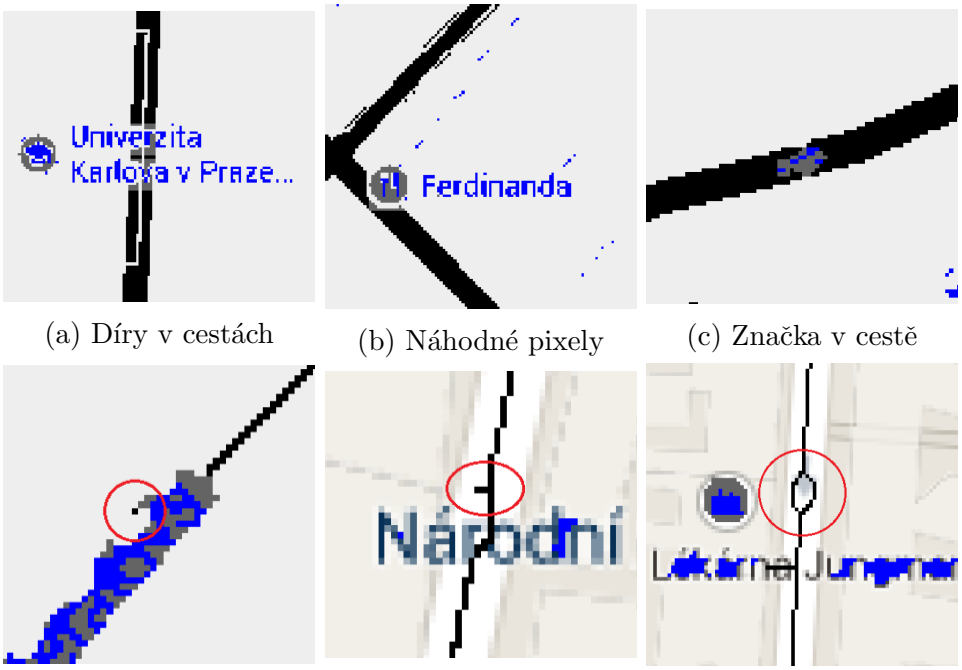
Jedná se o pixely nalepené na značky, které vznikly při ztenčování cest viz Obr. 6.1d. Pokud by tento šum nebyl odstraněn, mohly by při vektorizaci na jejich místech vznikat krátké cesty, které by komplikovaly rozpoznávání spojitosti cest.

6.1.5 Slepé uličky

Toto je jeden ze šumů vzniklých po ztenčování. Jedná se o krátké větve, často kratší než je šířka silnice, vedoucí do míst, kde se nic nenachází, příklad je ukázán v Obr. 6.1e. Vznikají v místech, kde došlo k narušení konzistence cesty. Neodstranění tohoto typu šumu by komplikovalo pokročilejší metody řešící spojitost cest.

6.1.6 Malé cykly

Stejně jako slepé uličky se objevují při ztenčování a vznikají stejným způsobem. Jsou to malé cykly, jejichž průměr často nepřekračuje šířku cesty viz Obr. 6.1f. Jejich přítomnost snižuje kvalitu rozpoznávání.



(a) Díry v cestách (b) Náhodné pixely (c) Značka v cestě

(d) Pixely v okolí značek (e) Slepé uličky (f) Malé cykly

Obrázek 6.1: Jednotlivé typy šumu. Zdroj [2]

6.2 Dilatace

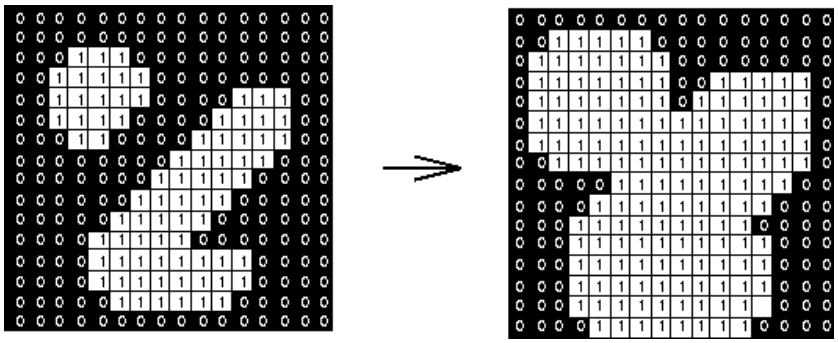
Dilatace je jednou ze dvou základních operací v oblasti matematické morfologie. Typicky je aplikována na binární obrázky (černobíle obrázky), ale existují i verze pracující v šedotónových snímcích. Dilatace slouží k tomu, aby rozšířila hranice oblastí nacházejících se v popředí. Zároveň také slouží k zaplnění malých děr [9].

Matematický popis dilatace pro binární obrázky je následující:

- Předpokládá se, že A je množina bodů v euklidovském prostoru, reprezentujících vstupní obrázek, a B je množina souřadnic strukturního prvku B.
- Nechť B_x značí posunutí B tak, že počáteční hodnota B je v bodě x.
- Potom dilatace A podle B je množina všech bodů x, pro které platí, že průnik množiny A a B_x je neprázdný.

Strukturní elementy mohou mít různé podoby, nejčastěji se však používá element ve tvaru čtverce o rozměrech 3x3 nebo kříž o stejných rozměrech. Příklady těchto strukturních elementů je možné vidět na Obr. 5.2.

Pro vypočtení dilatace binárního obrázku se jednotlivé pixely, označené jako pozadí, překryjí strukturním elementem tak, že jeho počáteční hodnota je v poloze pixelu. Poté, pokud některý z bodů strukturního elementu překrývá pixel popředí, je aktuální pixel také označen jako popředí. Jestliže jsou všechny body strukturního elementu označené jako pozadí, pak je aktuálnímu pixelu ponecháno označení pozadí. Příklad použití je zobrazen na Obr. 6.2, zde byl použit strukturní element ve tvaru čtverce o velikosti 3x3 [9].



Obrázek 6.2: Příklad dilatace. Zdroj [9]

Tato metoda se ve výsledné práci používá pouze jako součást metody uzavírání a proto její uplatnění bude popsáno v kapitole 6.4.

6.3 Eroze

Eroze je jednou ze dvou základních operací v oblasti matematické morfologie. Typicky je aplikována na binární obrázky, ale existují i verze pracující v šedotónových snímcích. Eroze slouží k tomu, aby zúžila hranice oblastí nacházejících se v popředí. Zároveň také slouží k odstranění samo lehlých pixelů [9].

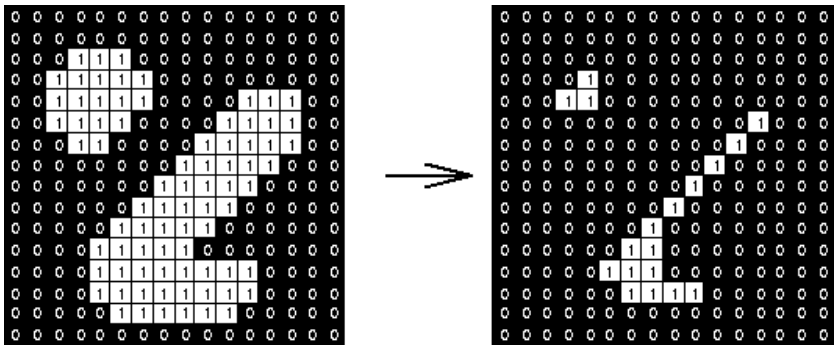
Matematický popis eroze pro binární obrázky je následující:

- Předpokládá se, že A je množina bodů v euklidovském prostoru, reprezentujících vstupní obrázek, a B je množina souřadnic strukturního

prvku B.

- Nechť B_x značí posunutí B tak, že počáteční hodnota B je v bodě x.
- Potom eroze A podle B je množina všech bodů x, pro které platí, že B_x je podmnožinou A.

Strukturní elementy mohou mít stejné tvary jako při dilataci viz Obr. 5.2. Pro výpočet eroze jsou kroky podobné, jako při výpočtu dilatace. Také se jednotlivé pixely, označené jako pozadí, překryjí strukturním elementem tak, že jeho počáteční hodnota je v poloze pixelu. Poté, pokud některý z bodů strukturního elementu překrývá pixel pozadí, je aktuální pixel také označen jako pozadí. Jestliže jsou všechny body strukturního elementu označené jako popředí, pak je aktuálnímu pixelu ponecháno označení popředí. Příklad použití je zobrazen na Obr. 6.3, kde byl také použit strukturní element ve tvaru čtverce o velikosti 3x3 [9].



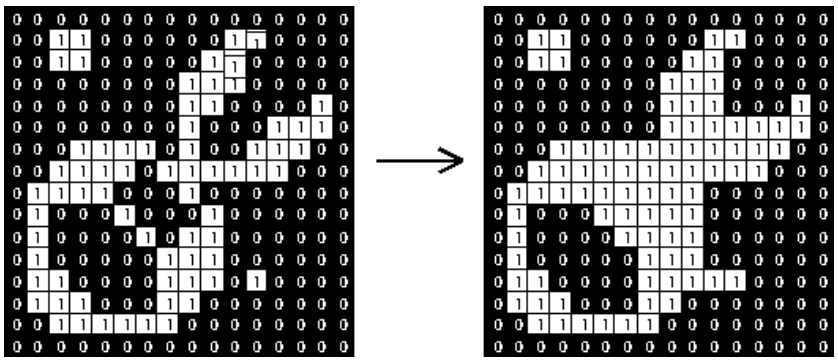
Obrázek 6.3: Příklad eroze. Zdroj [9]

V této práci slouží tato metoda k odstranění šumu „Náhodné pixely“. Efektivně se tak odstraní šum vzniklý po předzpracování. Potenciálně může být tato metoda nebezpečná, protože v některých mapách mají cesty různé šířky. Některé cesty proto mohou mít šířku jeden pixel a aplikováním této metody by mohly úplně zmizet. V Google mapách, pro které je rozpoznávání primárně určené, však tato situace nastat nemůže, protože tam jsou všechny cesty stejně široké a širší než jeden pixel.

6.4 Uzavření

Uzavření patří také mezi důležité operace matematické morfologie, stejně jako jeho opak otevření. Tato operace může být vytvořena s použitím základních operací, kterými jsou dilatace a eroze. Stejně jako tyto operace se běžně aplikuje na binární obrázky, ale existují verze i pro šedotónové obrázky. Uzavírání je v mnoha směrech podobné dilataci, protože zvětšuje některé hranice objektů v popředí a tím uzavírá mezery v nich, ale je mnohem méně destruktivní protože více zachovává původní tvary [9].

Prakticky uzavírání funguje tak, že se na obrázek nejdříve aplikuje operace dilatace a následně se aplikuje operace eroze s využitím jednoho strukturního elementu. Výsledek aplikace uzavření je, při použití strukturního elementu ve tvaru čtverce o velikosti 3x3, vidět v Obr. 6.4.



Obrázek 6.4: Příklad uzavírání. Zdroj [9]

Aplikací této metody na předzpracovaný obraz je možné, zbavit se šumu typu „Díry v cestách“. Tím se docílí efektu, že se některé díry v cestách uzavřou a nebudou tak komplikovat další rozpoznávání. Metoda je upravena tak, aby zacelovala i větší díry, proto na rozdíl od originálního provedení, kde se provede dilatace a pak eroze, v tomto případě se použije dvakrát dilatace a poté dvakrát eroze.

6.5 Slučování oblastí

Slučování oblastí je metodou z oblasti segmentace obrazu, což je skupina metod sloužící pro rozdělení obrázku na oblasti se společnými vlastnostmi.

Tato metoda slouží při rozhodování, zda sloučit dvě zpravidla sousední oblasti [9].

V této práci bude metoda aplikována na obrázky s třemi možnými hodnotami pixelů: 0 pro pozadí, 1 pro cestu a 2 pro značku. Slučování oblastí však bude pouze u oblastí s hodnotami 1 a 2 a pouze za účelem rozhodnutí zda oblast s hodnotami 2 sloučit do oblasti s hodnotou 1 [9].

Prvním krokem metody je nalezení oblastí, a protože se jedná o sloučení oblastí s hodnotami 2 do oblastí s hodnotami 1, stačí nalézt pouze oblasti s hodnotami 2 a poté zkoumat jejich sousední body. Oblast je možné definovat jako shluk bodů, kde každý bod má za souseda alespoň jeden bod z dané oblasti. Proto algoritmus pro nalezení jedné oblasti lze popsat následujícími kroky:

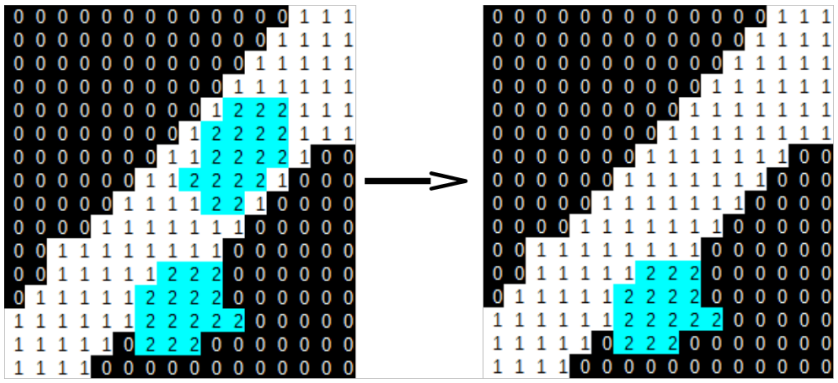
1. nalezení prvního bodu oblasti a jeho přidání,
2. pro všechny nově přidané body najít sousedy,
3. pokud má s soused stejnou hodnotu jako první bod a není již v oblasti, je přidán do oblasti,
4. pokud žádný takový bod není, konec algoritmu,
5. zpět ke kroku 2.

Po nalezení oblasti se přejde k výpočtu. Nejdříve se spočte obvod oblasti, to je počet bodů, které jsou na hranici oblasti, mají tedy souseda, který není součástí oblasti. Takováto hodnota bude označena jako N. Jako další se musí spočíst počet hraničních bodů, které sousedí s oblastí, se kterou má být daná oblast sloučena. V tomto případě se tedy hledá počet bodů oblasti, které mají za sousedy pixely s hodnotou 1, tento počet bude označen W [9].

Potom podmínka určující, zda se oblasti sloučí je:

$$\frac{W}{N} > T \tag{6.1}$$

kde T je minimální poměr, který je nutný ke sloučení oblastí. T může nabývat různých hodnot od 0 do 1. Příklad použití metody slučování oblastí je ukázán na Obr. 6.5, kde hodnota $T = 0{,}5$. Tato metoda se používá pro odstranění šumu typu Značka v cestě.

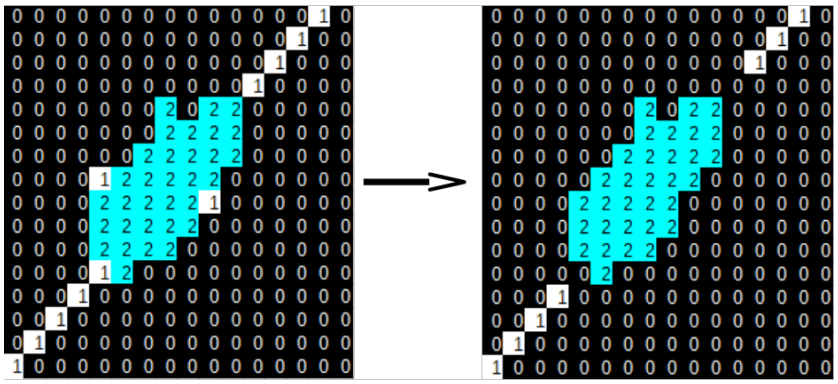


Obrázek 6.5: Příklad slučování oblastí.

6.6 Odstranění zbytků po ztenčování

Tato metoda nepaří do žádné oblasti rozpoznávání obrazu, ale byla vytvořena pouze pro účely této práce. Používá se pro odstranění pixelů zbylých po ztenčování v okolí značek viz Obr. 6.1d. Tradičně by se pro odstranění samostatných pixelů použila metoda eroze, to však v tomto případě není možné, protože tato metoda se aplikuje až po ztenčování, tedy v době kdy mají všechny cesty šířku jednoho pixelu. Eroze by potom odstranila veškeré cesty. Metoda funguje na obrázky se třemi typy pixelů, kde jeden typ je pozadí, další je značka a poslední je cesta. Využívají se zde také strukturní elementy viz Obr. 5.2.

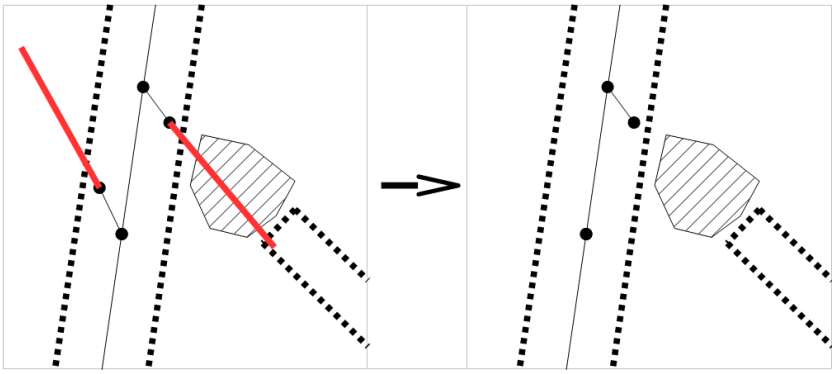
Pro výpočet této metody jsou kroky podobné, jako při výpočtu eroze či dilatace. Jednotlivé pixely, označené jako cesta, se překryjí strukturním elementem tak, že jeho počáteční hodnota je v poloze pixelu. Poté, pokud některý z bodů strukturního elementu překrývá pixel značky, je aktuální pixel označen jako pozadí. Jestliže jsou všechny body strukturního elementu označené jako cesta nebo pozadí, pak je aktuálnímu pixelu ponecháno označení cesta. Příklad použití je zobrazen na Obr. 6.6, kde byl také použit strukturní element ve tvaru čtverce o velikosti 3x3, a kde 0 označuje pozadí, 1 označuje cestu a 2 označuje značku.



Obrázek 6.6: Příklad odstranění zbytku po ztenčování.

6.7 Odstranění slepých uliček

Tato metoda odstranění šumu je velmi přímočará, je možné ji použít až po vektorizaci a byla vytvořena pouze pro účely této práce. Metoda funguje tak, že hledá všechny vektory kratší než je průměrná šířka cest. Po nalezení těchto vektorů kontroluje oblasti do které jednotlivé vektory směřují. Pokud je pokrytí oblastí větší než zadaná hodnota, je tento vektor odstraněn viz Obr. 6.7, kde horní vektor vede do značky a spodní nikam, proto je horní ponechán a spodní smazán. Pokrytí je poměr počtu pixelů oblasti označených jako značka nebo cesta a celkového počtu pixelů oblasti.



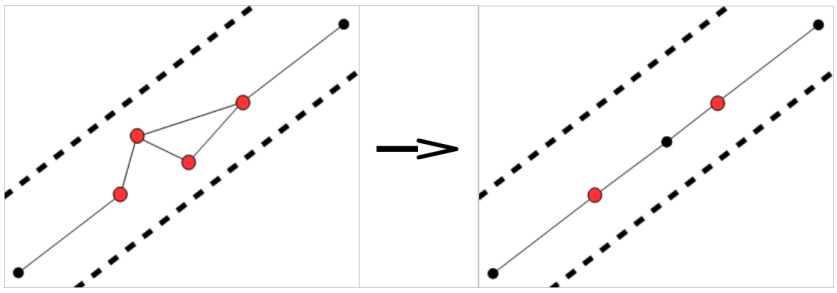
Obrázek 6.7: Příklad odstranění slepých uliček.

6.8 Odstranění cyklů

Odstranění cyklů je velmi komplikovaná metoda, protože obecně hledání cyklů v grafu je NP-úplný problém (ne však detekce cyklů grafu, tam je složitost nižší). Zároveň je také nutné hledat jen dostatečně malé cykly, protože větší cyklus už může být součástí silniční sítě a nemusí se tedy jednat o šum. Tuto metodu pro odstranění šumu je možné použít až po vektorizaci.

Metoda funguje tak, že hledá body, které od sebe leží ve vzdálenosti nižší než je dvakrát průměrná šířka silnice. Po nalezení takové skupiny bodů se zkontroluje, zda mají dostatečný počet bodů. Minimální počet bodů potřebný pro vytvoření cyklu je tři. Na Obr. 6.8 vlevo je vidět nalezení čtyř takovýchto bodů.

Potom následuje určení, zda někde mezi těmito body je cyklus. K tomu se využívá metoda prohledávání grafu do hloubky, kde graf je tvořen pouze tou skupinkou několika bodů. Pokud byl v této skupině nalezen cyklus, tak všechny body které jsou jeho součástí se sloučí do jednoho. Pozice tohoto bodu je průměrem pozic ostatních bodů.



Obrázek 6.8: Příklad odstranění cyklů.

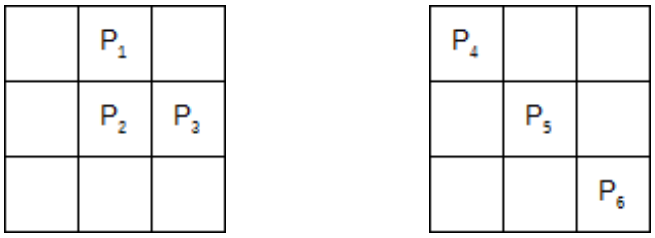
7 Vektorizace

Vektorizace v tomto případě znamená převedení předzpracované bitmapy na seznam bodů a seznam úseček (vektorů, hran), které dohromady tvoří síť. Obsahem této kapitoly bude popsání metod použitých pro získání základní sítě cest z předzpracovaného a ztenčeného obrázku zbaveného nejzávažnějšího šumu.

7.1 Vytvoření základní sítě

V první řadě je nutné nalézt základní síť, nebo-li nalézt pixely, které jsou součástí cesty, a nalézt spojení mezi nimi. Pokud spolu dva body sousedí ve smyslu čtyř-okolí (viz kapitola 5.1.2), je jasné, že mezi nimi existuje spojení, ale v případě, že spolu sousedí nepřímo, není již odpověď jednoznačná.

Na Obr. 7.1 jsou znázorněny dva případy propojování. Je vidět, že body P_1 a P_3 z Obr. 7.1a jsou ve stejném vztahu jako body P_4 a P_5 z Obr. 7.1b. Přesto je u obou případů různé propojení. V prvním případě je vhodnější vytvořit propojení přes bod P_2 a vznikne tak v síti propojení P_1, P_2, P_3. Kdežto v druhém případě je množné body propojit pouze v pořadí P_4, P_5, P_6. Kdyby v prvním případě nedošlo k propojení přes P_2, mohl by tento bod vytvořit novou větev sítě, což je nežádoucí.



(a) (b)

Obrázek 7.1: Příklady propojení

Algoritmus funguje tak, že po nalezení počátečního bodu cesty zkontroluje nejdříve jeho přímé sousedy. Pokud některý z přímých sousedů je také cesta, je přidán do zásobníku pro pozdější vyřízení a strana, na které se nachází je označena jako použitá. Pokud by se například bod P_1 z Obr. 7.1a bral jako

zkoumaný bod, tak bod P_2 by byl přidán do zásobníku a jižní strana by byla označena za použitou.

Poté se u stále stejného bodu zkontrolují nepřímí sousedé. Pokud se však bod nachází na použité straně není bod uvažován. Pokud se na použité straně nenachází a je označen jako cesta, je přidán do zásobníku. Algoritmus se opakuje pro všechny body v zásobníku dokud není prázdný a tím vznikne jedna podsíť. Po nalezení podsítě se hledá další počáteční bod pro který se algoritmus opakuje.

Následuje popis algoritmu procházení pixelů a hledání tak jejich spojitosti. V průběhu tohoto procházení se vždy soused, splňující podmínky pro označení za hotový, přidá jako potomek aktuálního pixelu. To platí i pro sousedy, kteří jsou označení za hotoví. Vznikne tak síť bodů.

Kroky algoritmu:

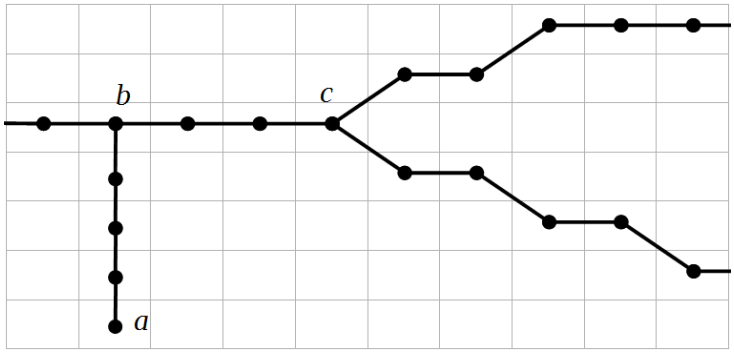
- Definuj pole použitých pixelů.
- Pro všechny pixely v obrázku:
  - Pokračuj, pokud pixel nebyl zpracován a je označen jako cesta.
  - Vytvoř zásobník.
  - Vlož pixel do zásobníku.
  - Prováděj dokud není zásobník prázdný:
    - Vyndej pixel ze zásobníku.
    - Vytvoř pole pro uložení použití strany (strany jsou severní, jižní, západní a východní).
    - Zkontroluj všechny přímé sousedy pixelu:
      - Pokud je soused cesta, přidej ho do zásobníku, stranu, na které je, označ za použitou a označ ho v poli použitých pixelů za hotového.
    - Zkontroluj všechny nepřímé sousedy pixelu:
      - Pokud je soused cesta a strana, na které je, není označena jako použitá, přidej ho do zásobníku a označ za hotového.

7.2 Redukce bodů

Po vytvoření základní sítě je nutné zredukovat počet bodů, protože v tuto chvíli mohou být i úsečky tvořeny několika body, i když jsou na to zapotřebí body pouze dva. Nejprve se síť bodů převede na množinu křivek a poté se zredukuje počet bodů v jednotlivých křivkách pomocí algoritmu pro zjednodušení lomených čas.

Síť je v tuto chvíli tvořena mnoha body, které mezi sebou mají vytvořená spojení viz Obr. 7.2. Body, které mají více než dvě spojení(na obrázku písmeno *b*, *c*), se označují jako křižovatky, body, které mají pouze jedno spojení(na obrázku písmeno *a*), se nazývají slepé uličky a body, které mají přesně dvě spojení(na obrázku všechny ostatní body), jsou středy cest.

Křivku je poté možno definovat, jako posloupnost bodů, kde první a poslední bod jsou křižovatky nebo slepé uličky a zbylé body jsou středy cest. Základní síť cest je poté snadné převést na seznam křivek. Například na Obr. 7.2 bude křivka z bodu *a* do bodu *b* nebo křivka z bodu *b* do bodu *c*.



Obrázek 7.2: Ukázka základní sítě a jejích bodů.

Na jednotlivé křivky je pak možné aplikovat algoritmus pro redukci bodů, takovéto algoritmy se snaží aproximovat původní křivku s minimálním počtem bodů. Přesnost aproximace je často možné nastavovat.

Algoritmů pro redukci bodů křivky, nebo-li pro zjednodušení lomených čas existuje několik a dají se rozdělit do několika skupin. Jedná se o algoritmy nezávislé na tvaru křivky, lokální, rozšířené lokální a globální [11].

Algoritmy nezávislé na tvaru křivky jsou ty nejjednodušší a fungují tak, že vynechají náhodný prvek. Může tak docházet ke ztrátě tvarově významných

vrcholů, což je v tomto případě neakceptovatelné [11].

Lokální algoritmy odstraňují body v souvislosti k svým sousedním bodům. Existuje zde několik možností, jak posuzování provádět, mezi nejčastější patří vzdálenost mezi body nebo úhel mezi body, kde potom platí, když jsou hodnoty menší než zadané kritérium, je bod odstraněn. Příkladem takového algoritmu může být Jenksův algoritmus [11].

Rozšířené lokální algoritmy na rozdíl od lokálních neposuzují bod pouze s jeho sousedy, ale pracují s množinou bodů v jeho okolí. Snaží se tak více zachovávat celkový tvar křivky. Mezi tyto algoritmy patří Reumann-Witkamův algoritmus a Langův algoritmus [11].

Globální algoritmy, jak vyplývá z jejich názvu, zohledňují tvar celé křivky. Při odstraňování bodu ho porovnávají se všemi ostatními body křivky. Z tohoto důvodu byly, pro zjednodušení křivek cest, otestovány právě algoritmy tohoto typu. Byly vyzkoušeny dva nejpoužívanější algoritmy Douglas-Peuckerův algoritmus a Visvalingam-Whyattův algoritmus [11].

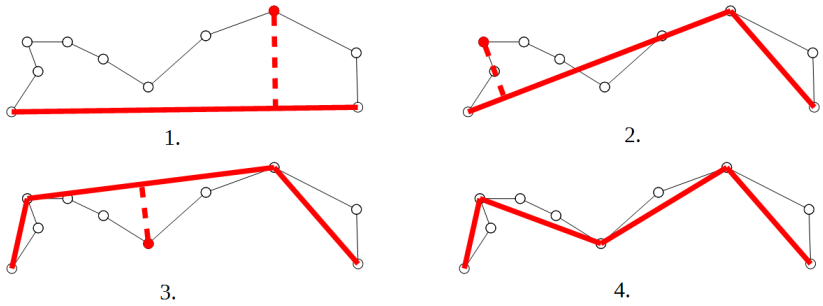
7.2.1 Douglas-Peuckerův algoritmus

Kromě velmi dobrých vizuálních výsledků, je tento algoritmus velmi jednoduchý na naprogramování a je schopný pracovat i ve více dimenzích, jelikož pracuje pouze se vzdáleností bodů a čar. Základním pravidlem aproximace je, že výsledek musí obsahovat podmnožinu bodů původní křivky a všechny ostatní body původní křivky musejí být do určité zadané vzdálenosti od výsledné křivky [12].

Vstupem algoritmu je tedy křivka, nebo-li sekvence bodů, a hodnota ϵ, která označuje maximální povolenou vzdálenost bodů od výsledné křivky. Algoritmus začíná tak, že vytvoří novou křivku mezi prvním a posledním bodem původní křivky, vznikne tak první úsek. Poté pokud existují body, jejichž vzdálenost je od úseku nové křiky větší než ϵ, nalezne se ten nejvzdálenější bod a přidá se do nové křivky. Nová křivka se poté bude skládat ze dvou úseků a na ty se pak aplikuje stejný postup. Tímto způsobem se rekurzivně pokračuje dokud neexistuje bod, který by byl k nové křivce vzdálený více než ϵ [12]. Náznak algoritmu je zobrazen na Obr. 7.3.

Při zvolení většího ϵ může docházet k situacím, že zjednodušená křivka protíná sama sebe. Existuje několik způsobů, jak toto řešit. V této práci to

to řešeno není, protože je při zjednodušování vyžadována velká přesnost a zvolené ϵ je velmi malé [12].

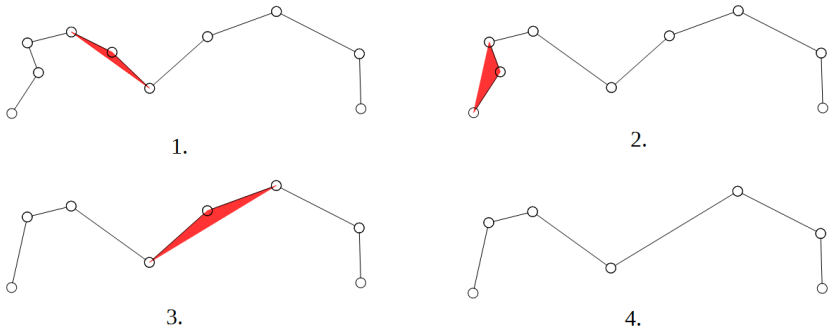


Obrázek 7.3: Douglas-Peuckerův algoritmus.

7.2.2 Visvalingam-Whyattův algoritmus

Algoritmus pracuje opačně než Douglas-Peuckerův algoritmus, který body vybíral a přidával, nyní se body naopak odebírají. Vstupem algoritmu je také sekvence bodů a hodnota ϵ, která značí mezní hodnotu. Algoritmus pracuje s velikostí trojúhelníku bodu P_N. Tato velikost je vlastně obsah trojúhelníku tvořeného bodem P_N a jeho dvěma sousedy P_{N-1} a P_{N+1} [13].

Při výpočtu se pak spočítají velikosti trojúhelníků pro všechny body a vybere se ten s nejmenší hodnotou. Pokud je jeho hodnota vetší než ϵ, algoritmus může skončit. V opačné případě je bod odebrán. Následně se přepočítají velikosti trojúhelníků bodů, které sousedili s odstraněným bodem, opět se provede výběr prvku s nejmenší velikostí trojúhelníku a algoritmus se opakuje. Ukázka algoritmu je vidět na Obr. 7.4 [13].



Obrázek 7.4: Visvalingam-Whyattův algoritmus.

8 Spojování cest

Protože mapy jsou plné značek, dochází k rozpojení cest. Tématem této kapitoly bude popsání způsobů použitých pro nalezení míst, kde k takovým rozpojením došlo, aby mohlo být spojení znovu utvořeno. Jde tedy o to, najít dva a více bodů, které mají být potenciálně propojeny. Způsoby rozpojení je možné rozdělit do několika kategorií a těmi jsou rozpojení přímých cest, kolmých cest a složitějších cest. Všechny popsané postupy byly vytvořeny v rámci této práce.

8.1 Přímé cesty

Rozpojení v přímé cestě je nejčastějším způsobem rozpojení a jeho rozpoznání je klíčové, protože často ovlivňuje i detekci rozpojení v kolmých cestách. Přímou cestou je myšlena téměř rovná cesta bez prudších zatáček. Pro rozpoznání rozpojení v přímých cestách je možné zavést pravidla, podle kterých je možné určit, zda v daných místech skutečně došlo k rozpojení. Tato pravidla se pak aplikují na dvojice bodů, respektive na dvojici vektorů, a ty se buď spojí nebo jsou ponechány v původním stavu.

Pravidla jsou následující:

1. rozdíl úhlů cest,
2. rozdíl posunutí cest,
3. pokrytí mezi cestami,
4. překřížení s jinou cestou.

Pravidlo rozdílu úhlů cest je základním pravidlem, rozhoduje totiž o tom zda mají cesty přibližně stejný sklon a jsou tedy do určité míry přímé. Je možné ho popsat následujícím vzorcem:

$$\Theta < 30°, \tag{8.1}$$

kde Θ značí rozdíl úhlů mezi dvěma vektory viz Obr. 8.1a. Hodnota 30° byla vybrána po sadě pokusů jako hodnota s nejlepšími výsledky. Nižší hodnoty

nepropojovali některé cesty a vyšší hodnoty naopak propojovaly cesty, u kterých se to již nehodilo. Druhým pravidlem je posunutí cest, to je nejlépe vidět na Obr. 8.1b. Vzorec pro určení, zda je pravidlo splněno, je následující:

$$d < max(w_1, w_2), \tag{8.2}$$

kde d je velikost posunutí a w_1, w_2 jsou šířky jednotlivých cest. Toto pravidlo je nutné prozkoumat v obou směrech, od jedné cesty k druhé a naopak, protože první cesta nemusí směřovat na druhou, ale druhá cesta může směřovat na první.

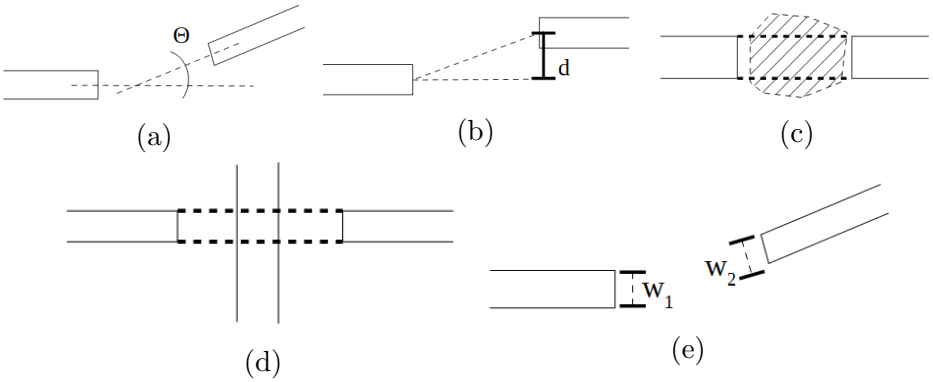
Dalším pravidlem je pokrytí mezi cestami. To zajišťuje, že se mezi cestami bude nacházet významná značka, která cesty rozpojila viz Obr. 8.1c, kde je šrafovaně vyznačena značka. Pokud by tam totiž značka nebyla, znamená to, že neexistuje nic, co by způsobilo rozpojení, a proto tam také cesta být nemohla, i když byly splněné ostatní podmínky.

Pravidlo funguje tak, že se mezi cestami vytvoří imaginární cesta. U této imaginární cesty se spočte její obsah (počet pixelů, kterými je tvořena) a poté se spočte obsah všech překrytých oblastí (počet pixelů značek a cest, ve vytvořené oblasti). Podmínka pro splnění pravidla je následující:

$$\frac{pocet(prekryti)}{pocet(celkem)} < 0.7, \tag{8.3}$$

kde hodnota 0.7 (nebo-li 70%) byla vybrána po sadě pokusů jako hodnota s nejlepšími výsledky.

Pravidlo překřížení cest slouží k tomu, aby nebyla vytvořena cesta, přes již existující cestu. Znamená to totiž, že se nejedná o přímé cesty nýbrž o křižovatky viz Obr. 8.1d, které jsou řešeny až v následujících kapitolách.



Obrázek 8.1: Ilustrace pravidel.

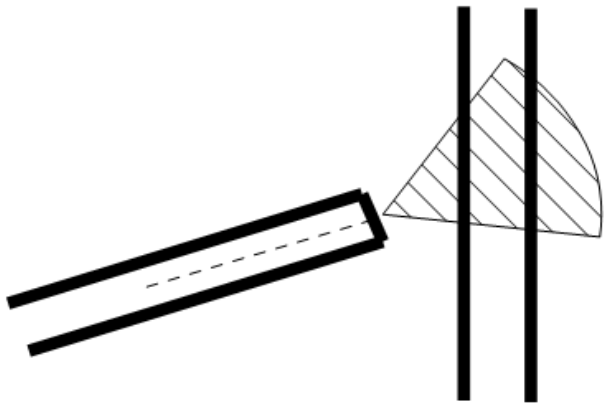
Na Obr. 8.1e je vyobrazeno pravidlo, které bylo také testováno. Jedná se o rozdíl šířek cest. Pravidlo je možné popsat následujícím vzorcem:

$$\frac{|w_1 - w_2|}{max(w_1, w_2)} < s, \tag{8.4}$$

kde w_1 a w_2 jsou šířky cest a s je desetinné číslo od 0 do 1 (standardně se používalo $s = 0.5$ jako hodnota, která měla při pokusech nejlepší výsledky). Protože často docházelo k rozbití cest a nemožnosti správně určit jejich šířku, nebylo pravidlo nakonec použito. Šířky se sice používají i v druhém pravidle, tam ale nejsou tak zásadní a je možné je nahradit jinou hodnotou.

8.2 Kolmé cesty

Jedná se o cesty, které se napojují na přímé cesty, většinou se to týká křižovatek ve tvaru písmena T. Jde o to rozhodnout, zda se má doposud slepá ulička připojit na přímou cestu, která se nachází v její blízkosti. Pro rozhodnutí, zda má dojít ke spojení se kontroluje oblast, ve tvaru kruhové výseče viz Obr. 8.2, na konci slepé cesty. U oblasti se hledá cesta, která ji protíná. Je-li takových cest více, použije se ta nejblíže koncovému bodu slepé ulice. Pokud není nalezena žádná cesta, k propojení nedojde.



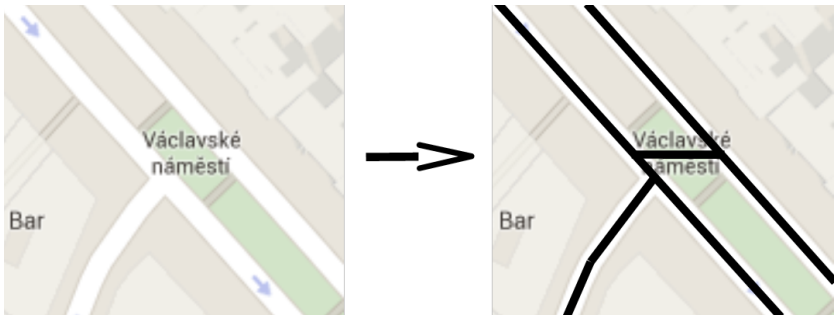
Obrázek 8.2: Spojitost kolmých cest.

Při použití této metody je možné nalézat i rozpojení v některých komplikovaných cestách, jedná se o cesty, které jsou přerušeny malou značkou v zatáčce.

8.3 Komplikované cesty

Mezi komplikované rozpojení cest patří přerušení vzniklé v zatáčkách cest nebo ve složitějších křižovatkách, na které nestačila aplikace předchozích dvou metod. Tuto metodu je možné aplikovat i na přímé a kolmé rozpojení cest, ale jejich následné propojení není tak kvalitní. Problém by třeba nastal při aplikaci na rovnoběžné cesty, přes které je vodorovný nápis (viz Obr. 8.3). Došlo by tak k propojení mezi rovnoběžnými cestami, které tam ve skutečnosti být nemá.

Algoritmus využívá takzvaných fiktivních cest. Ty vznikly při ztenčení a vektorizaci značek v obrázku. Tyto fiktivní cesty se použijí jako řešení, při hledání komplikované spojitosti.



Obrázek 8.3: Ukázka špatného spojení při komplikovaném propojování. Zdroj podkladu [10]

9 Analýza aplikace

V této kapitole budou nejprve specifikovány požadavky a popsány případy užití v aplikaci. Poté bude navržen postup zpracování celku a jednotlivých částí programu. Také zde budou navrženy struktury vstupních a výstupních souborů a nakonec bude vytvořen návrh uživatelského rozhraní.

9.1 Specifikace požadavků

Požadavkem na aplikaci je, aby dokázala z bitmapových obrázků vytvořit silniční síť. Aplikace tedy musí umět v prvé řadě načíst vstupní obrázek s mapou z disku. Musí být schopná načítat základní obrázkové typy JPEG, PNG, GIF, BMP a popřípadě další. Obrázky budou obsahovat obecné mapy, které se skládají ze zjednodušeného schématického zobrazení silniční sítě, základního rozlišení zástavby a porostu a mohou obsahovat různé body zájmu.

Zároveň bude aplikace také schopná načítat konfigurační soubor ve formátu XML. V tomto konfiguračním souboru budou specifikovány barvy cest a barvy značek. Konfigurační soubor bude možné v programu i vytvořit.

Po načtení obrázku a načtení nebo vytvoření konfiguračního souboru bude možné spustit rozpoznávání, které z načteného obrázku rozpozná silniční síť a převede ji do podoby neorientovaného grafu. Tento graf bude poté možné uložit ve formátu XML s nějakou přehlednou strukturou zpět na disk.

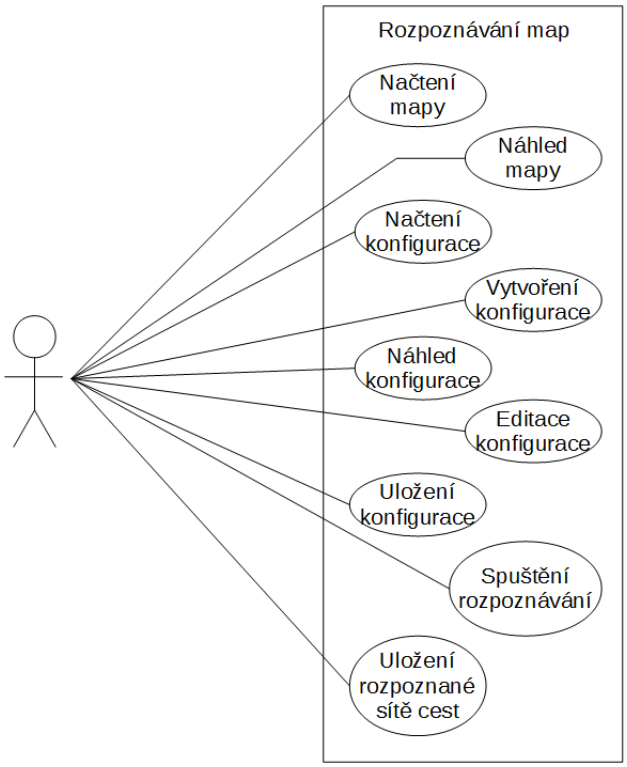
Aby mohl uživatel aplikaci snadno ovládat, bude k ní vytvořeno i uživatelské rozhraní. Pomocí tohoto uživatelského rozhraní bude možné provádět výše popsané akce.

9.2 Případy užití

V aplikaci bude několik případů užití viz Obr. 9.1 a těmi jsou:

- Načtení mapy - bude uživateli umožňovat vybrat obrázek z disku a načíst ho do aplikace.

- Náhled mapy - uživateli umožní prohlídnou načtený obrázek. Zároveň také bude možné s obrázkem pohybovat, přibližovat a oddalovat ho.
- Načtení konfigurace - bude uživateli umožňovat vybrat soubor s konfigurací ve formátu XML z disku a načíst ho do aplikace.
- Vytvoření konfigurace - dovolí uživateli vytvořit novou konfiguraci s pomocí načteného obrázku. Bude moci přidávat jednotlivé barvy z obrázku do seznamu barev cest nebo značek.
- Náhled konfigurace - umožní uživateli spustit předzpracování obrázku, aby viděl jak daná konfigurace s obrázkem pracuje.
- Editace konfigurace - umožní uživateli měnit načtenou nebo vytvořenou konfiguraci s pomocí načteného obrázku.
- Uložení konfigurace - dá uživateli možnost uložit aktuální konfiguraci v aplikaci uložit ve formátu XML na disk.
- Spuštění rozpoznávání - umožní uživateli spustit celý proces rozpoznávání, jehož výsledkem bude síť cest.
- Uložení rozpoznané sítě - umožní uživateli uložit soubor ve formátu XML, který bude obsahovat rozpoznanou síť silnic, na disk.

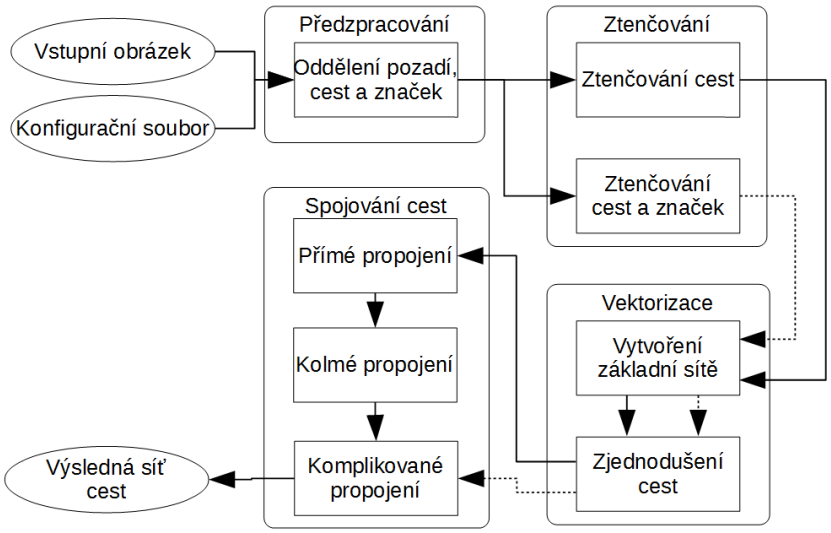


Obrázek 9.1: Diagram přpadů užití.

9.3 Návrh postupu zpracování

Z důvodu, že jsou si všechny mapy graficky velmi podobné bude snaha o to, vytvořit univerzální nástroj na rozpoznávání map z různých zdrojů. Přesto primárním zdrojem dat, pro který bude požadována nejvyšší přesnost, byly vybrány Google mapy pro jejich velkou přesnost a velmi dobrou strojovou čitelnost.

Tématem této kapitoly bude popsání možností využití algoritmů popsaných v kapitolách 4, 5, 6, 7, 8 a dalších algoritmů, které byly zvažovány a testovány. Zároveň zde budou popsány způsoby respektive struktury vstupních a výstupních souborů. Na Obr. 9.2 je potom vidět výsledný zvolený průběh rozpoznávání. Rozpoznání silniční sítě je možné dekomponovat na několik součástí. Tyto součásti jsou předzpracovaní, ztenčování, vektorizace a propojování cest.



Obrázek 9.2: Průběh rozpoznávání.

Prvním krokem algoritmu musí být načtení obrázku z disku, to je triviální záležitost, protože většina dnešních vyšších programovacích jazyků má na to své vlastní knihovny, proto není nutné toto dále rozvádět. Po načtení obrázku je nutné načíst nebo vytvořit konfiguraci (viz dále). Pak již následuje první práce s obrázkem a tou je předzpracování.

9.3.1 Předzpracování

Předzpracování je první z algoritmů, který bude aplikovaný na načtený obrázek. V kapitole 4 jsou popsány tři způsoby, které jsou zvažovány pro oddělení popředí a pozadí. Přestože metoda K-means se zdá být dobrým kandidátem tak, z důvodu nutnosti použití velké hodnoty K, je příliš časově náročná a její výsledky této časové náročnosti neodpovídají. Nicméně některé její části jsou použity v dalších dvou metodách.

Jedním z možných řešeních jsou i jiné algoritmy strojového učení, které byly otestovány za použití různých knihoven, ale žádný nevyhovoval. Buď byly jejich výsledky nedostačující nebo byly příliš časově náročné.

Bude tedy zapotřebí vyvinout vlastní metody pro oddělení pozadí a popředí (a značek). Jedná se tedy o metody jednoduchého výběru barev a složitého výběru barev, jejichž popis je v kapitolách 4.2 a 4.3. Použita nakonec

bude metoda složitého výběru barev, protože dokázala odlišit i značky.

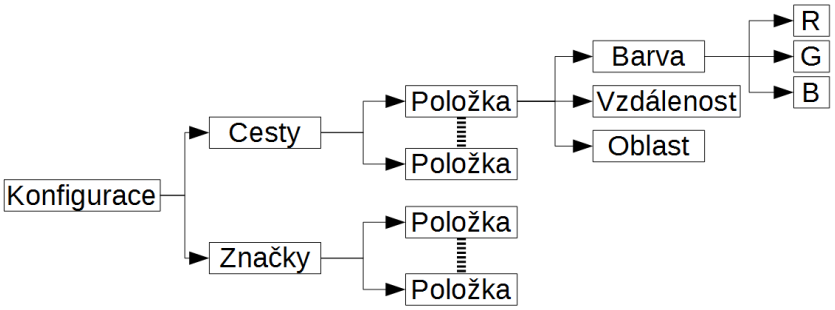
Protože tato metoda vyžaduje jako vstup seznam barev použitých na cesty a značky, je nutné ho někde získat. Jednou možností bude vytvoření těchto seznamů před spuštěním rozpoznávání viz kapitola 9.4 nebo načtením ze souboru. Tento soubor je dále označován jako konfigurační soubor.

Před začátkem ztenčování bude ještě nutné odstranit co možná nejvíce šumu, který v průběhu předzpracování může vznikat. Proto se použijí metody uzavření z kapitoly 6.4, eroze z kapitoly 6.3 a slučování oblastí z kapitoly 6.5. Metoda uzavření bude sloužit k odstranění děr vzniklých v cestách, metoda eroze k odstranění samostatných pixelů a metoda slučování oblastí ke sloučení některých značek do cest.

9.3.2 Konfigurační soubor

Aby mohlo být vytvořené jednotné ukládání a načítání konfiguračních souborů, je nutné nejdříve navrhnou jejich strukturu. Základním kamenem této struktury je položka (Item). Položka se skládá z barvy, vzdálenosti a okolí. Barva položky se skládá z červené, zelené a modré. Vzdálenost položky určuje rozsah barvy a oblast určuje působnost barvy viz kapitola 4.3.

Jednotlivé položky jsou umístěné v jedné ze dvou oblastí. První oblastí jsou cesty, které obsahují všechny položky s barvami cest. Druhou oblastí jsou značky, které obsahují všechny položky s barvami značek. Struktura konfiguračního souboru je naznačená v Obr. 9.3.



Obrázek 9.3: Struktura konfiguračního souboru.

9.3.3 Ztenčování

Ztenčování bude další nezbytnou fází, která slouží k usnadnění vektorizace. Popis ztenčování je uveden v kapitole 5. Metod pro ztenčování existuje nepřeberné množství, ze kterého je vybrán algoritmus Zhang-Suen. Ostatní algoritmy mají velmi podobné výsledky, byl ale zvolen tento, protože jeho implementace je velice jednoduchá a snadno pochopitelná (alespoň pro autora této práce). To je docela zásadní vlastnost, protože snadno pochopitelné algoritmy se dají snáze upravovat. Je tak možné ho přizpůsobit celkovému fungování programu.

Při ztenčování se budou vytvářet dvě bitmapy, v první z nich se ztenčí pouze oblasti označené jako cesta a v druhé se ztenčí vše (cesty i značky). První bitmapa bude sloužit k dalšímu rozpoznávání a její výhoda je, že oblasti značek zůstanou zachovány a je tak možné snáze určovat spojitost přímých a kolmých cest. Druhá bitmapa se bude později využívat při řešení spojitosti komplikovaných cest.

Po ztenčování bude ještě na první bitmapu aplikována metoda z kapitoly 6.6 na odstranění šumu způsobeného ztenčováním v blízkosti oblastí značek. U druhé bitmapy se toto odstranění používat nebude, protože tam bude docházet i ke ztenčení značek.

9.3.4 Vektorizace

Po dokončení ztenčování bude možné bitmapové obrázky převést na sadu bodů a vektorů. Vektorizaci je možné rozdělit na dvě fáze. První fází je vytvoření základní sítě, kdy bude nutné převést pixely do grafu, při zachování jejich sousednosti viz kapitola 7. Pro tento účel byl vytvořen vlastní algoritmus.

Druhou fází po vytvoření základní sítě je redukce bodů. Redukce bodů sníží celkový počet bodů sítě, respektive jejích jednotlivých křivek, za cenu snížené přesnosti. Zde se nabízí několik možných řešení, protože algoritmů pro redukci bodů je velké množství viz kapitola 7.2. Tyto algoritmy se dělí do několika skupin, jsou jimi nezávislé na tvaru křivky, lokální, rozšířené lokální a globální. Jelikož je nezbytné zachovat celkový tvar křivek, byly testovány pouze algoritmy globální.

Zde se jako řešení nabízí algoritmy Douglas-Peuckerův a Visvalingam-Whyattův. Oba generují velmi podobné výsledky a jejich výsledky jsou navíc dostatečně kvalitní pro další práci. Nicméně z důvodu nižší implementační náročnosti bude použit Douglas-Peuckerův algoritmus.

Po dokončení vektorizace bude možné odstranit některé šumy vzniklé při ztenčování. Použije se tedy metoda z kapitoly 6.7 pro odstranění slepých uliček, aby následně při kontrole spojitosti nedocházelo k dalším chybám.

9.3.5 Spojování cest

Hledání spojitosti cest je velmi specifický problém, proto neexistují žádné konkrétní algoritmy, jak tento problém řešit, je tedy nutné vyvinout vlastní. Zpočátku byla snaha vytvořit nějaký obecný seznam pravidel, podle kterého by se propojování provádělo. Bohužel po několika pokusech se došlo k závěru, že základním pravidlem silniční sítě je, že nemá pravidla.

Je tedy nutné problém dekomponovat na menší části, tyto části jsou přímé cesty, kolmé cesty a komplikované cesty viz kapitola 8. K takovéto dekompozici došlo, protože přímé a kolmé rozpojení cest jsou nejčastější typy rozpojení a je možné u nich zavést nějaká pravidla. Komplikované rozpojení jsou všechny ostatní rozpojení u kterých není možné žádná pravidla určit.

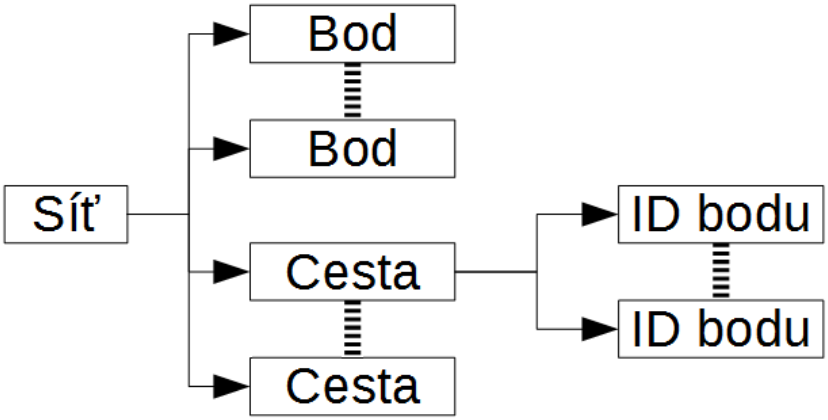
Po dokončení spojování bude již rozpoznání silniční sítě kompletní, může se zde však vyskytovat ještě jeden typ šumu a tím jsou cykly v cestách. Tento problém bude odstraněn aplikováním metody pro odstranění cyklů z kapitoly 6.8.

9.3.6 Uložení výsledků

Po dokončení rozpoznávání je nutné výsledky nějak uložit. Je tedy nutné navrhnout strukturu dat do které se budou výsledky ukládat. Struktura ukládaných dat byla zvolena tak, aby se co možná nejvíce podobala struktuře dat generované mapovými podklady OpenStreetMap viz Obr. 9.4, protože jsou obecně známé.

Struktura se tedy skládá z bodů (node) a cest(way). Každý bod má své identifikační číslo a své souřadnice. Cesta má také své identifikační číslo a

obsahuje v sobě reference na body, které obsahuje.



Obrázek 9.4: Struktura výstupního souboru.

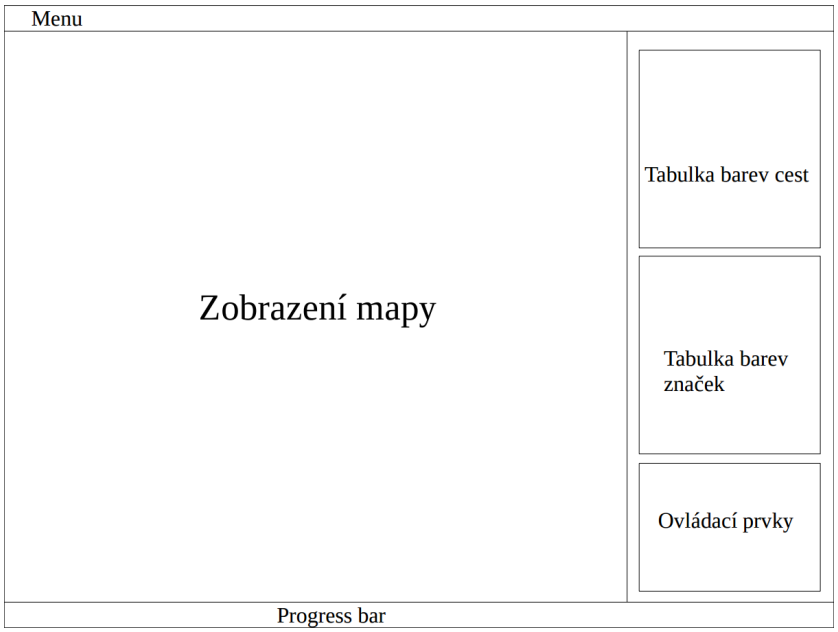
9.4 Návrh uživatelského rozhraní

Grafické uživatelské rozhraní(GUI) musí sloužit k vizualizaci výsledků a ovládání programu. Okno bude rozdělené do dvou hlavních částí. Jednou částí je samotné zobrazení načtené mapy a druhá část bude obsahovat ovládací prvky viz Obr. 9.5.

Zobrazení mapy v první části bude možné transformovat. Bude ji možné posouvat nebo ji přibližovat a oddalovat. Část s ovládacími prvky bude ještě dále rozdělena na část s tabulkou barev cest, část s tabulkou barev značek a na část pracovní, která bude obsahovat například tlačítka pro spuštění rozpoznávání.

Seznamy barev odpovídají konfiguraci rozpoznávání a bude možné naplnit buď načtením souboru nebo vytvořením přímo GUI. Bude zde tedy možnost přidávat barvy přímo z mapy. Při kliknutí pravým tlačítkem na mapu se objeví kontextové menu, ve kterém bude na výběr přidat barvu do seznamu barev cest nebo značek.

V horní části okna potom bude menu s dalšími ovládacími prvky sloužícími například k načítání souborů nebo ukládání silniční sítě. V dolní části okna bude progress bar, který bude zobrazovat průběh rozpoznávání.



Obrázek 9.5: Rozvrh GUI.

10 Programové řešení

Tato kapitola se bude věnovat způsobu implementace aplikace. Jsou zde popsány jednotlivé části programu a jejich fungování. Zároveň jsou zde zmíněny technologie použité při vývoji.

Aplikace byla navržena tak, aby bylo možné jednotlivé části programu snadno vyměnit, například zaměnit stávající způsob ztenčování za jiný. Proto většina balíků obsahuje rozhraní, ve kterém jsou deklarovány potřebné metody. Tato rozhraní zde popisována nebudou. Zároveň zde také nebudou popsány třídy, které nakonec ve finální aplikaci použity nebyly.

10.1 Použité technologie

Aplikace byla vyvíjena pod operačním systémem Windows 7 a jako vývojové prostředí použit program Eclipse Luna. Programovacím jazykem byla zvolena Java spravovaná společností Oracle a to především kvůli její přenositelnosti. Pro grafické uživatelské rozhraní byla zvolena knihovna Swing a pro ukládání struktury silniční sítě byla zvolena technologie XML a konkrétně parser DOM, protože načítané a ukládané soubory nejsou příliš objemné a DOM je snazší na implementaci. Oba tyto prostředky jsou obsažené v balících v Java Core API.

10.2 Struktura aplikace

Aplikace je rozdělena do několika balíků. Každý balík v sobě má třídy se stejným nebo podobným zaměřením viz příloha B. Balíky reprezentují jednotlivé obecné činnosti, které aplikace provádí pro úspěšné rozpoznávání struktury silniční sítě. Balíky aplikace jsou:

- balík s pomocnými třídami,
- balík předzpracování,
- balík ztenčování,

- balík vektorizace,
- balík spojování,
- balík operací pro odstranění šumu,
- balík vizualizace.

10.3 Balík s pomocnými třídami

Tento balík obsahuje třídy, které nebylo vhodné zařadit do jiných balíků pro jejich univerzálnost. Jedná se o třídy načítající a vytvářející vstupní a výstupní XML soubory a o obecně pomocnou třídu. Výčet jednotlivých tříd a jejich popis je následující:

`XMLWorker` – jedná se o třídu schopnou pracovat s XML soubory. Je to třída, která se používá k načítání a ukládání konfiguračních souborů a k ukládání souboru s rozpoznanou silniční sítí.

`Pixel` – třída reprezentující jeden pixel v obrázku. Udržuje v sobě pozici pixelu a jeho barvu. Je používán napříč celou aplikací k různým účelům.

`Help` – pomocná třída. Jedná se o skupinu statických metod používaných v různých částech programu. Jedná se například o metody vytvářející osmiokolí z kapitoly 5.1.2 nebo o metody kontrolující, zda jsou zadané souřadnice stále ještě v obrázku.

10.4 Balík předzpracování

Balík pro předzpracování obsahuje tříd použitých při předzpracování obrazu viz kapitola 4. Třídy jsou následující:

`HardClear` – provádí složitý výběr barev z kapitoly 4.3. Protože jednotlivé pixely jsou na sobě nezávislé, provádí se výpočet paralelně, aby došlo k urychlení na více-jádrových procesorech.

`ImagePreprocessing` – třída provádějící předzpracování. S pomocí třídy `HardClear` vyčistí obrázek od pozadí a zvýrazní cesty nebo značky a jejich

okolí. Poté provede operaci pro uzavření některých děr s třídou `Closing`. Následně provede třídou `Erode` erozi, která odstraní zbylý šum.

10.4.1 Metoda `clearMap()`

Tato metoda je součástí třídy `HardClear` a provádí oddělení pozadí, cest a značek. V programu je zpracována tak, aby pracovala paralelně. Zde je ukázka pouze její neparalelní verze a podobnost barev je dána jejich euklidovskou vzdáleností a zadaným rozsahem barev.

```
1: function clearMap(obrazek)
2:     for všechny pixely v obrázku do
3:         nastav pixel na pozadi
4:         nalezeno ← false
5:         for všechny barvy cest do
6:             if barva pixelu podobná aktuální barvě then
7:                 nastav pixel na cesta
8:                 nalezeno ← true
9:             end if
10:        end for
11:        if !nalezeno then
12:            for všechny barvy značek do
13:                if barva pixelu podobná aktuální barvě then
14:                    nastav pixel na znacka
15:                end if
16:            end for
17:        else
18:            okoli ← najdiOkoliPixelu()
19:            for všechny barvy značek do
20:                for všechny pixely okoli do
21:                    if barva pixelu okolí podobná aktuální barvě then
22:                        nastav pixel na mozna_cesta
23:                    end if
24:                end for
25:            end for
26:        end if
27:    end for
28: end function
```

10.5 Balík ztenčování

Tento balík slučuje třídy, ve kterých se provádí ztenčování viz kapitola 5.

`ZhangSuenThinning` – třída provádí ztenčování v obrázku. Pokud byl v metodě `thinning()` nastaven parametr na kompletní ztenčení, provede se ztenčení všeho co nebylo rozpoznáno jako pozadí. Pokud tak parametr nastaven nebyl, provede se pouze ztenčení oblastí označených jako cesty.

10.6 Balík vektorizace

V tomto balíku se nacházejí třídy potřebné k převedení ztenčeného obrázku na seznam bodů a vektorů viz kapitola 7. Jeho třídy jsou následující:

`PixelNetworkNode` – reprezentuje jeden bod základní sítě (ta se může skládat z více podsítí). Obsahuje v sobě seznam všech svých dalších sousedů, proto stačí reference pouze na jeden bod k získání přístupu do celé podsítě. Zároveň v sobě obsahuje metodu na převedení základní sítě na seznam křivek, který je důležitý v následujících krocích.

`PixelConnectivity` – třída vytváří základní síť ze vstupního obrázku viz kapitola 7.1 a používá k tomu třídu `PixelNetworkNode`. Výsledkem je poté seznam počátečních bodů jednotlivých podsítí.

`PixelLine` – reprezentuje jednu křivku. Jedná se o posloupnost sousedních bodů. Zároveň také obsahuje metodu na převedení křivky na vektory (třídy typu `RoadVector` a `RoadNode`)

`RoadNode` – reprezentuje jeden bod zpracované sítě. Obsahuje v sobě seznam vektorů, kterých je součástí.

`RoadVector` – reprezentuje jednu spojnici (vektor) mezi dvěma body. Obsahuje velké množství metod, které se dále využívají při hledání spojitosti cest.

`Vectorization` – hlavní metoda tohoto balíku. Za použití všech ostatním tříd v balíku převádí zadaný ztenčený obrázek na sadu bodů a vektorů, které dohromady tvoří síť. Nejprve pomocí třídy `PixelConnectivity` vytvoří základní sít, kterou následně, pomocí stejné třídy, převede seznam křivek. Na

jednotlivé křivky se poté aplikuje metoda pro zjednodušení lomené čáry. Tím vznikne seznam bodů a vektorů.

10.6.1 Metoda `cutNetwork()`

Tato metoda slouží k převedení základní sítě na seznam křivek, které mohou být následně zjednodušeny. Metoda je součástí třídy `PixelNetworkNode`. Protože základní síť se může skládat z několik podsítí, je nutné tuto metodu spustit pro každý počáteční bod každé podsítě. Tento počáteční bod podsítě musí být typu křižovatka nebo slepá ulička.

```
1: function cutNetwork(rodic, bod, krivka)
2:     if krivka != null then
3:         krivka přidej bod
4:     end if
5:     if bod je slepá ulička || bod je křižovatka then
6:         for všechny sousedy bod do
7:             if soused nebyl zpracováván then
8:                 vytvoř novou křivku novaKrivka
9:                 novaKrivka přidej bod
10:                cutNetwork(bod, soused, novaKrivka)
11:            end if
12:        end for
13:    else
14:        for všechny sousedy bod do
15:            if soused != rodic then
16:                cutNetwork(bod, soused, krivka)
17:            end if
18:        end for
19:    end if
20: end function
```

10.7 Balík spojování

Balík spojování obsahuje třídy sloužící k nalezení cest, kde došlo k rozpojení, a jejich následné spojení viz kapitola 8.

`Chain` – jedná se o třídu, kterou je možné vytvořit stromovou strukturu, kdy každý potomek zná svého rodiče, ale rodič nezná své potomky. Slouží při prohledávání sítě k rekonstrukci procházené cesty.

`Cone` – reprezentuje kruhovou vyseč, která se používá při hledání kolmé spojitosti. Využívá se pro skenování oblasti, do které směřuje nějaký vektor, za účelem zjištění, zda se tam nachází jiná cesta.

`DirectConnection` – třída používaná pro detekci a propojování přímých cest. Jsou v ní implementována pravidla popsána v kapitola 8.1. Přímé cesty se propojí tak, že se mezi dvěma body, pro která byla splněna pravidla, vytvoří nový vektor.

`OrthogonalConnection` – používá se pro detekci a propojení kolmých cest. Pro kontrolu zda slepá ulička někam pokračuje používá třídu `Cone`. Při propojení kolmých cest je nejdříve nutné nalézt nejvhodnější místo pro křižovatku. Po nalezení tohoto místa se původní vektor, do kterého vede kolmá cesta, smaže a nahradí se dvěma dvěma vektory, vedoucími z původních bodů vektoru do nového bodu křižovatky. Nakonec je do bodu křižovatky připojena i kolmá cesta.

`AdvancedConnection` – třída, která řeší komplikované propojení cest. Algoritmus této metody je popsán v kapitole 8.3.

`Connection` – třída, která za použití tříd `DirectConnection`, `OrthogonalConnection` a `AdvancedConnection` řeší propojování cest v síti. Také se zde nachází metoda pro převedení nalezené sítě do podoby vhodné k uložení do souboru.

10.8 Balík operací pro odstranění šumu

Tento balík slučuje všechny třídy, které nějakým způsobem slouží k odstranění šumu. Jsou tu třídy pracující s bitmapovým obrázkem nebo s rozpoznanou sítí. Oba tyto případy mají vlastí rozhraní. Všechny tyto třídy vycházejí z metod popsaných v kapitole 6.

`Dilate` – třída pro provádění dilatace v bitmapovém obrázku.

`Erode` – třída pro provádění eroze v bitmapovém obrázku.

`Closing` – třída, která za použití tříd `Dilate` a `Erode`, provádí uzavírání v bitmapovém obrázku.

`Cluster` – třída reprezentující skupinu pixelů. Slouží k uložení všech bodů jednoho shluku pixelů rozpoznaných jako značka.

`ClusterMerging` – tato třída, s využitím třídy `Cluster`, provádí slučování oblastí.

`DeleteLeftovers` – třída odstraňující zbytky po ztenčování.

`DeleteNoRoads` – v této třídě se provádí odstranění všech pixelů v obrázku, které nebyly rozpoznány jako cesta.

`RemoveTails` – třída sloužící k odstraňování šumu typu slepá ulička z rozpoznané sítě.

`MergeCloseNodes` – tato třída v síti hledá body, které leží těsně vedle sebe, a spojuje je do jednoho bodu.

`CycleRemoval` – třída která v síti hledá šum typu malé cykly a slučuje jejich body do jednoho.

10.9 Balík vizualizace

Tento balík seskupuje všechny třídy potřebné pro správné zobrazení GUI. Tyto třídy slouží pouze k ovládání a vizualizaci programu.

`Frame` – hlavní metoda z tohoto balíku. Vytváří okno programu a všechny jeho panely, kromě panelu s mapou. Navíc vytváří i menu programu a progress bar. Vytváří i ovládací prvky, kterým přidává funkčnost.

`ImagePanel` – panel okna zobrazující mapu. Umožňuje také pohyb po mapě a její přibližování a oddalování. Zároveň také přidává funkci kontextového menu pro vytváření konfigurací.

`ContextMenu` – třída vytvářejí kontextové menu. Menu vznikne v místě, kam uživatel klikne pravím tlačítkem, a umožní mu přidat barvu v dané pozici do seznamu barev cest nebo seznamu barev značek.

`ColorTableModel` – třída vytváří vlastní model tabulky pro zobrazení barev, jejich rozsahů a velikost okolí.

`ColorCellRenderer` – jednoduchá třída pro vytvoření specifické buňky tabulky, která místo textu zobrazuje barvu.

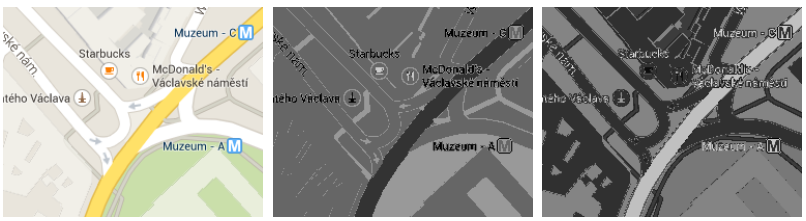
11 Testování

V této kapitole budou popsány výsledky testování některých částí programu, především pak těch nepoužitých, aby byly jasné důvody jejich nepoužití. Nejprve tedy budou ukázány výsledky algoritmu K-means používaného při předzpracování. Poté bude následovat testování aplikace jako celku, navržení způsobu bodování a dosažené výsledky v různých mapách.

11.1 K-means

Při testování metody K-means bylo zjištěno, že metoda nevyhovuje hned z několika důvodů. Hlavním důvodem však bylo to, že pro dostatečné rozlišení cest, značek a pozadí bylo nutné, aby počet středních hodnot K byl mezi 40 a 50. To jsou velmi vysoká čísla, která naznačují, že k příliš dobrému rozlišení nedošlo. Samozřejmě nešlo očekávat dokonalé rozlišení cest, značek a pozadí, ale takovýto výsledek byl až příliš špatný.

Dalším problémem je doba výpočtu, která i při nižších hodnotách K byla velice dlouhá a neúměrná získanému výsledku. Výsledky metody K-means je možné vidět na Obrázku 11.1.



(a) Originál (b) K = 5 (c) K = 20

(d) K = 30 (e) K = 40 (f) K = 50

Obrázek 11.1: Výsledky metody K-means. Zdroj [2]

11.2 Výsledky

V této kapitole bude zhodnocena přesnost výsledného programu pro generování struktury silniční sítě na základě rozpoznávání map ve formě bitmapových obrázků.

11.2.1 Způsob bodování

Každá rozpoznávaná mapa bude ohodnocena dvěma procentuálními hodnotami, kde první hodnota bude značit množství správně rozpoznaných cest a druhá množství chyb. Jako jedna cesta je definován úsek vedoucí od křižovatky nebo slepé uličky do křižovatky nebo slepé uličky. Jako slepé uličky jsou brány i cesty vedoucí na okraj mapy.

Při bodování se budou hledat hodnoty celkový počet rozpoznaných cest, počet nenalezených cest a počet cest, které byly nalezeny, ale nejsou to cesty. S těmito hodnotami je možné spočítat počet správně nalezených cest. Ten je roven počtu nalezených cest bez spatně určených cest:

$$spravne = nalezeno - spatne \tag{11.1}$$

Počet chyb na mapě je dán součtem nenalezených cest a špatně rozpoznaných cest:

$$chyb = nenalezeno + spatne \tag{11.2}$$

Celkový počet cest na mapě se určí jako součet nalezených cest bez cest učených špatně a cest nenalezených:

$$celkem = spravne + nenalezeno \tag{11.3}$$

Množství správně rozpoznaných cest je potom poměr správně rozpoznaných cest a celkového počtu cest:

$$uspesnost = \frac{spravne}{celkem} \tag{11.4}$$

Množství chyb vzniklých při rozpoznávání mapy je roven poměru nalezených chyb a počtu správně rozpoznaných cest:

$$chybovost = \frac{chyb}{nalezeno} \tag{11.5}$$

Důvodem proč jsou pro měření použity dvě hodnoty je, aby se rozlišil počet správně detekovaných cest a celková chybovost. Protože v případě, že byly rozpoznány všechny cesty (tedy 100% úspěšnost), mohou být rozpoznány i nějaké cesty navíc, které ve skutečnosti neexistují.

11.2.2 Úspěšnost

Podle výše uvedených výpočtů byly spočítány úspěšnosti rozpoznávání a jeho chybovost. Testování probíhalo na 12 různých mapách (3 pro OpenStreetMap, 3 pro Bing mapy, 3 pro Google mapy a 3 pro Mapy.cz) a bylo v různě osídlených oblastech. Oblasti byly tři: řídce osídlená oblast, hustě osídlená oblast a hustě osídlená oblast s větším oddálení. Oblasti stejného typu byly ze stejných míst. V příloze C jsou všechny testované mapy a v Tabulce 11.1 jsou vidět naměřené výsledky.

Je důležité si uvědomit, že výsledek je přímo ovlivněn kvalitou konfigurace. Při hůře nastavené konfiguraci mohou být výsledky horší a při lepší konfiguraci mohou být výsledky lepší. Nicméně při testování byla snaha o použití nejlepší možné konfigurace.

Mapa	Nalezeno	Nenalezeno	Špatně	Chyb	Správně	Celkem	Úspěšnost	Chybovost
osm_ld	36	2	1	3	35	37	94,5946%	8,3333%
osm_hd	52	1	3	4	49	50	98,0000%	7,6923%
osm_dist	93	0	2	2	91	91	100,0000%	2,1505%
bing_ld	50	1	0	1	50	51	98,0392%	2,0000%
bing_hd	41	2	1	3	40	42	95,2381%	7,3171%
bing_dist	61	1	1	2	60	61	98,3607%	3,2787%
google_ld	56	2	0	2	56	58	96,5517%	3,5714%
google_hd	49	2	1	3	48	50	96,0000%	6,1224%
google_dist	67	0	2	2	65	65	100,0000%	2,9851%
mapy_ld	86	1	6	7	80	81	98,7654%	8,1395%
mapy_hd	89	1	5	6	84	85	98,8235%	6,7416%
mapy_dist	304	2	33	35	271	273	99,2674%	11,5132%

Tabulka 11.1: Výsledky testování (osm jsou OpenStreetMap, bing jsou Bing mapy, google jsou Google mapy a mapy jsou Mapy.cz, ld znamená řídce obydlené oblast, hd je hustě obydlená oblast a dist je oddálená hustě obydlená oblast).

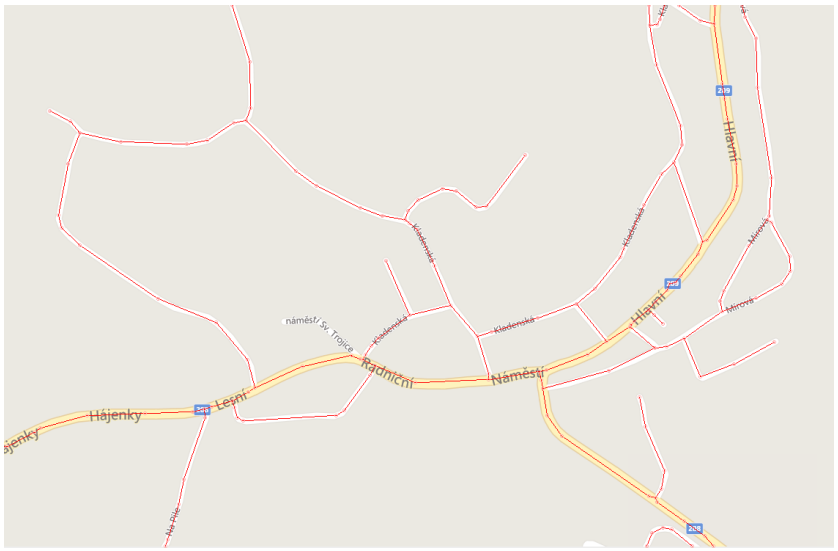
Ze získaných hodnot je možné spočíst průměrnou úspěšnost a průměrnou chybovost u jednotlivých mapových podkladů a celku. Spočtením poměru je

poté možná vidět celkovou kvalitu rozpoznávání pro daný typ map a celku viz Tabulka 11.2.

Typ	OSM	Bing mapy	Google mapy	Mapy.cz	Celkem
Úspěšnost	97,53%	97,21%	97,52%	98,95%	97,80%
Chybovost	6,06%	4,20%	4,23%	8,80%	5,82%
Poměr	6,2121%	4,3190%	4,3339%	8,8913%	5,9512%

Tabulka 11.2: Průměrné hodnoty výsledků testování.

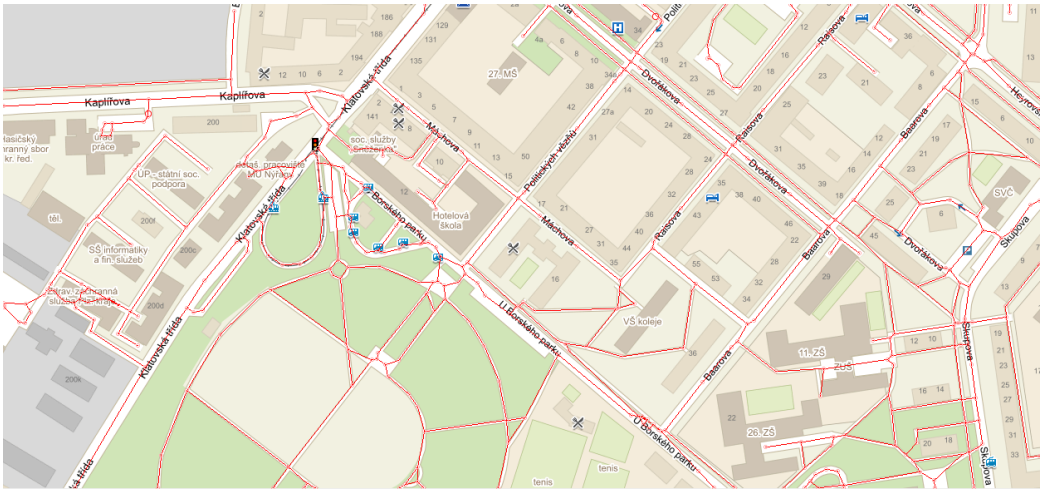
Je vidět, že nejlépe rozpoznávané mapové podklady jsou Bing mapy viz Obr. 11.2, což odpovídá kapitole 2.3, kde byly označeny jako nejlépe strojově čitelné. Nicméně je důležité si uvědomit, že samotná přesnost těchto map není dobrá. Druhé nejlépe rozpoznávané mapové podklady jsou Google mapy, na které byla tato aplikace především mířena.



Obrázek 11.2: Testovaná mapa s nejlepším výsledkem(červené čáry znázorňují nalezené cesty). Zdroj podkladu [3]

Dalším zajímavým úkazem jsou Mapy.cz, kde je největší úspěšnost rozpoznávání, je tedy nalezeno největší procento cest. Nicméně tyto výsledky jsou sráženy velkou chybovostí. Ta je způsobena především tím, že chodníky jsou zde brány jako cesty. To také zapříčiňuje větší množství nalezených cest ve stejném typu oblastí oproti ostatním mapovým podkladům. Proto pokud

by tyto chyby chtěl uživatel ručně odstranit, měl by mnohem více práce než u ostatních podkladů. Je to také důvod proč nejhůře rozpoznaná mapy je z mapových podkladů Mapy.cz viz Obr. 11.3



Obrázek 11.3: Testovaná mapa s nejhorším výsledkem(červené čáry znázorňují nalezené cesty). Zdroj podkladu [1]

Jinak je vidět, že celková úspěšnost rozpoznávání je velice vysoká. Teoreticky by mohla být i vyšší pokud by se vynechaly chyby, které vznikají u cest vedoucích mimo obrázek. Tyto cesty totiž většinou nejsou příliš významné a představují větší procento nenalezených cest. Nicméně chyba to je a proto byly do výsledků zahrnuty.

12 Závěr

Práce měla za úkol vytvořit aplikaci pro generování struktury silniční sítě na základě rozpoznávání bitmapových obrázku z běžně dostupných on-line mapových podkladů.

Nejdříve bylo nutné vybrat vhodné mapové podklady, ale díky jejich podobnosti byl navržen a vytvořen program, který je schopný zpracovávat mapové podklady z různých zdrojů. Problém odlišností barev v mapách byl vyřešen přidáním dalšího vstupního souboru, který obsahuje seznam barev cest a značek.

Před samotným vytvořením však bylo ještě nutné seznámit se s metodami pro zpracování obrazu a rozpoznávání. Pro samotné rozpoznání cest byl vytvořen vlastní algoritmus a metody pro zpracování obrazu byly použity pro odstranění šumu a pro ztenčování.

Následně bylo nutné převést obrázek do podoby bodů a vektorů. K tomuto účelu byl také vytvořen vlastní algoritmus. Po tomto převodu bylo nutné snížit počet bodů jednotlivých cest, proto bylo nutné seznámit se z metodami pro zjednodušení křivek.

Protože takto vytvořené rozpoznávání nebylo tak kvalitní, jak by být mohlo, byly vymyšlena pravidla a algoritmy pro doplnění cest, které nebyly nalezeny v předchozích krocích.

Závěrečné testy poté ukázaly, že rozpoznávání silniční sítě je velmi kvalitní. Pro všechny mapy bylo v testovacích mapách nalezeno přes 97% cest a pouze něco přes 5% nalezených cest bylo špatně, ale u některých konkrétních mapových podkladů jsou výsledky ještě lepší.

Aplikace má i potenciál k vylepšování. Jednou možností je propojení s některým z dostupných API mapových podkladů, která zde nebyla uvažována kvůli jejich svazujícím licenčním podmínkám. Další možností rozšíření by mohla být schopnost zpracovávat pole navazujících map a jejich propojování.

Na práci se pracovalo průběžně celý rok. Velké množství času bylo věnováno zkoumání algoritmů rozpoznávání, které byly považovány za klíčové. Nakonec se však ukázalo, že důležitější bylo navržení pravidel a algoritmů pro nalezení rozpojených cest. Přesto všechny body zadání byly splněny.

Literatura

[1] Seznam.cz a.s., "Mapy.cz." `https://www.mapy.cz`, citováno 1.5.2015.

[2] Google, "Google mapy." `https://www.google.cz/maps`, citováno 1.5.2015.

[3] Microsoft, "Bing mapy." `https://www.bing.com/maps/`, citováno 1.5.2015.

[4] "OpenStreetMap wiki." `http://wiki.openstreetmap.org/`, citováno 8.5.2015.

[5] "OpenStreetMap." `https://www.openstreetmap.org/`, citováno 1.5.2015.

[6] "OpenCycleMap." `http://www.opencyclemap.org/`, citováno 3.5.2015.

[7] "OpenPisteMap." `http://openpistemap.org/`, citováno 3.5.2015.

[8] E. Sülün, *Improvements in K-means Algorithm to Execute on Large Amounts of Data*. Izmir Institute of Technology, 2004.

[9] A. W. Robert Fisher, Simon Perkins and E. Wolfart, *Hypermedia Image Processing Reference*, 2000.

[10] T. Y. Zhang and C. Y. Suen, *A Fast Parallel Algorithm for Thinning Digital Patterns*, 1984.

[11] J. Hrnčířová, *Algoritmy geometrické generalizace pro PostGIS*. ČVUT v Praze, 2009.

[12] S. T. Wu and M. R. G. Marques, *A non-self-intersection Douglas-Peucker Algorithm*. State University of Campinas.

[13] M. Visvalingam and J. D. Whyatt, *Line Generalisation by Repeated El imination of the Smallest Area*. University of Hull, 1992.

A Uživatelská dokumentace

V této příloze budou popsány požadavky pro spuštění, způsoby spuštění aplikace a ovládání aplikace.

A.1 Požadavky a spuštění

Pro úspěšné spuštění aplikace je nutné mít nainstalovanou Javu minimálně ve verzi 1.7. Aplikaci je poté možné spustit otevřením souboru RozpoznavaniMap.jar.

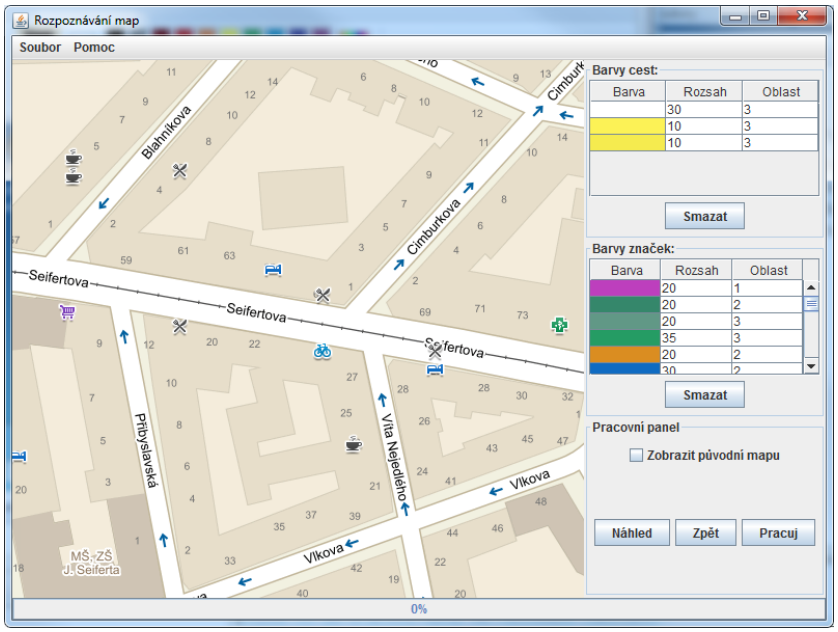
A.2 Ovládání aplikace

Ovládání aplikace by mělo být jednoduché a intuitivní, kromě vytváření konfigurace viz dále. Vzhled okna a rozmístění komponent je možné vidět na Obr. A.1.

V horní části okna se nachází menu ve kterém jsou tlačítka pro načítání a ukládání souborů. Jedná se o tlačítka *Otevřít*, *Uložit síť cest*, *Uložit konfiguraci*, *Načíst konfiguraci* a *Konec*. Tlačítko *Otevřít* umožní vybrat obrázek s mapou. Tlačítko *Uložit síť cest* umožní uložit rozpoznanou síť cest. Pokud ještě nedošlo k rozpoznání, nebude tlačítko aktivní. Tlačítka *Uložit konfiguraci* a *Načíst konfiguraci* umožňují ukládat a načítat konfigurační soubory. Tlačítko *Konec* ukončí aplikaci.

Dále okno obsahuje oblast pro zobrazení mapy, ta je prázdná dokud se nějaká mapa nenačte. Po načtení mapy se zde mapy bude vyskytovat a bude jí možné pohybovat na přibližovat. Pohyb mapy je možné provést stisknutím levého tlačítka myši na plochu mapy a tažením. Přibližování a oddalování je možné provádět pomocí kolečka myši.

V pravé části okna se pak nachází pracovní panel, ten je rozdělen na tři části. V první části je seznam barev cest, v druhé seznam barev značek a v třetí ovládací tlačítka. První a druhá část se využívá při nastavování konfigurace viz dále. Třetí část obsahuje tři tlačítka a jedno zaškrtávací pole.



Obrázek A.1: Ukázka aplikace.

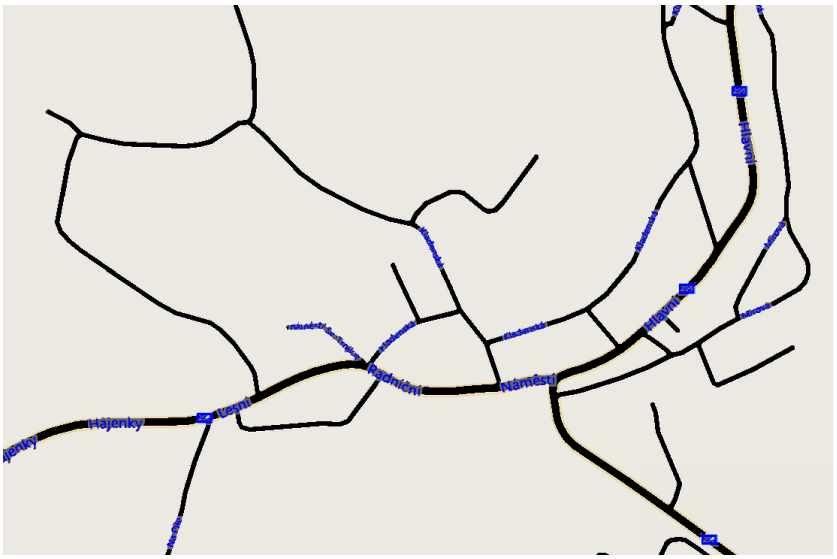
Tlačítko *Náhled* složí k ukázce, jak aktuální konfigurace předzpracovává obrázek s mapou. Tlačítko *Pracuj* spustí rozpoznávání struktury silniční sítě. Průběh předzpracování a rozpoznávání je možné sledovat na progress baru v dolní části okna. Tlačítko *Zpět* vrátí provedené akce. Zaškrtávací pole *Zobrazit původní mapu* slouží k vykreslení původní mapy na pozadí. Tato akce je viditelná až po předzpracování nebo rozpoznávání.

A.3 Vytváření konfiguračního souboru

V aplikaci je možné vytvořit vlastní konfigurační soubor. V pravé části okna se nachází dva seznamy. První seznam zobrazuje barvy cest a druhý barvy značek. Po spuštění aplikace jsou oba tyto seznamy prázdné a je možné je naplnit načtením konfiguračního souboru nebo přidáním vlastních barev.

Vlastní barvy je možné přidat tak, že se v mapě klikne pravím tlačítkem na barvu, kterou chce uživatel přidat. Po kliknutí ze zobrazí kontextové menu, která dává uživateli možnost, přidat barvu do seznamu cest nebo do seznamu barev. Po kliknutí na jednu z možností se barva přidá do zadaného seznamu.

U každé barvy jsou v seznamu další dva parametry. Parametr *Rozsah* ur-

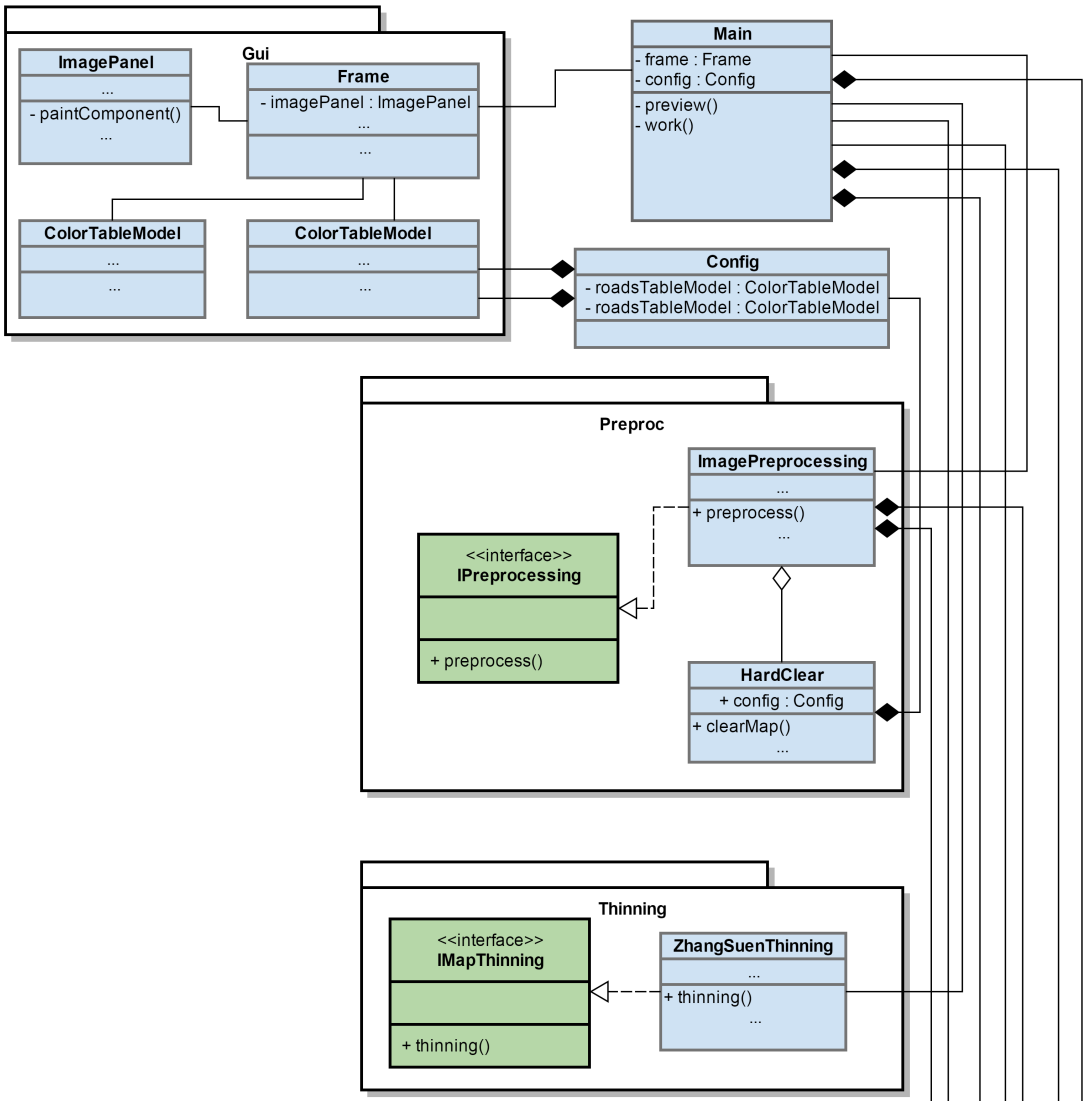


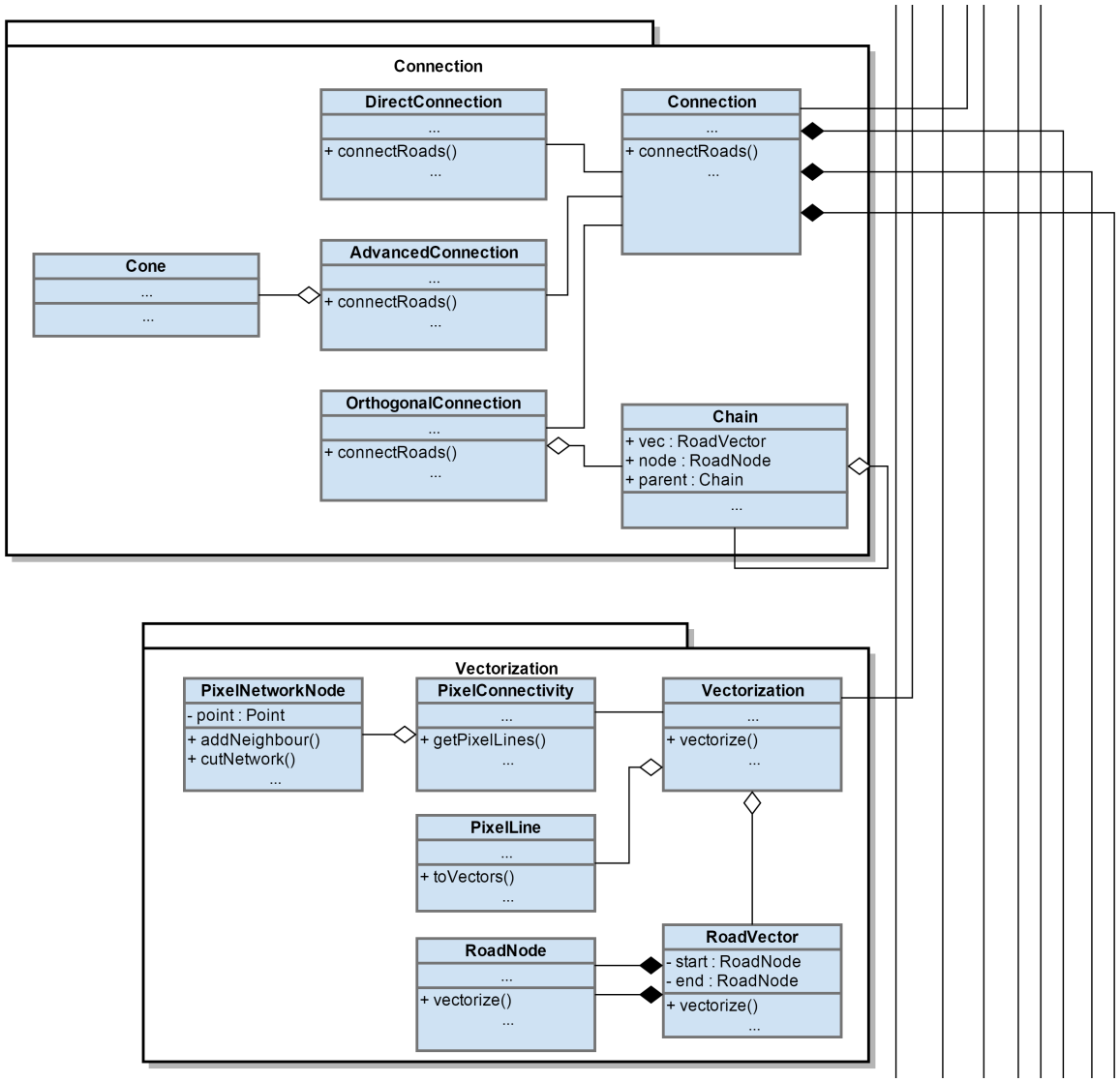
Obrázek A.2: Ukázka správného působení konfiguračního souboru.

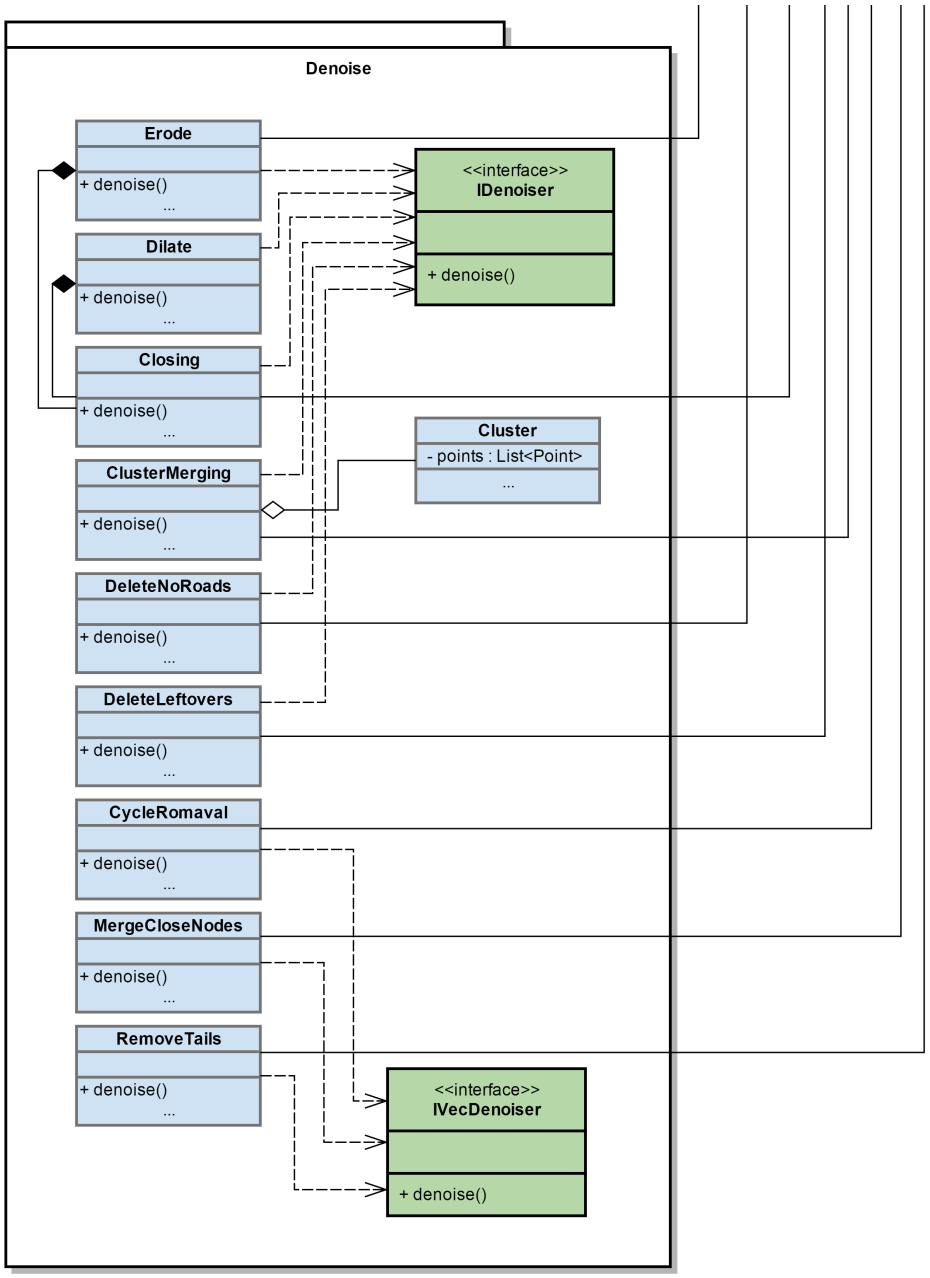
čuje barevný rozsah barvy při rozpoznávání, čím větší číslo tím větší rozsah. Například přidání žluté barvy s rozsahem 0 bude přijímat pouze zadanou barvu a zadání barvy s rozsahem 10 bude přijímat i barvy velmi podobné žluté. Parametr *Oblast* určuje rozsah působnosti barev značek. To znamená, že pixel barvy cesty bude rozpoznán jako značka, pokud se nachází ve vzdálenosti zadané v parametru od pixelu s barvou značky.

Vliv přidávání barev je možné zkontrolovat tlačítkem *Náhled*. Zde je žádoucí, aby cesty byly pokryté černou barvou a všechny značky šedou a modrou. Značky mají přednost před cestami, proto v místě, kde se nachází cesta i značka, by mělo být pokrytí šedé a modré viz Obr. A.2.

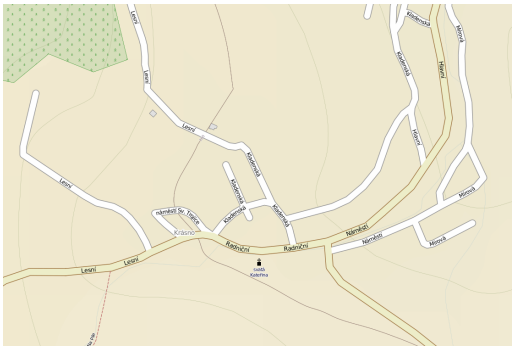
B Zjednodušený diagram tříd



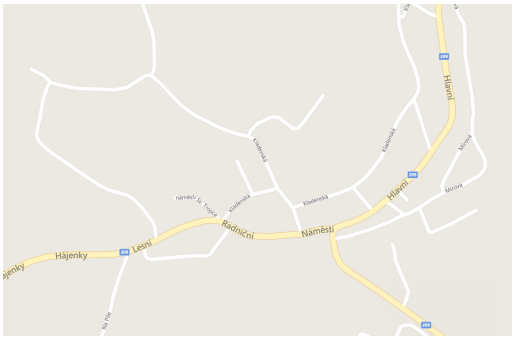

Connection
DirectConnection
...
+ connectRoads()
...
Connection
...
+ connectRoads()
...
Cone
...
...
AdvancedConnection
...
+ connectRoads()
...
OrthogonalConnection
...
+ connectRoads()
...
Chain
+ vec : RoadVector
+ node : RoadNode
+ parent : Chain
...
Vectorization
PixelNetworkNode
- point : Point
+ addNeighbour()
+ cutNetwork()
...
PixelConnectivity
...
+ getPixelLines()
...
Vectorization
...
+ vectorize()
...
PixelLine
...
+ toVectors()
...
RoadNode
...
+ vectorize()
...
RoadVector
- start : RoadNode
- end : RoadNode
+ vectorize()
...


Denoise
Erode
+ denoise()
...
Dilate
+ denoise()
...
Closing
+ denoise()
...
ClusterMerging
+ denoise()
...
DeleteNoRoads
+ denoise()
...
DeleteLeftovers
+ denoise()
...
CycleRomaval
+ denoise()
...
MergeCloseNodes
+ denoise()
...
RemoveTails
+ denoise()
...
<<interface>>
IDenoiser
+ denoise()
Cluster
- points : List<Point>
...
<<interface>>
IVecDenoiser
+ denoise()

C Přehled testovacích map



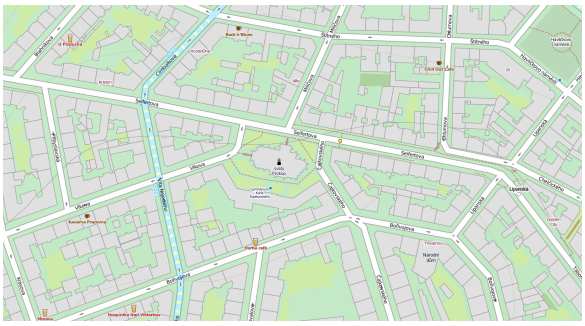
(a) test_1_1



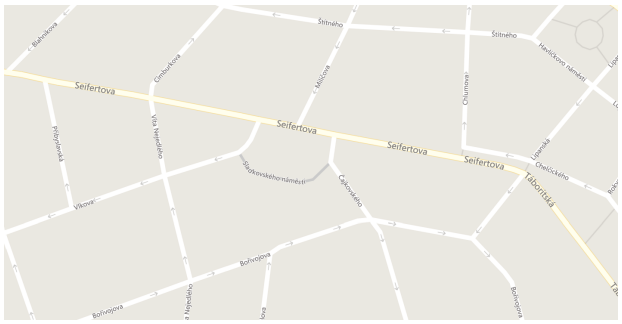
(b) test_2_1

(c) test_3_1

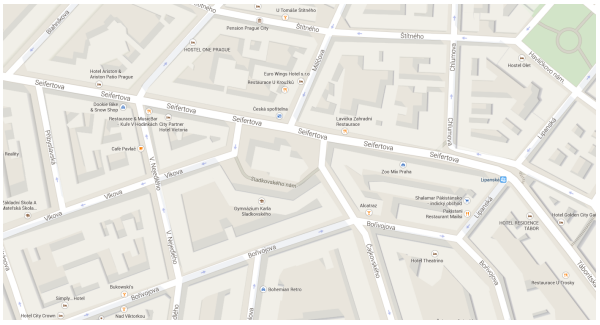
(d) test_4_1



(a) test_1_2

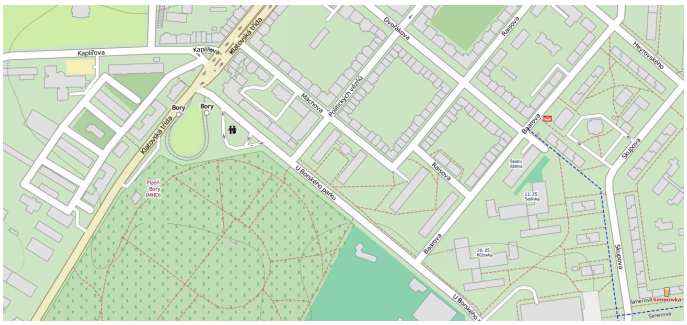


(b) test_2_2

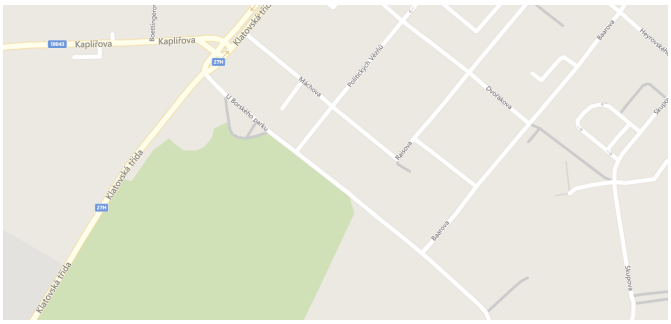


(c) test_3_2

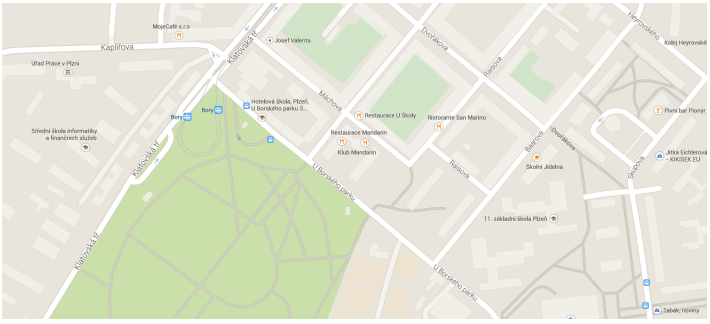
(d) test_4_2



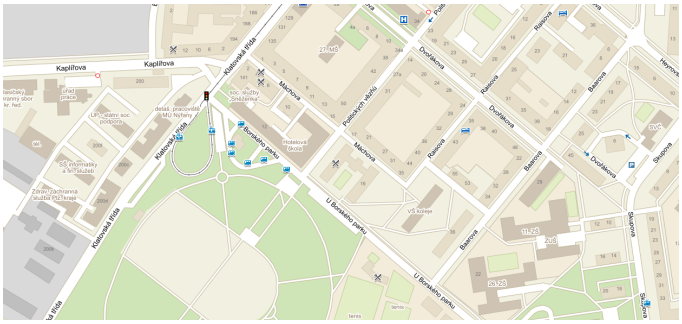
(a) test_1_3



(b) test_2_3



(c) test_3_3



(d) test_4_3

D Obsah CD

Složka ”doc”:

- Obsahuje JavaDoc programu.

Složka ”spustitelne”:

- Obsahuje spustitelný program ”RozpoznavaniMap.jar”. K jeho spuštění je nutné mít nainstalovanou Javu minimálně verze 1.7.

Složka ”konfiguracni soubory”:

- Obsahuje několik konfiguračních souborů, které se v aplikaci dají použít.

Složka ”testovaci mapy”:

- Obsahuje několik map, které byly použity při testování.

Složka ”src”:

- Obsahuje složku, ve které jsou všechny zdrojové soubory a
- soubor build.xml, který slouží k přeložení programu.

Složka ”texty”:

- Obsahuje text diplomové práce.