VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POKROČILÉ METODY DETEKCE HRAN V OBRAZE

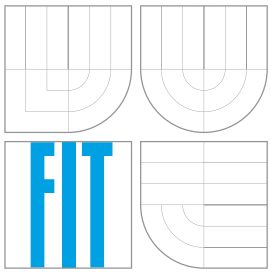
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN MEZÍRKA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POKROČILÉ METODY DETEKCE HRAN V OBRAZE

ADVANCED IMAGE EDGE DETECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN MEZÍRKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2015

Abstrakt

Cílem této práce je zjistit možnosti aplikování trénovatelného algoritmu detekce hran Structured forest for fast edge detection na extrakci informací z historických mapových podkladů a medicínských snímků. Pro práci byl vytvořen vlastní anotovaný dataset, na kterém byl otestován tento trénovatelný detektor. Structured forest v porovnáním s klasickými detektory dosahoval lepších výsledků na mapových podkladech. Úspěšnost nalezení hran kostních tkání byla u obou přístupů podobná. Následující zaměření práce je orientováno na porovnání různých stylů anotování obrázků, experimentování s datasetem, včetně určování parametrů a vyhodnocování úspěšností metod.

Abstract

The goal of this work is to investigate options how to apply trainable edge detection algorithm Structured forest for fast edge detection to information extraction from historical maps and medical images. For the work, annotated dataset was created and the detector was tested on it. Structured forest achieved better results on map data, compared with classical detectors. Success rate of finding edges of bones was similar at both approaches. Aim of the work is focused on comparing different image annotation styles, experiments with dataset, including determining parameters and evaluation of the methods.

Klíčová slova

Detekce hran, anotační nástroj, anotovaný dataset hran.

Keywords

Edge detection, annotation tool, boundary dataset.

Citace

Martin Mezírka: Pokročilé metody detekce hran v obraze, diplomová práce, Brno, FIT VUT v Brně, 2015

Pokročilé metody detekce hran v obraze

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Michala Španěla, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Mezírka
27. května 2015

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Michalu Španělovi, Ph.D. za rady a připomínky při konzultacích, kterými mi pomáhal při vypracování této práce.

© Martin Mezírka, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Proč je důležité zajímat se o detekci hran? Hrany v klasickém pojetí můžeme charakterizovat jako prudkou změnu intenzity jasu v obraze. K takovým změnám často dochází na rozhraní dvou objektů ve scéně, nebo mezi objektem a pozadím scény. Jinými slovy fyzické umístění objektů může být naznačeno prudší změnou jasu, ale také nemusí. Širšímu pojetí problému detekce hran odpovídá nalezení skutečných hranic objektů. Chceme označit hranu u malého – nevýznamného objektu ve scéně? Chceme postihnout hranice stínů? Určit tyto a další případy je typicky velmi těžké a bývá vyžadováno zpracování dodatečné vysoce úrovňové informace. Nicméně hrany se nejčastěji objevují v místech, které jsou pro člověka nějakým způsobem zajímavé a důležité, stejně tak jsou důležité i pro další zpracování obrazu. A proto je vhodné mít informaci o poloze hran [4].

Cíl práce spočívá v prozkoumání možností trénování detektoru *Structured forest for fast edge detection* v závislosti na volbě různých stylů anotací dat a vlivu parametrů na úspěšnost detekce hran. Experimenty byly prováděny na dvou speciálních datasetech. První je tvořen mapovými podklady z počátku 20. století, ze kterých se budou zjišťovat možnosti extrakce jednotlivých mapových elementů. Druhý dataset se skládá z CT snímků hlavy, na kterých bude aplikován trénovatelný detektor s cílem zvýraznit pouze určité hrany patřící kostní tkáni. Pro účel tvorby vlastního anotovaného datasetu byl vytvořen grafický anotační nástroj. Nástroj pracuje s odpovídajícím formátem datasetu univerzity Berkeley.

V rámci práce vznikla demonstrační implementace některých metod jako Marr - Hildreth, Cannyho, Derichův hranový detektor a filtrace anizotropickou difuzí. Tyto metody byly také aplikovány nad uvedenými úlohami, se snahou přiblížení se k výsledkům trénovatelných detektorů pomocí vhodné volby parametrů. V extrakci mapových elementů bylo možné volbou parametrů částečně zlepšit výsledek odstraněním vrstevnic, nicméně v této oblasti měl trénovatelný detektor výrazně lepší výsledky. V případě reakce na kostní tkáň bylo možné dosáhnout klasickými detektory podobné odezvy, jako při použití trénovatelného detektoru.

Vedlejším výsledkem této práce je návod pro použití detektoru Structured forest, jehož implementace v Matlabu je dostupná na webových stránkách Microsoft Research [8].

V úvodní kapitole Detekce hran v obraze jsou nastíněny základní principy a problémy, které se objevují při detekci hran. Následující kapitola pokračuje představením současných pokročilejších metod a také informuje o způsobu vyhodnocování úspěšností. Mezi současnými metodami je představen aktuální trénovatelný detektor Structured forest, nebo metoda EdgeFlow. Další kapitola se věnuje možnostem vytváření datasetu a volby stylu anotace hran. Kapitola o implementaci popisuje vytvořený anotační nástroj, program implementující některé konvenční detektory a další detaily, které je potřeba znát pro použití trénovatelného detektoru. V poslední části jsou shrnuty experimenty a porovnání výsledků.

Kapitola 2

Detekce hran v obraze

Kapitola představuje úvodní pojmy a přehled prvních, ale i pokročilejších přístupů k řešení problému z oblasti detekce hran. Část kapitoly se věnuje šumu v obraze. Možnostem odstranění šumu společně s následným vlivem na výslednou kvalitu detekce hran. Původní obrázky, na kterých jsou demonstrovány vybrané metody, pocházejí z *Berkeley Segmentation Dataset* [7].

2.1 Typy metod

Hrana je nejčastěji charakterizována jako ostrá změna jasové složky. Rané metody detekce hran jsou založeny na prostém provádění rozdílů jasových hodnot v obraze (tzv. diferenciace pixelů). Nejznámější a nejpřímočařejší metody využívají první derivaci jasové složky obrazu s následným eventuálním prahováním. Derivace se aproximuje v malém okolí obrazu pomocí konvoluce. Velikosti konvolučních jader se typicky volí 3×3. Konvoluční jádra nesou jména po svých tvůrcích (Sobel, Roberts, Prewitt, ...).

Původní algoritmy byly následně upravovány a zdokonalovány a na jejich základech stojí i nejznámější Cannyho detektor. Metody založené na zpracování gradientu intenzity pixelů tvoří hlavní směr detekce hran. Tyto detektory pracují v jasové oblasti spolehlivě, ovšem objevily se nové požadavky směrem k detekci s nastavitelnou jemností hran. Aby detektor dokázal reagovat dle nastavení pouze na nejvýznamnější hrany, nebo naopak dokázal zachytit jemnější detaily, nebo v ideálním případě, aby si detektor zvolil optimální parametr rozlišení sám. Tato vlastnost detektorů se nazývá *Multiscale*. Multiscale chování detektorů lze napodobit vyhlazováním obrazu na více úrovních. Větší úroveň vyhlazení odstraní jemné hrany, ale zanechá výraznější. Protože vyhlazováním obrazu se navíc redukuje i jeho šum, je multiscale přístup popsán podrobněji v části zabývající se vlivem šumu v obraze 2.4.

Existují další metody, které berou v úvahu i jiné obrazové příznaky než pouze intenzitu pixelů. Takovou technikou je například *EdgeFlow* (Wei-Ying Ma, B. S. Manjunath). EdgeFlow kombinuje dohromady jednotlivé „toky hran“ intenzity/barvy, textury a fáze [6]. Stejně jako v případě gradientních metod, může být i EdgeFlow používána jako multiscale metoda.

Doposud všechny uvedené přístupy patřily mezi obecné metody. Pokud máme jasně definovaný problém, lze využít trénovatelné detektory hran.

2.2 Cannyho kritéria

John F. Canny definoval již v roce 1986 ve článku *A Computational Approach to Edge Detection* tři významná kritéria, která by měla být při detekci hran brána v úvahu. Přesněji řečeno, měla by být maximalizována.

1. Poměr signálu k šumu (Signal-to-Noise Ratio) – detektor by měl více odpovídat na skutečnou hranu než na přítomný šum v obraze (maximalizace SNR). Neboli snaha o maximalizaci pravděpodobnosti detekování skutečné hrany a o minimalizaci pravděpodobnosti planých poplachů.

2. Lokalizace – pozice nalezené hrany by se měla co nejvíce blížit pozici skutečné hrany.

3. Jedinečná odezva – jedna skutečná hrana nezpůsobí více než jednu odezvu.

Ve stejném článku popsal J. F. Canny hranový detektor, který se tyto kriteria snaží maximalizovat [1]. Jedná se o nejznámější hranový detektor, který nese jméno po svém vynálezci a jehož algoritmus je popsán v sekci 2.3.

2.3 Základní metody

V této části jsou uvedeny historicky první pokročilejší přístupy k detekci hran.

Marr - Hildreth

V roce 1980 byl publikován způsob detekce hran založený na Laplaceovém operátoru (Laplacian). Laplaceův operátor pro dvě dimenze nad intenzitami obrazu I o souřadnicích x, y je definován rovnicí 2.1. Obvykle při detekci hran bývá aplikován spolu s vyhlazovacím filtrem (např. Gaussův filtr).

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial^2 x} + \frac{\partial^2 I(x,y)}{\partial^2 y} \tag{2.1}$$

Nejmenší diskrétní konvoluční jádra, která aproximují druhou derivaci laplaciánu jsou uvedeny v příkladu 2.2. Po aplikaci operátoru se problém detekce hran změní na problém detekcí průchodů nulou.

$$H = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \begin{pmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{pmatrix} \tag{2.2}$$

Těchto průchodů bývá ale velké množství, proto je vhodné ověřit, zda odpovídající body mají dostatečnou hodnotu gradientu. Hrany mají tendenci tvořit uzavřené linie. Marr-Hildreth algoritmus často produkuje falešné hrany [4]. Lepších výsledků lze dosáhnout pomocí Cannyho detektoru.

Cannyho hranový detektor

Nejznámější hranový detektor byl vytvořen v roce 1986 [1]. Celý algoritmus se skládá z pěti základních fází, které lze rozdělit do následujících částí:

1. eliminace šumu,

2. nalezení gradientu,
3. nalezení lokálních maxim,
4. dvojité prahování,
5. odstranění nevýznamných hran.

Eliminace šumu

K odstranění šumu ze vstupního obrázku se používá prostý Gaussův filtr. Více informací o šumu a jeho vlivu na detekci hran lze nalézt v sekci 2.4.

Hledání gradientu

K nalezení gradientu lze využít konvoluci, například s uvedenými Sobelovými jádry v horizontálním a vertikálním směru 2.3.

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{2.3}$$

Hodnoty gradientu, které označují hranu, mohou dle typu přechodu hrany nabývat jak záporných tak i kladných hodnot. Výsledná velikost vektoru gradientu neboli sílu hrany lze určit z dílčích směrů pomocí 2.4, nebo 2.5.

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{2.4}$$

$$G(x,y) = |G_x(x,y)| + |G_y(x,y)| \tag{2.5}$$

Až doposud se postup Cannyho detekce příliš nelišil od předchozích přístupů založených na konvoluci. Změna nastává v dalších částech algoritmu. Další krok provádí zúžení širokých (neostrých) hran.

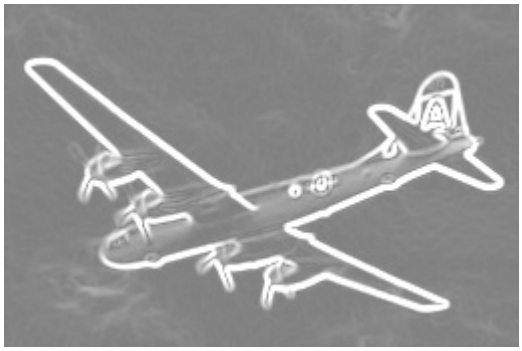
Potlačení nemaximálních hodnot

Tento krok v podstatě říká, že se snažíme zachovávat lokální maxima ve směru vypočítaného gradientu. Všechny ostatní hodnoty se potlačují. Směr gradientu θ lze vypočítat z rovnice 2.6.

$$\theta(x,y) = \arctan \frac{G_y(x,y)}{G_x(x,y)} \tag{2.6}$$

Potlačení nemaximálních hodnot je založeno na prohledávání blízkého okolí hodnoty gradientu, kde hledáme lokální maxima. Jako okolí jednoho pixelu volíme jeho nejbližší okolí ve směru gradientu θ a to jak v kladném tak i v záporném směru.

Význam kroku potlačení nemaximálních hodnot lze vidět na obrázcích 2.1. Na prvním jsou zobrazeny hodnoty gradientu v obou směrech (vertikálním, horizontálním) po aplikaci euklidovské metriky (rovnice 2.4). Hrany jsou reprezentovány bílou barvou a jsou velmi neostré. Výsledek s mnohem přesnější lokalizací hran je na obrázku 2.1b.



(a) Velikost gradientu (síla hrany).

(b) Tenké hrany po potlačení nemaximálních hodnot.

Obrázek 2.1: Potlačení nemaximálních hodnot gradientu.

Dvojité prahování

Cílem je detekovat pouze opravdové (výrazné) hrany. Na předchozí ukázce sice vidíme, že lokalizace hran se zlepšila, ale stále si můžeme všimnout spousty náznaků falešných hran. Tyto hodnoty gradientu poukazující na falešné hrany nejsou veliké, nejčastěji mohou představovat šum v obraze nebo nepatrné změny nuance v jasu, které ale za hrany obecně nepovažujeme. Řešení, které se samo nabízí, je využití prahování. Ale abychom předešli chybě při špatně zvoleném prahu a neodstranily kromě šumu i méně zřetelné hrany, Cannyho detektor používá dvojité prahování.

Obrázek 2.2: Dvojité prahování.

Na obrázku 2.2 jsou označeny hlavní (silné) hrany bílou barvou a vedlejší (slabé) hrany šedou barvou. Předpokládáme, že silné linie odpovídají skutečným hranám v obraze, ale u slabých hran si nejsme jistí. Rozhodnutí o tom, které slabé hrany zahrneme do konečného výsledku, pojednává následující sekce.

Odstranění nevýznamných hran (hystereze)

Nevýznamné hrany určené k odstranění učiníme podle logického závěru, kdy předpokládáme, že nevýznamné hrany nenavazují na silné. Z obrázku 2.2 (dvojité prahování) vychází obrázek 2.3a, kde jsou červeně označeny všechny slabé hrany, které se dotýkají silných hran. Šedá barva reprezentuje slabé hrany určené k odstranění. Uvedený příklad návaznost dvou linií definoval skrze odpovídající hodnoty bodů v jejich nejbližším osmiokolí.

(a) Odstranění nevýznamných hran (hystereze). (b) Konečný výsledek.

Obrázek 2.3: Hystereze a výsledný obrázek Cannyho detektoru.

Obrázek 2.3b přestavuje konečný výstup Cannyho hranového detektoru. Vedlejší červené hrany se změnily na silné a šedé (vedlejší) hrany byly odstraněny.

2.4 Šum

Reálný obraz není ideální, šum je v určité míře všudypřítomný v každém obrázku. Vzniknout může při snímání obrazu kamerou, přenosu nebo zpracování obrazu. Ať už se jedná o tepelný nebo kterýkoli jiný šum, je celkem přirozené, že šum způsobuje problémy při detekci hran. Velká část metod se spoléhá v menší či větší míře na výpočet jasového gradientu v obraze, například pomocí diferenciace pixelů (konvoluce). Provedením diferenciací šumu v obraze zvýrazníme vysoké frekvence, což bude mít za následek náhodně se objevující velké odchylky ve velikostech gradientu [4].

Jako obecný přístup k redukci šumu se vžila filtrace obrazu Gaussovým filtrem. Jedná se o konvoluci obrazu s maskou, která je složená z hodnot 2D Gaussovy funkce. Nastavitelné parametry filtru jsou dva: směrodatná odchylka a velikost masky filtru. Volba těchto parametrů závisí na míře přítomnosti šumu v obraze. Čím je šum výraznější, tím je větší potřeba rozmazání obrazu. To je ale při detekci hran problém. Hrany se stávají neostrými. Je problematické určit přesnou polohu hrany, a proto se porušuje kritérium *lokalizace* hran, které zavedl J. F. Canny (sekce 2.2).

Zvyšující se rozmazání obrazu má kromě zhoršené lokalizace ještě jeden efekt. V případě například Cannyho detektoru dochází k mizení jemnějších hran. Chování Cannyho detektoru se tedy může jevit jako multiscale metoda.

Alternativa ke Gaussovu filtru může být například použití pokročilejších metod anizotropní difuze (Perona - Malik), nebo Derichova detektoru. Anizotropní difuze potlačuje vyhlazování obrazu v okolí hran [9]. Nedochází proto k výraznému zhoršení kvality hrany. Interpretaci této metody si lze představit jako aplikaci Gaussova vyhlazení s adaptivní změnou směrodatné odchylky. Pixely v blízkosti hran jsou vyhlazovány Gaussovým filtrem méně než pixely nacházející se v homogenních oblastech.

Příklady detekce hran ve větším měřítku rozlišení a tedy při použití výraznějšího vyhlazení jsou uvedeny v následující části na obrázku 2.4.

2.5 Derichův hranový detektor

Pouze rok po představení Cannyho hranového detektoru se objevuje Derichův detektor (Rachid Deriche 1987). Algoritmus je opět vícekrokový a vychází z Cannyho algoritmu, který vylepšuje. Rozdíl detektorů spočívá hlavně v prvních dvou krocích *eliminace šumu* a *nalezení gradientu*.

Na rozdíl od Cannyho detektoru, který používá Gaussovo vyhlazení (filtr s konečnou impulzní odezvou). Derichův detektor používá filtr s nekonečnou impulzní odezvou. Filtr obsahuje zpětnovazební smyčku, kdy k výpočtu aktuální výstupní hodnoty využívá několik předcházejících výstupů.

Nejprve se provádí filtrace přes všechny řádky zleva doprava (2.7), poté zprava doleva (2.8), přičemž nezáleží na pořadí. Výsledek těchto dvou filtrací se zkombinuje do pomocné matice λ (2.9). V uvedených rovnicích proměnná x odkazuje na vstupní obraz, index i se vztahuje k řádkům obrazu a index j ke sloupcům.

$$y^{(1)}_{i,j} = a_1 x_{i,j} + a_2 x_{i,j-1} + b_1 y^{(1)}_{i,j-1} + b_2 y^{(1)}_{i,j-2} \tag{2.7}$$

$$y^{(2)}_{i,j} = a_3 x_{i,j+1} + a_4 x_{i,j+2} + b_1 y^{(2)}_{i,j+1} + b_2 y^{(2)}_{i,j+2} \tag{2.8}$$

$$\lambda_{i,j} = c_1 (y^{(1)}_{i,j} + y^{(2)}_{i,j}) \tag{2.9}$$

Nad výsledkem předchozího kroku – maticí λ je aplikována filtrace ve vertikálním směru shora dolů (2.10), poté zdola nahoru (2.11). Dílčí výsledky jsou následně zkombinovány do výsledného obrazu Θ (2.12).

$$y^{(1)}_{i,j} = a_5 \lambda_{i,j} + a_6 \lambda_{i-1,j} + b_1 y^{(1)}_{i-1,j} + b_2 y^{(1)}_{i-2,j} \tag{2.10}$$

$$y^{(1)}_{i,j} = a_7 \lambda_{i+1,j} + a_8 \lambda_{i+2,j} + b_1 y^{(1)}_{i+1,j} + b_2 y^{(1)}_{i+2,j} \tag{2.11}$$

$$\Theta_{i,j} = c_2 (y^{(1)}_{i,j} + y^{(2)}_{i,j}) \tag{2.12}$$

Koeficienty a, b, c, k se liší v závislosti výpočtu horizontálních, nebo vertikálních složek gradientů. Koeficienty mají předem definovaný postup výpočtu na základě vstupního parametru α, hodnoty parametrů jsou uvedeny v dokumentu [10].

Výhodou Derichova filtru je přítomnost pouze jednoho nastavitelného parametru α, který ovládá míru vyhlazení obrazu. Pro málo zašuměné obrázky se volí větší parametr, přibližně $\alpha > 1$. Naopak pro potřebu většího vyhlazení a tedy odstranění málo výrazných hran se α nastavuje na menší hodnoty blížící se k nule. Další výhoda Derichova přístupu spočívá v konstantním čase zpracování obrazu nezávisle na požadované hodnotě α tj. míře vyhlazení obrazu, na rozdíl od Gaussova filtru [10].

Obecně Derichův přístup vede k lepším výsledkům detekce hran než původní Cannyho algoritmus. Na obrázku 2.4 byly porovnány tři přístupy k vyhlazování vstupních dat při velké míře rozmazání. Jak je vidět z obrázků, příliš veliké Gaussovo zprůměrování má negativní vliv na kvalitu nalezených hran a proto základní Cannyho detektor není považován za multiscale metodu. Naopak filtrování pomocí Anizotropické difuze a Derichovým filtrem vedlo k lepším výsledkům. Obě metody lze považovat za multiscale přístupy. V případě Derichova detektoru tedy volbou jediného parametru α ovládáme množství detailů, na které detektor reaguje. Nevýhodou anizotropické difuze je iterační způsob výpočtu a přítomnost až třech nastavitelných parametrů (K – ostrost hran, počet iterací a λ – intenzita vyhlazování).

(a) Vyhlazení obrázku – Cannyho hranový detektor s parametry $kernelSize = 13, \sigma = 9$.

(b) Potlačení nemaximálních hodnot – Cannyho hranový detektor s parametry $kernelSize = 13, \sigma = 9$.

(c) Vyhlazení obrázku – Derichův detektor s parametrem $\alpha = 0,3$.

(d) Potlačení nemaximálních hodnot – Derichův detektor s parametrem $\alpha = 0,3$.

(e) Vyhlazení obrázku – anizotropní difuze s parametry $K = 1, iterace = 300, \lambda = 0,05$.

(f) Potlačení nemaximálních hodnot – anizotropní difuze s parametry $K = 1, iterace = 300, \lambda = 0,05$.

Obrázek 2.4: Příklady zanedbání jemných hran.

Kapitola 3

Současný přístup k detekci hran v obraze

Kapitola navazuje na předchozí část, která byla zaměřena hlavně na obecný úvod a první metody založené na hledání lokálních maxim v gradientu intenzity pixelů. Jsou zde uvedeny novější a robustnější metody, které mají tendenci postihnout kromě hran získaných při zkoumání změn intenzity obrazu i skutečné hranice objektů. Tento postup vede k detekci hran, které jsou bližší k vnímání člověka. V kapitole jsou dále zmíněny postupy, které se používají při dnešním vyhodnocováním úspěšností metod.

3.1 Metoda EdgeFlow

V úvodu bylo řečeno, že hranice mezi běžnými objekty v obecných obrázcích nejsou a ani nemohou být rozlišitelné pouze prostřednictvím porovnáváním intenzity pixelů. EdgeFlow bere v úvahu kromě jasového gradientu také další obrazové příznaky. Hrany v reálných datech jsou mimo jiné tvořeny změnami v barvě (intenzitě), textuře, nebo fázi. Proto algoritmus kombinuje všechny tyto příznaky s cílem dobré detekce hran a segmentace obrazu v reálných datech. Základní princip spočívá ve zkonstruování v každém bodě obrazu tok vektorů hran (*edge flow*). Hrany – obecně hranice mezi objekty, lze nalézt na pozicích, kde se střetávají dva opačné směry těchto toků. Postup algoritmu je následovný:

- V každém bodě se vypočte lokální energie (například známá velikost změny intenzity) a odhadne se směr tohoto toku.
- Iteračním výpočtem se propaguje tato energie do sousedních bodů v závislosti na směru toku v daných bodech.
- Propagace energie se zastaví, pokud sousední body mají opačný směr vektorů. Algoritmus iteruje, dokud se nedosáhne stabilního stavu.

Rovnice 3.1 definuje vektor toku F v obrázku na pozici s s orientací vektoru θ.

$$F(s,\theta) = F[E(s,\theta), P(s,\theta), P(s,\theta+\pi)] \tag{3.1}$$

$E(s,\theta)$ představuje celkovou edgeflow energii. $P(s,...)$ představuje pravděpodobnost nalezení hranice od pozice s ve směru θ respektive opačného směru. Tyto komponenty se vypočítají z dílčích částí obrazových atributů váženým průměrem s váhou $w(a)$ (rovnice 3.2).

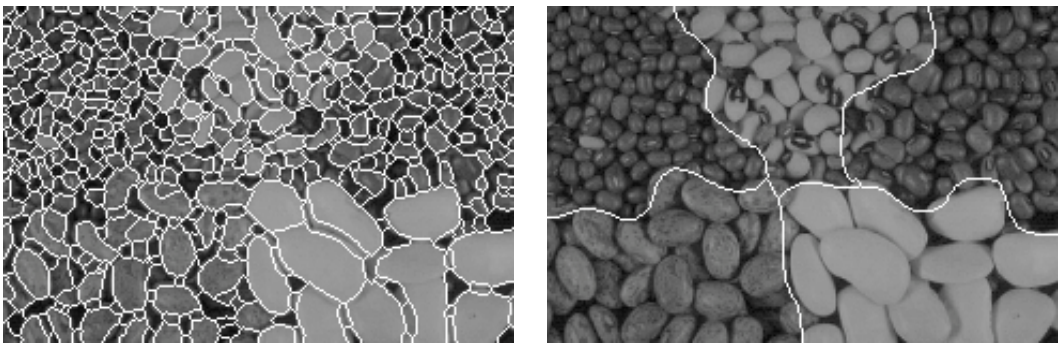
Množina A může obsahovat například tyto hodnoty obrazových atributů: intenzita, textura a fáze. Výpočet výsledného $P(s, \theta)$ je obdobný.

$$E(s, \theta) = \sum_{a \in A} E_a(s, \theta) w(a), \quad \text{kde} \sum_{a \in A} w(a) = 1 \tag{3.2}$$

Výpočet texturového toku se provádí na základě gradientu v texturových příznacích. Pro extrakci texturových příznaků se používají Gaborové vlnky. Banka Gaborovych filtrů se generuje dle velikosti parametru rozlišení σ. Po filtraci všemi filtry se získá vektor texturových příznaků, který se přepočítá na energii v textuře. Informaci o fázi lze zjistit také z výstupu Gaborového filtru. Využití fázové informace v reálných obrazech sice není příliš velké, ale tyto příznaky dokážou postihnout dokonce pouze iluzorní hrany (lze si je typicky představit jako ohraničení posunuté výseče vyšrafované oblasti).

Po výpočtu $F(s)$ následuje iterativní propagování toku hran za účelem nalezení hranic objektů. Lokální energie se propaguje k sousedům ve směru toku, pokud úhel mezi směry toků je menší než 90 ° [6].

Metoda podobně jako předešlé přístupy potřebuje explicitně určit uživatelem měřítko rozlišení (scale parametr σ, viz převzatý obrázek 3.1). Hrany budou detekovány, pokud budou větší než zadaný parametr σ. Obdobně v případě texturových příznaků pouze vzory textury menší než σ budou považovány za texely. Základní verze EdgeFlow byla následně rozšířena do čisté multiscale podoby, ve které není potřeba specifikovat úroveň rozlišení. Tato nová metoda upřednostňuje hrany, které se vyskytují ve více různých rozlišení. Lokalizace hran z důvodu větší přesnosti je prováděna v jemnějším měřítku rozlišení. Multiscale přístup dokáže zvýšit kvalitu detekce hran na obecných obrázcích [11].



(a) Rozlišování mezi jednotlivými objekty. Obrázek analyzován se scale parametrem mezi hodnotami 8 až 9.

(b) Rozlišování mezi shluky (textury) objektů. Obrázek analyzován se scale parametrem mezi hodnotami 38 až 42.

Obrázek 3.1: Explicitní volba parametru rozlišení.

3.2 Structured Forests for Fast Edge Detection

Spousta důležitých hran neodpovídá gradientu intenzity obrazu, ale je závislých na mnoha dalších obrazových vlastností a situací. Trénovatelné detektory hran mohou úspěšně řešit tento problém. Trénovatelné detektory jsou vhodné v případech, kdy je dána specifická oblast problému, na kterou se mohou velmi dobře naučit. Některé trénovatelné metody dokážou zobecnit naučený hranový model napříč datasety a tím mohou také zastat práci

obecných hranových detektorů. Obvykle se používá více nezávislých prediktorů, jejichž výsledné hodnoty jsou zkombinovány. Jedna z nejnovějších metod je založena na strukturovaném učení aplikované na rozhodovací les.

Původní idea byla vzít záplatu obrazu (patch) a určit zda se jedná o pozitivní vzor, jehož středový pixel obsahuje hranu, nebo nikoliv v negativním případě. V podstatě by se jednalo o binární klasifikátor. Problémem je ale velká rozmanitost hran a trénování by v tomto případě bylo velmi obtížné. V případě binárního klasifikátoru se totiž ignoruje lokální struktura hrany. Pokročilejší přístup je založen na tzv. sketch tokenech a spočívá v zařazení obrazových záplat do skupin na základě struktury hrany. Různorodost struktury hran v těchto skupinách je menší a vhodnější pro učení. Sketch tokeny berou v úvahu ground thruth obrazové záplaty a klastrují je do skupin – paralelních čar, T-spoje apod., několik převzatých příkladů tokenů je zobrazeno na obrázku 3.2. Metoda tedy z obrazové záplaty přímo predikuje skech token. [5]

Obrázek 3.2: Příklady naučených sketch tokenů.

Rozhodovací les (random forest) pro metodu sketch token je složen z n nezávislých rozhodovacích stromů. Při vyhodnocování se v každém uzlu provede test s porovnáním obrazového příznaku na práh a na základě porovnání se pokračuje do levého nebo pravého podstromu rekurzivně až k listovému uzlu. V listových uzlech je typicky pro klasifikaci do více tříd distribuované rozložení příslušnosti vstupních dat do těchto tříd. V případě sketch tokenu příslušnost obrazového vzoru do sketch tokenu. Vyhodnocení se provádí nezávisle nad všemi stromy. Zkombinováním těchto dílčích výsledků nalezneme nejpravděpodobnější token.

Myšlenka, která stojí za strukturovaným lesem, byla v postupném rozšiřování binárního klasifikátoru na klasifikátor do více tříd (sketch token) a nyní Structured Forests dále rozšiřují klasifikaci o možnost předvídat z obrazové záplaty lokální strukturu hran této záplaty. Strukturovaný les je tedy náhodný les, který má v listových uzlech uloženu „strukturní informaci“.

Klasický přístup trénuje jeden uzel tak, aby rozdělení uzlu dávalo maximální informační zisk, tedy aby byla minimalizována entropie. V případě strukturovaných výstupů je snaha, aby v jednotlivých podčástech byly hrany, které jsou si co nejvíce podobné. Toho lze docílit před provedením rozdělovací funkce klastrováním strukturovaných výstupů do tříd. Poté lze používat standardní náhodný les.

Z pohledu učení je důležité, aby rozhodovací stromy byly na sobě vzájemně nezávislé. V opačném případě hrozí přetrénování. Nezávislosti rozhodovacích stromů lze dosáhnout náhodným podvzorkováním trénovacích dat, nebo zavedením náhodného podvzorkování příznaků v rozhodovacích funkcích. Stromy jsou trénovány odděleně s cílem nalezení parametrů rozdělovací funkce, která povede k maximalizaci zisku informačního kriteria. Trénování se provádí rekurzivně pro oba podstromy, dokud nebude dosaženo požadované přesnosti, nebo maximální hloubky stromu.

Při zkoumání vstupní záplaty získáváme při použití jediného stromu celou výstupní predikovanou záplatu, tedy strukturní informaci hran, což je velmi výhodné. Daná skutečnost způsobuje, že při velikostech záplat například 16×16 jeden zkoumaný pixel přináší až 256 predikovaných hodnot do výstupního okolí. Výstupní hodnoty se překryjí přes sebe

a zprůměrují. Proto metoda nevyžaduje veliké množství stromů, jejichž výsledky by se kombinovaly dohromady. Z tohoto důvodu se při použití malých záplat a tedy malého množství překrývajících vzorků objevuje ve výstupním obraze šum typu pepř a sůl. Okolí strukturního výstupu obsahuje korelaci hodnot, a proto výsledné hrany jsou spojité a vyhlazené.

Díky používáním méně stromů se metoda vyznačuje až překvapivě vysokou výpočetní efektivností. Což je výhoda hlavně při použití detekce hran s cílem předzpracování obrazu pro další metody počítačového vidění [3].

3.3 Vyhodnocovací metriky

Abychom mohli porovnat úspěšnosti metod detekce hran, musíme použít vyhodnocovací metriku. Jedna z používaných metrik společně s veřejným datasetem nese název *The Berkeley Segmentation Dataset and Benchmark*. Výhoda spočívá v porovnávání vlastních výsledků s ostatními implementacemi různých metod, které jsou veřejně přístupné. Vyhodnocovací benchmark úspěšnosti metod je zahrnut k Matlab implementací detektoru Structured Forest [8].

Pro porovnání úspěšností metod se používá precision – recall křivka. Nad spojitými výstupy detektorů hran se provádí několikanásobné prahování. Nad binárním výstupem se vypočtou parametry precision a recall (rovnice 3.3 a 3.4), které jsou zakresleny do výsledného grafu.

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{3.3}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{3.4}$$

Jinými slovy parametr *precision* udává poměr správně označených hran k počtu všech hran, které byly algoritmem označeny jako možné hrany. Parametr *recall* udává poměr správně označených hran k počtu všech hran označenými lidmi (tedy počtu všech správných hran).

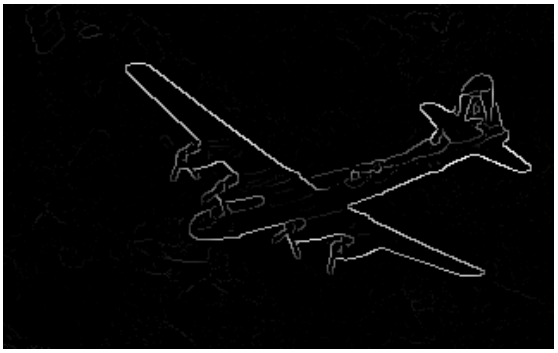
Obrázek 3.3 zobrazuje příklady vizualizace výsledků při provedení prahování s různými hodnotami prahu.

- Zelená barva – hrany nalezené detektorem, u kterých došlo ke shodě s hranami označenými lidmi (*true positive*).
- Modrá barva – hrany nalezené detektorem, které neodpovídají žádným hranám označenými lidmi (*false positive*).
- Červená barva – hrany, které detektor nenašel (*false negative*).

Výsledná anotace letadla byla složena z pěti dílčích anotací. Tyto anotace lze nejlépe vidět na obrázku 3.3d (červená barva), kde hrany s větší intenzitou červené barvy byly označeny více lidmi. Všechny hranice anotované lidmi jsou považovány za platné, což lze vidět na obrázku 3.3b (tenká hranice vedoucí okolo letadla).

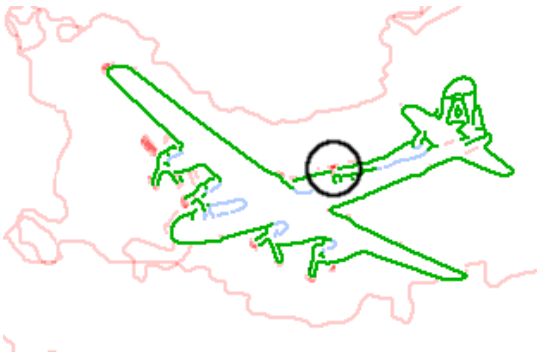
Anotované a porovnávané hrany nemusí vždy ležet na přesně stejné pozici – při porovnání obrázků 3.3c a 3.3d ve vyznačeném místě si lze všimnout, že zelená hrana nalezená Derichovým detektorem se nachází těsně pod červenou anotovanou hranou (červený schodek v obrázku 3.3c).

Pro snadnější porovnávání úspěšností křivek precision - recall se používá veličina F, která je definována vztahem 3.5. Hodnota F je tedy definována pro všechny úrovně prahu. Pro

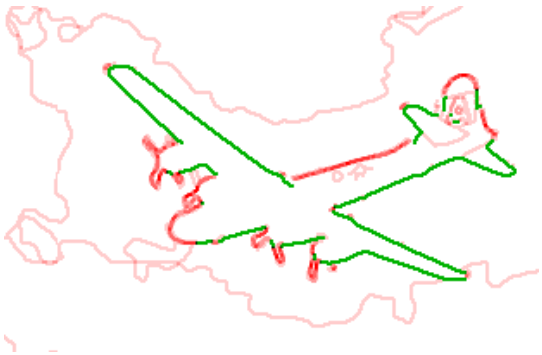


(a) Výstup z Derichova hranového detektoru parametr $\alpha = 1,0$.

(b) Vizualizace vyhodnocení obrázku s nastaveným prahem na 1 % ($precision = 0,22; recall = 0,93$).



(c) Vizualizace vyhodnocení obrázku s nastaveným prahem na 20 % ($precision = 0,89; recall = 0,68$).



(d) Vizualizace vyhodnocení obrázku s nastaveným prahem na 60 % ($precision = 1,0; recall = 0,42$).

Obrázek 3.3: Příklady vizualizace vyhodnocovací metriky na výstupním obrázku Derichova hranového datektoru ($\alpha = 1,0$) při zvolené intenzitě prahů: 1, 20 a 60 %.

porovnávání algoritmů se bere pouze maximální hodnota této veličiny.

$$F = \frac{2 * precision * recall}{precision + recall} \tag{3.5}$$

3.4 Berkeley Segmentation Dataset

Tento dataset vytvořila univerzita Berkeley pro svůj výzkum obecných detektorů hran a segmentačních algoritmů. Dataset slouží k vyhodnocování a porovnávání úspěšností jednotlivých metod.

Každý člověk má odlišné vnímání světa. Někdo může považovat hranu za významnou, jiný nikoliv. Aby byla pokryta tato divergence, dataset se skládá z obrázků, které byly anotovány prostřednictvím až třiceti lidí [7]. Některé obrázky byly navíc zpracovány více osobami. Z těchto důvodů nemá smysl vytvářet vlastní obecný dataset.

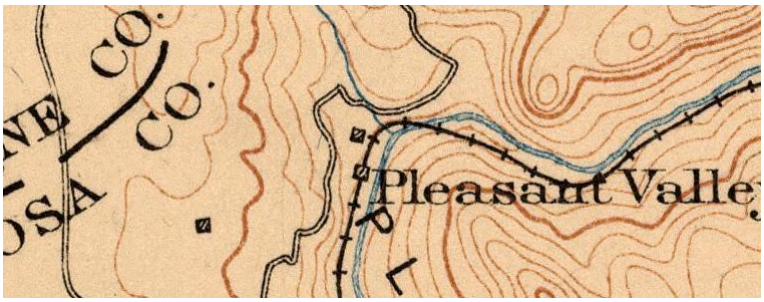
Součástí práce bude implementování anotačního nástroje pro anotaci speciálních obrázků (mapy a CT snímky). Anotační nástroj bude vytvořen s ohledem na existující formát dat, který zavedla univerzita Berkeley. V programu bude tedy možné si zobrazit a editovat anotované obrázky tohoto datasetu. Detaily anotačního nástroje a jeho formátu jsou uvedeny v sekci 5.1.

Kapitola 4

Návrh experimentů s mapovými podklady

Vyhodnocování a porovnávání úspěšnosti metod je velmi důležité. K tomu je ale zapotřebí mít data reprezentující skutečné pozice hranic (tzv. ground truth hrany), které byly nejlépe označeny přímo člověkem. K vyhodnocování obecných metod lze používat dataset univerzity Berkeley. Jeho vlastnosti a výhody jsou uvedený v sekci 3.4. Pro vytváření speciálního anotovaného datasetu je potřeba implementovat grafický anotační nástroj. Nástroj exportuje anotovaná data do textového souboru v souladu s formátem datasetu Berkeley. Následující část představuje možnou podobu různých stylů anotace vlastního datasetu.

Mapy uvedené v této práci, na kterých byly prováděny experimenty patří U.S. Geological Survey z roku 1912 Yosemitského národního parku v americkém státě Kalifornie [1]. Výřez této historické mapy je ukázán na obrázku 4.1. Na mapové podklady byl aplikován algoritmus *Structured Forests for Fast Edge Detection* s cílem prozkoumat možnosti extrakce významných mapových elementů.



Obrázek 4.1: Výřez zpracovávané mapy.

Cílem je zvýraznění pouze požadovaných hran například pro další případnou vektorizaci a potlačení všech ostatních hran (jako jsou vrstevnice, veškeré názvy apod.). Práce se zaměřuje na zjišťování charakteristických elementů, u kterých existuje teoretická možnost „dobrého“ natrénování detektoru. Zobrazený mapový výřez obsahuje tyto významné prvky, které budou extrahovány:

- Silnice – dle struktury dvojice paralelních čar se přímo vybízí k testování.
- Železnice – linie s výstupky.

[1] David Rumsey Historical Map Collection dostupné na `http://www.davidrumsey.com/`

- Vodní toky – tenké linie potoků včetně širších vodních toků a ploch.

Experimenty budou mimo jiné zaměřeny také na stanovení parametrů detektoru a určení jejich vlivu na úspěšnost detekce v závislosti na specifickém datasetu, jakým jsou mapové podklady. Postupnými změnami parametrů budou sledovány trendy ve změnách dosažených úspěšností (křivky precision - recall) a na základě dosažených výsledků budou stanoveny závěry a doporučení. Mezi zkoumané parametry patří: minimální velikost datasetu, počet stromů v lesu, rozměry záplat (výsledky jsou uvedeny v sekci 6.2).

Poslední část experimentů je zaměřena na porovnání klasických detektorů a detektoru Structured forest na speciálních datasetech. Experimenty zjišťovaly, zda úpravou parametrů klasického detektoru je možné se přiblížit k výstupu trénovatelného algoritmu.

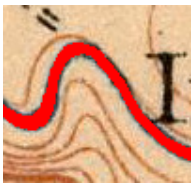
4.1 Možnosti anotování mapových elementů

Při anotování mapových elementů existuje více stylů anotace. Na vybraný neměnný dataset při definování stejných parametrů bude detektor natrénován.

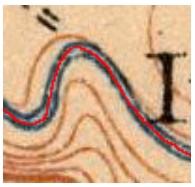
Řeky

Podobně jako v případě cest a železnic lze zvolit různé anotace vodních toků. Vodní toky se v mapách vyskytují ve formě od tmavě modrých tenkých čar ($\sim$ 1 pixel) až k širším řekám tloušťky několika pixelů, jak je vidět na obrázku 4.2a. Anotace v případě tenkých linií řek je zřejmá. Pro anotování širokých toků budou testovány dva styly anotací. První styl 4.2b označuje všechny modré pixely náležící k řece. Druhý styl 4.2c vede středem všech řek tenkou čáru o tloušťce jednoho pixelu.

(a) Mapový výřez určený k anotaci řek.



(b) Styl 1



(c) Styl 2

Obrázek 4.2: Anotace vodních toků.

Cesty

V tabulce 4.1 jsou ukázány možné způsoby - styly anotace. Příklady se vztahují k mapovému výřezu cesty 4.3, kde šířka značené silnice odpovídá přibližně 9 pixelům.

Obrázek 4.3: Mapový výřez určený k anotaci cest.

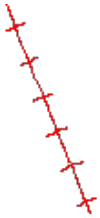
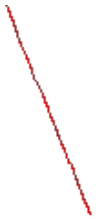
1		Dvě tenké paralelní jedno pixelové linie vedené přes střed černých čar.
2		Vyplněný světlý prostor mezi dvěma černými čarami. Jedná se o inverzi prvního způsobu anotace.
3		Sjednocení předešlých dvou stylů.
4		Tenká jedno pixelová linie vedená středem cesty.

Tabulka 4.1: Možnosti anotace cest.

Železnice

Různé styly anotace železnic popisující mapový výřez 4.4 jsou uvedeny v tabulce 4.2. Doposud všechny se všemi mapovými elementy bylo zacházeno jako se spojitými křivkami. Pro odlišení byl třetí styl popisu železnice zvolen jako osamocené průsečíky křížení úseček na železniční trati.

Obrázek 4.4: Mapový výřez určený k anotaci železnice.

1			Kopírování tvaru celé křivky včetně kolmých výstupků.
2			Pouze tenká jedno pixelová linie vedená středem hlavní čáry.
3			Označená pouze místa křížení.

Tabulka 4.2: Možnosti anotace železnic.

4.2 Získání vektorové reprezentace hran

Dalším stupněm zpracování obrazu po předchozí detekci hran může být převedení získaných hran z rastrové reprezentace do vektorové. Postup vektorizace získaných hran z map může být následovný:

U správně natrénovaného detektoru můžeme předpokládat, že maximální odezvy na hrany v obrázku patří opravdu požadovaným hranám (cestám). Tato pozice maxima je brána jako výchozí bod cesty, od které se začíná prohledávat blízké okolí. V okolí se nalezne další druhý bod, který se přidá do posloupnosti bodů tvořící cestu. Bod je vybrán na základě ohodnocení (skóre) podle zvolené funkce. Aby se zabránilo opakovanému průchodu již zpracovaných částí, lze průběžně mazat rastrovou plochu mezi dvěma body tvořícími cestu v šířce předpokládané rastrové tloušťky cest. Vynulování intenzity hran způsobí, že skóre bodů s nulovou intenzitou budou malé. Funkce počítající skóre může být například jednoduchá funkce intenzity a vzdálenosti, kdy bližší body s vyšší intenzitou budou preferovány.

Popsaný způsob by našel pouze jednu možnou cestu. Cesty se ale reálně vždy kříží, někdy končí jako slepé ulice, nebo jsou useknuty okraji obrazu. Aby se algoritmus znovu slepě nespouštěl z následujícího lokálního maxima a aby byla zachována topologie cest, je použito zpětné sledování cesty (backtracking). Od koncového bodu křivky se postupně navracíme k předchozím bodům a zkoumáme jejich okolí stejným způsobem jako v prvním případě. Při nalezení okolních vyhovujících bodů dojde k rozdvojení původní cesty. Místo zpětného sledování cesty je také možné při vložení nového bodu prohledat jeho okolí pro více možných cest.

Kapitola 5

Implementace

Vzhledem k požadavkům na vytvoření datasetu anotovaných obrázků, byl implementován anotační nástroj. Dále byly implementovány některé z metod uvedených v předchozích kapitolách. Implementace zkoumaného trénovatelného detektoru Structured forest for fast edge detection včetně vyhodnocovacích prostředků v jazyce Matlab byla získána z Microsoft Research [8]. Pro použití je potřeba prostředí Matlab (testováno na verzi R2013a) s nainstalovaným Image Processing Toolboxem a s dalším doplňkovým toolboxem Piotr's Computer Vision Matlab Toolbox, který je dostupný na webových stránkách [2]. V této části budou popsány nutné kroky k použití této implementace trénovatelného detektoru s ostatními programy, jakým je anotační nástroj. Jako rozšiřující dodatek byla implementována základní podoba nástroje, která převádí rastrové hrany do vektorové reprezentace.

5.1 Anotační nástroj

Nástroj umožňuje kreslení oblastí (segmentů) přes původní obrázek. Jednotlivé oblasti (třídy) jsou barevně odlišeny. Hrany se tedy nacházejí mezi hranicemi jednotlivých oblastí. Protože program umožňuje rozlišovat jednotlivé oblasti, mohou být anotovaná data využívána kromě detekce hran i pro segmentaci obrazu.

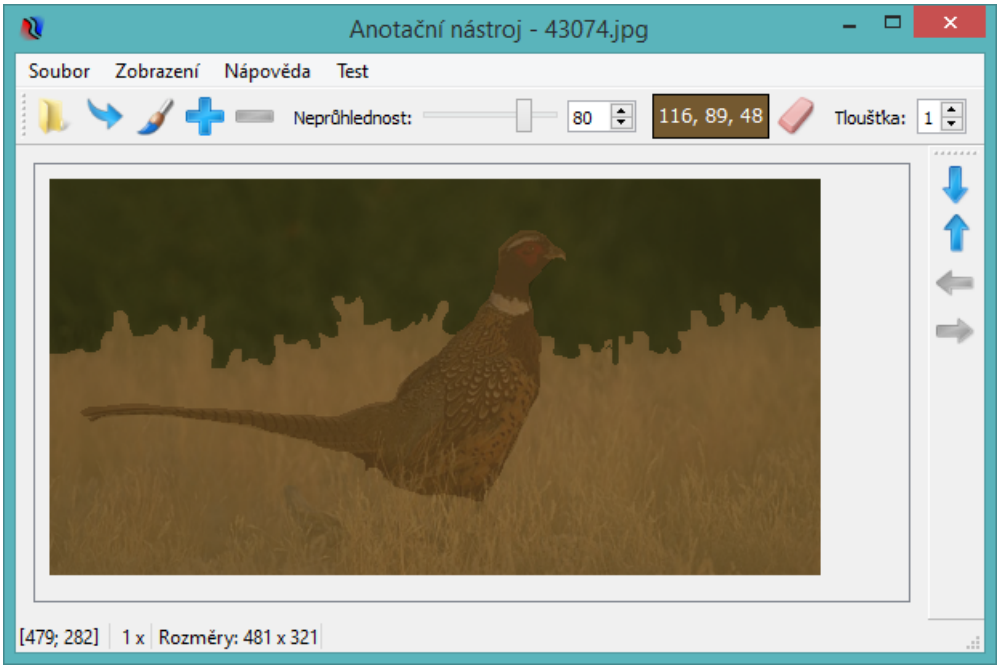
Hlavní okno aplikace s příkladem již hotové anotace je zachyceno na obrázku 5.1. Zobrazený příklad se skládá ze tří segmentů a byl načtený z anotovaného datasetu *The Berkeley Segmentation Dataset* (číslo obrázku: 43074, anotováno uživatelem číslo: 1117).

Program byl vytvořený v jazyce C++ s použitím grafického frameworku Qt (verze 5 a novější). Většina z použitých ikon v programu pochází z balíčku ikon Visual Studio Image Library.

Formát anotovaných dat

Následuje formát hlavičky textového souboru. Jediný podporovaný typ formátu je *ascii cr*, který je zde dále uveden. Položky *width, height* a *segments* jsou povinné (šířka, výška v pixelech a počet všech segmentů-oblastí). Další nepovinné položky jsou například datum, identifikátor obrázku, uživatele a příznak šedotónového obrázku (*date* <*string*>, *image* <*int*>, *user* <*int*>, *gray* $\{0 \mid 1\}$). Následuje nejmenší možný tvar hlavičky souboru.

```
format ascii cr
width <int>
height <int>
```



Obrázek 5.1: Anotační nástroj.

```
segments <int>
data
...
```

Hlavička souboru je ukončena klíčovým slovem *data*, za kterým následuje popis segmentů v obrázku. Každý řádek v souboru obsahuje čtyři celé čísla odděleny mezerami:

```
<number> <row> <from> <to>
```

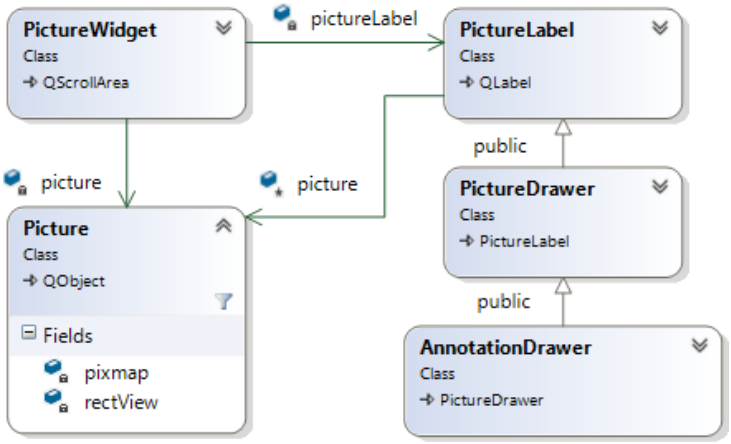
Význam je postupně následující: číslo segmentu (třídy), řádek obrázku a interval sloupců v pixelech, které náleží danému segmentu. Číslování všech hodnot začíná od nuly.

Implementace

Standardní přístup k zobrazení obrázku v prostředí Qt spočívá ve vykreslení pixmapy na pozadí widgetu (např. QLabel umístěného ve skrolovací oblasti). Jakékoliv provedené modifikace anotované vrstvy vyžaduje překreslení celého obrázku (originálního obrázku, i jeho anotované vrstvy). V případě obrázků s velikým rozlišením dochází ke znatelnému zhoršení výkonu. Stejný problém nastává i při zvetšení malých obrázků.

Aby program umožňoval anotaci obrázků s velkým rozlišením a zvětšením (testováno s rozlišením obrázku 9000×9000 pixelů), byl vytvořen univerzální *PictureWidget*, který zobrazuje a překresluje vždy pouze viditelný výřez obrázku. Diagram popisovaných tříd je zobrazený na obrázku 5.2. PictureWidget je potomkem QScrollArea a zapouzdřuje základní ovládání pohybu a zvětšování obrázku. PictureWidget obsahuje dva objekty tříd *PictureLabel* a *Picture*. PictureLabel slouží především pro prosté vykreslení obrázku a jako bázová třída pro odvození PictureDrawer. Třída Picture obaluje objekt QPixmap, ze kterého zpřístupňuje pouze obdélníkový viditelný výřez. Třída Picture obsahuje metody umožňující

pohyb výřezu po celém obrázku, včetně zvětšení obrázku, kterého je dosaženo zmenšením ořezového obdélníku.



Obrázek 5.2: Diagram tříd widgetu anotačního nástroje pro zobrazování a editaci rozměrných obrázků.

Pro pohodlnější segmentaci obrázku je možné upravovat průhlednost barevné vrstvy. Na uvedeném příkladě lze vidět prosvítání originálního obrázku s 80% průhledností anotované vrstvy. Aktuální barva, průhlednost barevné vrstvy, tloušťka obrysové hrany a další funkce lze ovládat pomocí prvků v nástrojové liště. Po každém výběru nové barvy proběhne kontrola přítomnosti barvy v anotaci. Pokud již barva byla použitá, uživateli se objeví upozornění. Označením tlačítka *Mazání* lze místo „kreslení“ nanesenou barvu „odstraňovat“. Obdobně lze třídu barvy odstranit z anotace prostřednictvím položky *Aktivní barva do průhlednosti* v menu *Zobrazení*.

Aplikace předpokládá, že textový soubor anotace se jmenuje stejně jako anotovaný obrázek (koncovka .seg) a nachází se ve stejném adresáři. Po načtení jsou jednotlivým třídám přiřazeny náhodné barvy. Aplikace také umožňuje barevnou vrstvu (obrázek odpovídající anotaci) exportovat, nebo importovat (například pro zachování požadovaných barev).

Jednotlivé segmenty lze jednoznačně obarvit podle průměrné barvy pixelů anotovaného obrázku náležících k těmto segmentů. Přitom různým segmentům jsou vždy přiřazeny různé barvy. Tato možnost se v aplikaci jmenuje *Obarvit dle originálu*. Podrobnější ovládání anotačního nástroje je popsáno v příloze A.

5.2 Hranové detektory

V rámci semestrálního projektu vznikla demonstrační implementace konvenčních hranových detektorů uvedených v kapitole 2. Algoritmy byly implementovány v jazyce C++ s využitím knihovny OpenCV. Výstupy některých implementovaných algoritmů včetně jejich mezivýsledků jsou uvedeny v téže kapitole. Konzolový program se jmenuje EdgeDetections, přijímá tři parametry:

```
EdgeDetection [vstupní_obrázek|adresář_obrázků] výstupní_adresář param.txt
```

Program pracuje ve dvou režimech:

- Pokud je na vstupu zadán jediný obrázek, výstupní adresář bude kromě výsledku detekce hran obsahovat také jednotlivé mezivýsledky zadaných algoritmů (například

obrázky před a po potlačení nemaximálních hodnot, po dvojitém prahování, ...). Běh algoritmů nad jediným obrázkem řídí objekt třídy `FileRun`.

- V případě, že na vstupu byl zadaný adresář, bude nad každým obrázkem v adresáři provedena požadovaná metoda detekce hran a do výstupní složky bude uložen pouze výsledek detekce hran. Běh algoritmů nad všemi obrázky zadaného adresáře řídí objekt třídy `DirRun`.

Definice metod a jejich parametrů, které budou provedeny nad vstupy, je určena v textovém souboru *param.txt*. Každá metoda je zadána na novém řádku. Program obsahuje implementaci metod: Marr - Hildreth, Cannyho, Derichův hranový detektor, Anizotropická difuze (mh, c, d, ad). V závorce jsou uvedeny zkratky metod, které se používají v souboru s parametry. Tyto zkratky jsou následovány parametry příslušných metod oddělené mezerami (žádný z parametrů nemůže být vynechán):

```
mh kernel_size sigma threshold
c kernel_size sigma low_threshold high_threshold
d alpha low_threshold high_threshold
ad K number_iteration lambda low_threshold high_threshold
```

Hodnoty parametrů prahů jsou uvažovány v rozsahu 0 až 1. Na základě názvu použité metody a jejich parametrů vznikne ve výstupním adresáři jednoznačně pojmenovaný podadresář s výslednými obrázky.

5.3 Použití Structure forest for fast edge detection

Zprovoznění trénovatelného detektoru, nad vlastním datasetem vyžaduje několik kroků, které jsou zde popsány. Dataset obrázků a jejich anotace se rozdělí do následujících složek.

```
Dataset
   images
      train
      test
   groundTruth
      train
      test
```

Složka *images* obsahuje pouze obrazová data, složka *groundTruth* obsahuje pouze soubory anotovaných dat v binárním formátu Matlabu (soubory s koncovkou **.mat*). Přičemž názvy souborů v obou složkách train i v obou složkách test se musejí až na přípony shodovat.

Vytvoření souborů groundTruth

Aby anotovaná data vytvořená v anotačním nástroji mohly být použita v Matlab implementaci detektoru, musí být převedena do odpovídajícího binárního formátu. Pro tento účel byla vytvořena funkce `seg2matDir(segDir, matDir)`, která převede všechny soubory anotace zadané v adresáři segDir (formát se kterým pracuje anotační nástroj) na odpovídající binární soubory do adresáře matDir. Struktura souborů ve vstupním adresáři může být dvojího typu:

- Samostatné seg soubory – každý soubor odpovídá jedné anotaci jednoho obrázku.

- Vnořené adresáře obsahující skupinu seg souborů – adresář představuje jeden obrázek, který byl anotován vícekrát. Všechny vnořené anotace budou obsaženy v jediném mat souboru, jehož jméno se shoduje s názvem adresáře.

Trénování detektoru

Před samotným trénováním je potřeba nastavit parametry modelu. Výchozí parametry lze získat a upravovat následujícími příkazy:

```
opts = edgesTrain();                % Získání výchozích parametrů detektoru
opts.nPos = 1e5; opts.nNeg = 5e5; % Počet pozitivních a negativních záplat
opts.nTrees = 6; ...                % Počet stromů
```

Příklad 1: Nastavení parametrů trénování v Matlabu.

Seznam všech možných parametrů je obsažen v souboru *edgesTrain*. Třemi vyžadovanými parametry jsou: název adresáře datasetu, adresář a název výsledného modelu. Samotné trénování v závislosti na nastavených parametrech může trvat dlouhou dobu.

```
opts.bsdsDir = 'dataset';
opts.modelDir = 'model_dir';
opts.modelFnm = 'modelName';
model = edgesTrain(opts);       % Zahájení trénování
```

Příklad 2: Vyžadované parametry jmen souborů a složek s daty v Matlabu.

Vyhodnocování úspěšnosti

Natrénovaný model lze vyhodnotit pomocí funkce `edgesEval(model, 'show', 1);`. Tato funkce aplikuje algoritmus Structure forest nad všemi obrázky ze složky *test* a provede vyhodnocení modelu.

Pokud chceme vyhodnotit úspěšnost vlastních hranových detektorů, například těch které jsou uvedeny v předchozí sekci 5.2, můžeme vyhodnocení provést dle příkladu 3.

```
arg={'resDir', 'custom_detector/result', 'gtDir', 'custom_detector/gT'};
edgesEvalDir(arg);
edgesEvalPlot(arg(2), [{'Name'}], [{'b'}]);   % precision-recall graf
```

Příklad 3: Vyhodnocení úspěšnosti detekce hran v Matlabu.

Adresář *result* obsahuje výstupy hranového detektoru (obrázky png formátu). Intenzita bílé barvy odpovídá síle hrany. V adresáři gT jsou umístěny všechny anotace testovaných obrázků (binární soubory Matlabu). Vyhodnocování dat je opět časově náročná operace.

5.4 Propojovací nástroj

K převodu rastrového výstupu hran znázorňující silniční sítě byl vytvořen nástroj pro příkazovou řádku. Program byl vytvořen v jazyce C++ s využitím knihovny OpenCV. Základní

použití programu:

```
ConnectionTool vstup.png výstup.png šířka_linie ...
```

Vstupním obrázkem nástroje je výstup detektoru hran. Ve výstupním obrázku jsou zakresleny body tvořící nalezené křivky. Šířka linie odpovídá přibližné tloušťce cesty v pixelech – tloušťka odezvy detektoru hran na požadovanou cestu. Další volitelné parametry se předávají třídám, které počítají ohodnocení bodů.

Hlavní třída `Connector` řídí základní běh algoritmu popsaný v části 4.2. Přidává nové body do výsledné křivky, provádí backtracking apod. Výběr nejvhodnějších bodů závisí na abstraktní třídě `Finder`. Potomek musí implementovat abstraktní metodu `findNextPoint`, která vrací bod s nejvyšším skóre vzhledem k bodu zadanému ve svém parametru. Tento vrácený bod bude tvořit výslednou křivku. Popsaným způsobem lze vytvářet složitější strategie výběru bodů, které berou v úvahu další vlastnosti obrazu.

Funkce určující další bod posloupnosti může být libovolně složitá. Potomci třídy Finder mají k dispozici původní vstupní obraz a také kopii původního obrazu, u které se v průběhu algoritmu postupně vymazávají okolní intenzity zpracovaných hran. Ukázková implementace preferuje bližší body s vyšší intenzitou (třída `IntensityDistance`).

Program generuje výstupní obrázek se zakreslenou nalezenou cestou. Reprezentace symbolů je následovná: Křížek značí počátek prohledávání. Propojené body jedné barvy náleží jedné souvislé cestě. Propojení více cest na základě backtrackingu je v obrázku značeno bílými kružnicemi. Velká kružnice obsahuje bod, ze kterého došlo k rozdvojení cesty. Malá kružnice představuje první bod navázané cesty. Tyto větvení jsou očíslovány dle pořadí jejich vzniku.

Kapitola 6

Výsledky experimentů

V této části se nachází vyhodnocení a porovnání různých stylů anotace zavedených v sekci 4.1. Dále je zde diskutována volba parametrů trénování modelu a vliv těchto parametrů na výslednou kvalitu detekovaných hran. Tato sekce může být chápána jako pomoc při volbě parametrů modelu detektoru *Structured forest for fast edge detection*. Na konci kapitoly je porovnání trénovatelného detektoru s klasickým Derichovým detektorem na specifických datasetech.

6.1 Porovnání různých stylů anotace na mapových podkladech

Aby se zdůraznily rozdíly mezi různými styly anotací, byly pro trénování detektoru zvoleny stejné podmínky v rámci jedné skupiny. Trénováno bylo prováděno na stejném datasetu se stejným nastavením parametrů detektoru.

Cesty

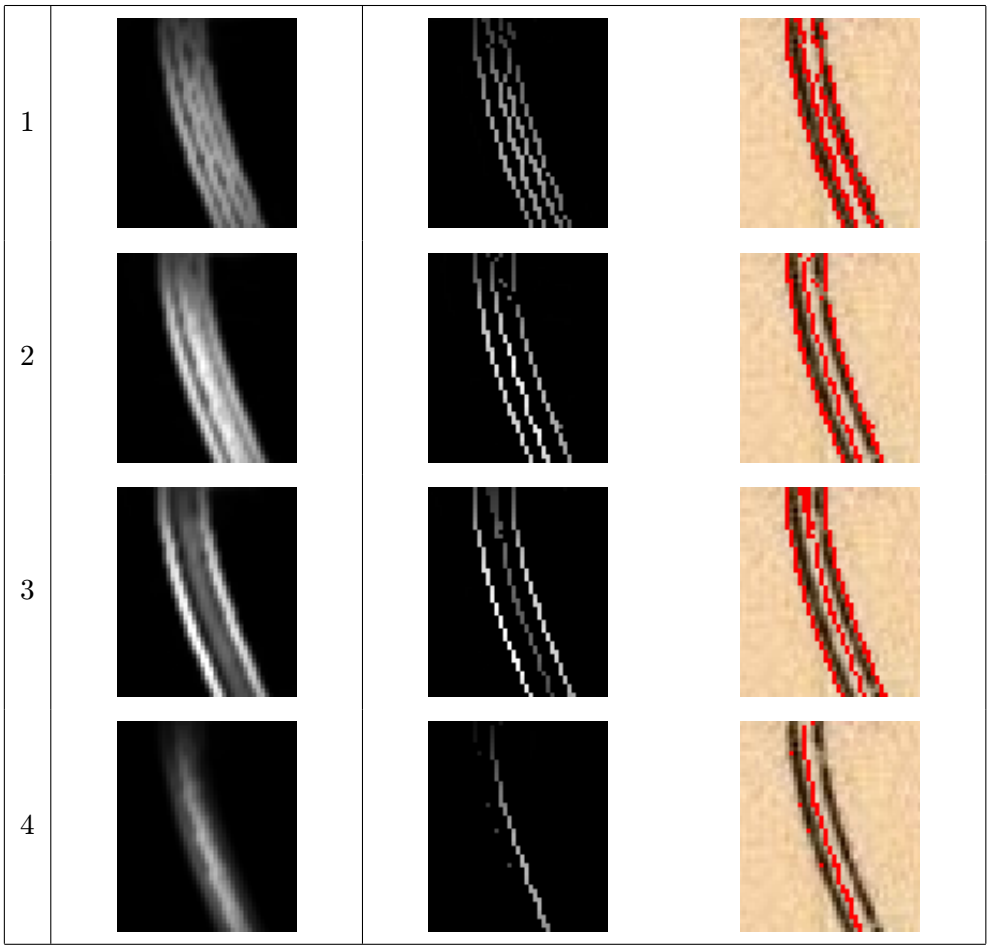
Tabulka 6.1 předvádí rozdíly mezi různými styly anotování cest. Výřez části cesty byl zvolený tak, aby dobře ilustroval rozdíly mezi použitím různých stylů. Čísla uvedených možností odpovídají číslování zavedené v sekci 4.1. Aby byly dobře vidět rozdíly, všechny uvedené výřezy obrázků jsou zvětšeny. První sloupec v tabulce je výstup z trénovatelného detektoru. Na prostřední obrázky bylo aplikováno potlačení nemaximálních hodnot (non-maximum suppression 2.3). Hodnoty pixelů v těchto obrázcích jsou váženy dle síly hrany. Poslední sloupec zobrazuje tenké obarvené hrany vykreslené přes zdrojový obrázek. Aby byly hrany v původním obrázku dobře viditelné, nad všemi pixely hran bylo provedeno prahování.

V prvním případě byly samostatné tmavé paralelní čáry označeny jako segmenty. Odezvy na hrany jsou skutečně viditelné po obou okrajích křivky (lépe viditelné na prostředním a pravém obrázku). Nicméně vzhledem k blízkosti těchto paralelních křivek, na některých místech se odezvy na vnitřní hrany sčítají – spojují a vytvářejí podobnou odezvu jako ve druhém řádku tabulky.

Druhý a třetí styl se od sebe lišily pouze v tloušťce segmentu. Druhý styl vyplňoval především oblasti vnitřní části cesty, třetí (tlustější) styl zahrnoval i část okolních čar. Výsledky se ale poměrně hodně liší. Obrysové hrany ve třetím řádku jsou znatelně dominantnější a také ve vnitřním prostoru mezi hranami je méně šumu než ve druhém případě.

Poslední část je nejodlišnější. Místo detekce krajních hran cest, měla být získána odezva

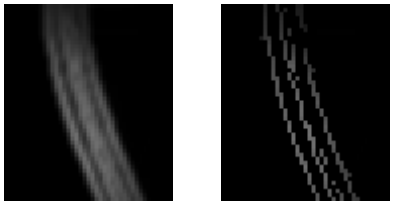
na střed cesty. Z obrázku se zdá, že tomu tak skutečně je, ale na rozsáhlejším datasetu se stávalo, že ne vždy nejsilnější odezva vedla středem cesty. V některých případech se výsledná hrana uchylovala k jedné či druhé straně silnice. Další nevýhodou bylo poměrně široké maximum. Z těchto důvodu se po provedení non-maximum suppression vyskytovaly kromě středové křivky také slabší postraní linie případně šum (jako je vidět několik pixelů na obrázku ve čtvrtém řádku vpravo).



Tabulka 6.1: Výsledky anotace cest.

Z výsledků experimentů je vidět, že používáním odlišných anotací cest lze výrazně ovlivnit výslednou odezvu trénovatelného detektoru. Třetí styl je nejvhodnější v případech, kdy chceme získat odezvu převážně kopírující původní linie silnic. Při převodu silnic na vektorovou reprezentaci se třetí styl jeví jako naopak nejhorší ze všech. Protože maxima jsou rozprostřena na okrajích silnic, převedené křivky mají tendenci přiléhat k jednomu z okrajů, nebo mezi nimi příčně přecházet. V případě vektorizace je nejlepší použít čtvrtý styl z důvodu zvýraznění odezvy na střed silnice.

Poslední pokus s anotacemi cest spočíval ve zkombinování všech stylů dohromady. Kdy se ke každému obrázku přiřadí více anotací – například od více uživatelů, kteří mají odlišný styl a přesnost anotování (obrázek 6.1). Výsledek se podobá zprůměrováním jednotlivých výsledků.



(a) Výstup z detektoru.

(b) Potlačení nemaximálních hodnot.

Obrázek 6.1: Výsledek při zkombinování všech stylů anotací cest.

Celkově detekce cest v mapách pracovala spolehlivě, jak lze vidět z následujících grafů v této kapitole. Nicméně v některých situacích detektor generoval slabé odezvy na nepravé hrany (false positive). Tato situace nejčastěji nastávala pokud se v mapě objevila plná černá čára o stejné tloušťce jako byla tloušťka silnice.

Řeky

V případě vodních toků výsledky obou zvolených anotací dopadly velmi dobře a velmi podobně. Na rozdíl od předchozích případů různá forma anotace vodních toků měla zanedbatelný vliv na výsledek. Jediný viditelnější rozdíl se objeví v některých oblastech řeky po provedení *nms* (obrázek 6.2), kdy hrany u druhého stylu plynule nekopírují hrany řeky ve vstupní mapě. Z tohoto důvodu jako lepší forma anotace vychází první možnost, kdy jsou široké řeky anotovány jako jeden objekt.

(a) Mapový výřez řeky.

(b) Výsledek po anotaci stylem 1.

(c) Výsledek po anotaci stylem 2.

Obrázek 6.2: Výsledky anotace vodních toků.

Železnice

První styl považuje veškerou tmavou oblast železnice za segment. Druhý styl naopak vede středem železnice tenkou jedno pixelovou čáru. Jak je vidět na obrázcích z tabulky 6.2, ve druhém případě je opravdu výsledná čára i přes náznaky výstupků přímější než při použitím prvního stylu. V případě křížků je odezva pouze na hranách v okolí křížení čar.

Nalezení železniční sítě vykazovalo nejnižší úspěšnost ze všech třech sledovaných útvarů.

1			
2			
3			

Tabulka 6.2: Výsledky anotace železnic.

6.2 Vliv parametrů na úspěšnost detekce

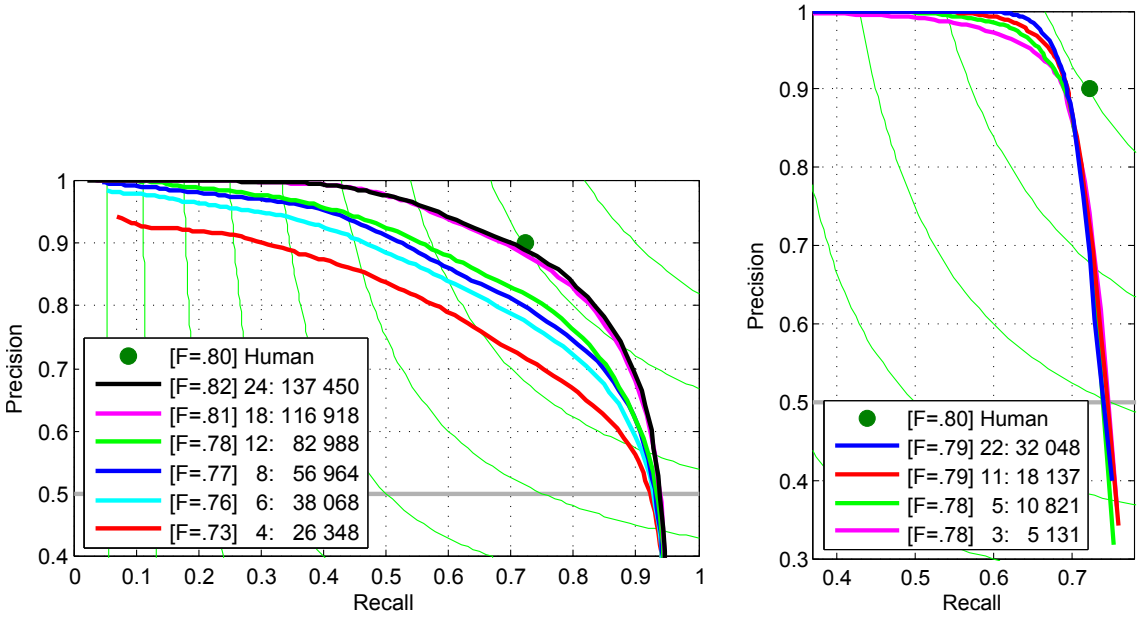
Následuje porovnání parametrů na úspěšnosti detekcí požadovaných hran. Důraz bude kladen na přibližné zjištění hranice, kdy nemá smysl dále zvyšovat parametry modelu. Každá z následujících podsekcí se věnuje jednomu parametru.

Počet trénovacích dat

V této části určíme minimální potřebnou velikost trénovacího datasetu. Neboli jaká je hranice, při které další zvětšování datasetu přináší pouze malé zvyšování úspěšnosti detekce hran. Experiment se vztahuje ke specificky zaměřeným úlohám jako jsou například uvedené mapové podklady.

Měření bylo prováděno na obrázcích o velikostech přibližně 500×500 pixelů. Všechny modely v jednom grafu byly spuštěny se stejnými parametry na stejném vyhodnocovacím datasetu. Mezi jednotlivými běhy byl pouze upravován počet obrázků, ze kterých byl detektor trénován. Výsledky jsou zakresleny v grafech 6.3a (detekce cest třetí styl) a 6.3b (detekce řek první styl). Čísla před dvojtečkou u výsledných křivek jsou počty obrázků použitých při trénování. Čísla za dvojtečkou odpovídají počtu pixelům, které byly použity pro anotaci útvaru.

V případě řek k dosažení velmi dobrých výsledků stačilo málo vzorových obrázků (10 obrázků $\sim 15\,000$ anotovaných pixelů), zvětšováním datasetu se pouze mírně zlepšoval parametr precision. V případě cest je zlepšení znatelnější. Jak je vidět z grafu k základnímu



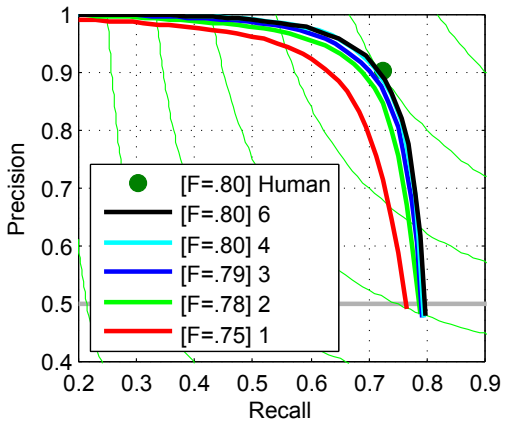
(a) Cesty (b) Řeky

Obrázek 6.3: Závislost velikosti trénovacího datasetu na úspěšnosti detekce hran.

natrénování detektoru postačuje přibližně 20 obrázků $\sim 120\,000$ anotovaných pixelů.

Počet stromů

Opět byl na stejný dataset aplikován detektor se stejnými inicializačními parametry, s výjimkou počtu stromů v modelu. Na grafu 6.4 jsou zobrazeny precision - recall křivky patřící detekci cest (použita kombinace všech stylů dohromady). Čísla u křivek odpovídají počtu stromů použitých při trénování.

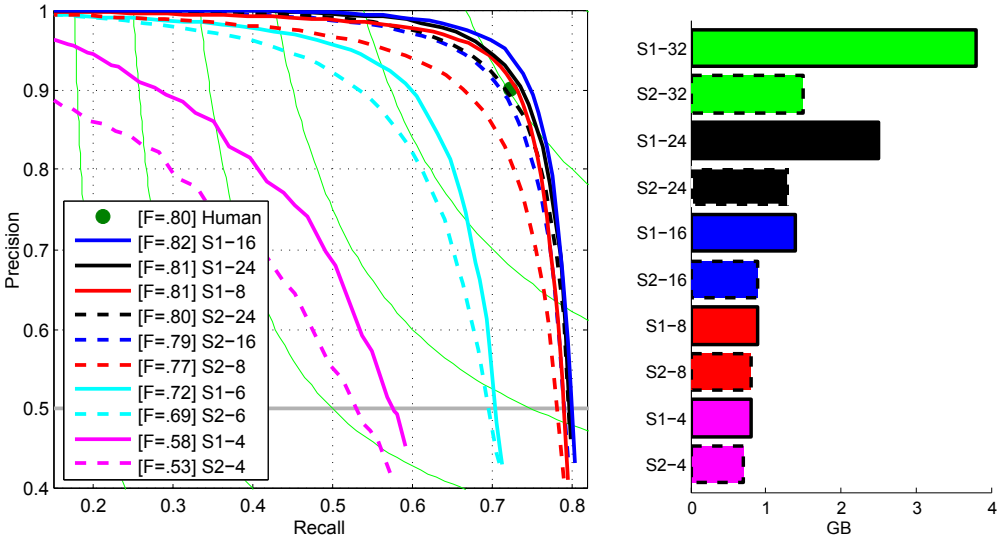


Obrázek 6.4: Závislost počtu stromů na úspěšnosti detekce hran.

Podle teorie využívání strukturní informace 3.2, by měla metoda dosahovat dobrých výsledků za použití menšího množství stromů, než je pro metody, které jsou založeny na random forest typické. Jak je vidět z grafu, za dostačující lze uvažovat počet pouze 4 až 6 stromů.

Rozměry záplat a podvzorkování obrazu

Graf 6.5a zobrazuje úspěšnost detekce cest v závislosti na rozměrech záplat a podvzorkování obrazu. Plnou čarou jsou zakresleny modely, u kterých nedošlo k převzorkování dat (značení *S1*). Čárkovaně jsou vyvedeny modely, u kterých bylo provedeno dvojnásobné zmenšení obrazu (značení *S2*). Záplaty o stejných rozměrech mají stejnou barvu (číslo za pomlčkou odpovídá velikosti záplaty v pixelech). Tyto dva grafy byly vytvořeny za použitím stejných modelů – barvy a styly čar si vzájemně odpovídají včetně použitého značení. Ve sloupcovém grafu je navíc pro zachycení trendu zobrazeno vytížení paměti při větší velikost záplat 32×32 pixelů. Dle očekávání použití příliš malých záplat vede ke zvýšené zrnitosti ve výsledných hranách. Větší rozměry vedou k vyhlazenějším průběhům cest.



(a) Úspěšnosti detekce cest.

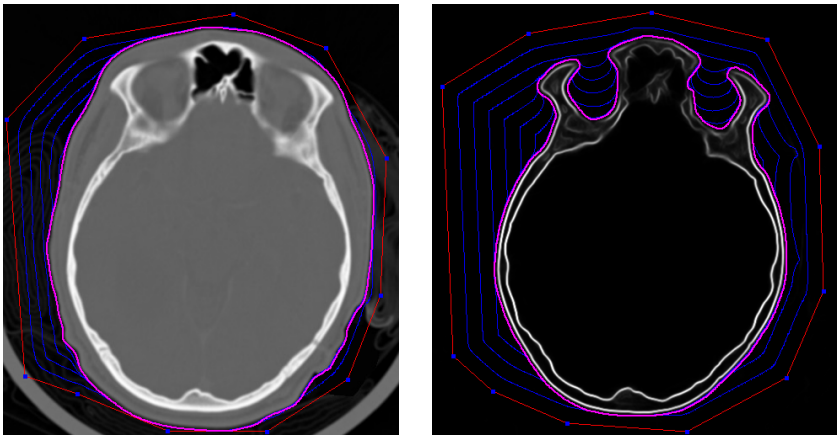
(b) Průměrná spotřeba paměti při trénování.

Obrázek 6.5: Závislost rozměrů záplat a podvzorkování obrazu.

Dle charakteru zpracovávaných dat (struktura cesty) si můžeme všimnout, že dobrých výsledků v případě nepodvzorkovaných obrázků lze dosáhnout už při malých velikostech záplat – pouze 8 pixelů. Z naměřených výsledků se jeví jako nejlepší možnost používání záplat o velikostech 16×16. Větší rozměry nepředstavovaly výrazné zlepšení. Jedním z důvodů proč se provádí podvzorkování je vidět na grafu 6.5b. Při původním rozlišení obrázků a větších rozměrech záplat rostou požadavky na paměť RAM daleko rychleji. Více paměti je potřeba pro trénování, ale i při samotné detekci hran.

6.3 Detekce hran v medicínských snímcích

Doplňkový experiment ukazuje možnosti použití detektorů hran v medicínské oblasti. Na *CT* (Computed Thomography) snímky hlavy byly aplikovány detektory hran s cílem předzpracování obrazových dat pro další metody – například aktivní kontury. Obrázek 6.6a ukazuje použití aktivních kontur. Červená křivka označuje počáteční (zadefinovanou) konturu, fialová reprezentuje výslednou nalezenou konturu. Modré vrstevnice znázorňují postupné rozpínání/zmenšování kontury.



(a) Aktivní kontura aplikovaná na CT snímek.

(b) Aktivní kontura aplikovaná na CT snímek po provedení detekce hran s důrazem na kostní tkáň.

Obrázek 6.6: CT snímky hlavy.

Kontura obtahuje hranici objektu, v tomto případě celé hlavy. Aktivní kontury pracují s hranami v obrazu, na které se snaží umístit křivku. Pokud nás zajímá určitá specifická část snímku například lebeční kosti, je možné před aplikací aktivních kontur snímek předzpracovat a zvýraznit tak pouze hrany odpovídající kostem.

Hranový detektor byl natrénován na rozhraní kostní tkáně. Nad obrázkem 6.6a byla provedena detekce hran za použití metody Structured forrest, jejíž výsledek je zobrazený v 6.6b. Na oba snímky byly aplikovány aktivní kontury v přibližně stejných oblastech. Jak je vidět na těchto obrázcích, odstraněním mnoha zbytečných hran se aktivní kontura jednoduše zformovala kolem kostí.

Testovací snímky ve formátu *DICOM* (Digital Imaging and Communications in Medicine) byly získány z projektu: Retrospective Image Registration Evaluation Project.

6.4 Porovnání proti klasickým detektorům

Tato část odpovídá na otázku, zda v uvedených specifických problémech je použití trénovatelného detektoru lepší volbou než použití klasických detektorů s vyladěnými parametry. Zástupcem klasických detektorů byl zvolen Derichův detektor. Tento detektor totiž na rozdíl od ostatních obsahuje jediný parametr α, kterým lze upravovat množství detailů v obrázku. Dalšími parametry k ladění Derichova detektoru jsou spodní a horní práh související s dvojitým prahováním a hysterezí 2.3.

Mapové podklady

Na základě prohledávání prostoru možných parametrů, nebyla nalezena vhodná kombinace, při které by detektor reagoval pouze na požadovaná data. Nicméně vhodnou úpravou parametrů lze odstranit alespoň část nechtěných hran. Hrany, které lze úspěšně odstranit náleží převážně vrstevnicím, jak je vidět na dvojicích obrázků 6.7b, 6.7e a 6.7c, 6.7f. Dále si na těchto obrázcích lze všimnout vlivu parametru α na odlišnou odezvu na hrany silnice. Větší hodnoty odpovídají detailnějšímu měřítku a tedy hranám na okrajích každé černé čáry.

(a) Originální výřez.

(b) Derichův detektor s parametrem $\alpha = 1$.

(c) Derichův detektor s parametrem $\alpha = 2$.

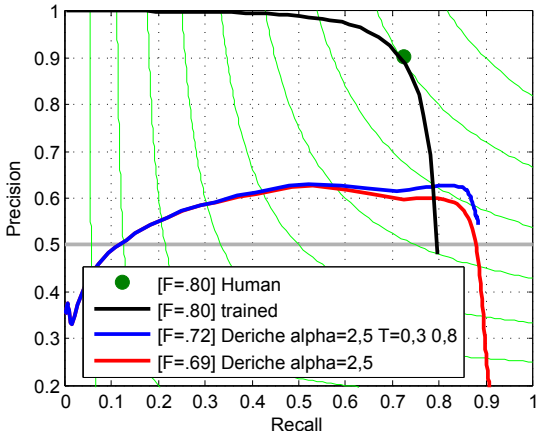
(d) Trénovatelný detektor.

(e) Derichův detektor s parametry: $\alpha = 1$, dolní práh = 40%, horní práh = 70%.

(f) Derichův detektor s parametry: $\alpha = 2$, dolní práh = 40%, horní práh = 70%.

Obrázek 6.7: Příklad trénovatelného a Derichova detektoru s různými parametry na mapových podkladech.

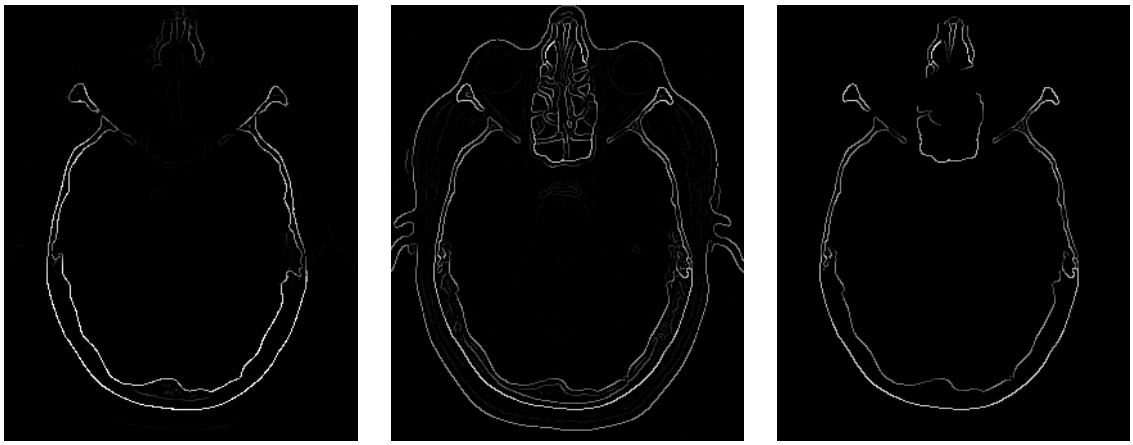
Klasickými detektory nešlo odlišit hrany, patřící k popisným textům. Proto ani úpravou parametrů nedocházelo k výraznějšímu zlepšení úspěšností detektorů hran. Porovnání precision - recall křivek trénovatelného detektoru (kombinace všech anotačních stylů) a Derichova detektoru je na grafu 6.8. Z grafu je patrné pouze malé zlepšení úspěšnosti detektoru při manipulaci s parametry detektoru. Nízké hodnoty parametru precision jsou způsobeny vysokým počtem označených nepravých hran (false positive).



Obrázek 6.8: Porovnání úspěšnosti trénovatelného detektoru Structure forest a klasického Derichova detektoru nad detekcí silniční sítě z mapových podkladů.

Kostní tkáň

V případě CT snímků hlavy, úprava parametrů Derichova detektoru vedla k odstranění většiny nechtěných hran. Příklad na obrázku 6.9b obsahuje všechny nalezené hrany klasickým detektorem. Po nastavení horního prahu na 60% hodnotu vidíme výsledek Derichova detektoru na obrázku 6.9c. Tento obrázek se opravdu podobá výstupu z trénovatelného detektoru (obrázek 6.9a).



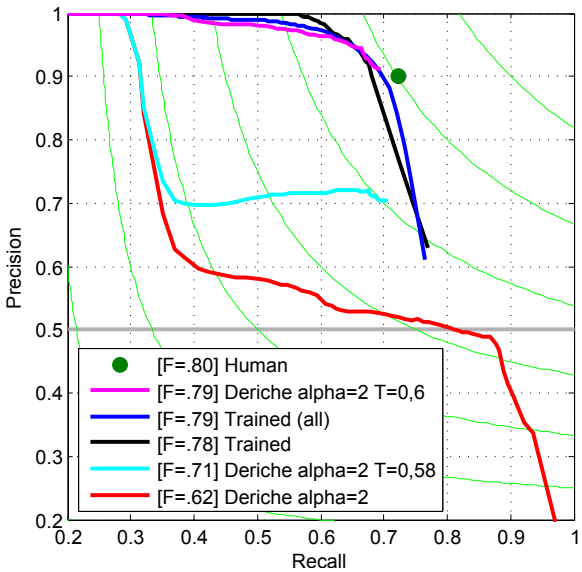
(a) Výsledek po použití trénovatelného detektoru.

(b) Výsledek po použití Derichova detektoru s parametrem $\alpha = 2$.

(c) Výsledek po použití Derichova detektoru s parametry: $\alpha = 2$, horní práh = 60%.

Obrázek 6.9: CT snímky hlavy.

Porovnání detekce hran kostí je na grafu 6.10. Křivky černá, červená a fialová náleží postupně zobrazeným obrázkům 6.9a, 6.9b a 6.9c. Azurová křivka zobrazuje vyhodnocení



Obrázek 6.10: Porovnání úspěšnosti trénovatelného detektoru Structure forest a klasického Derichova detektoru nad CT snímky s důrazem na hrany odpovídající kostní tkáni.

Derichova detektoru s nastaveným neoptimálním prahem. Tyto křivky zobrazují úspěšnost

detekce na vybraném obrázku. Modrá křivka odpovídá celému testovacímu datasetu, na který byl použit trénovatelný detektor Structured forest.

Z grafu je vidět, že metoda Structured forest se úspěšně natrénovala na rozhraní kostí. Také vidíme, že vhodnou modifikací parametrů klasického detektoru lze zvýšit úspěšnost detekce hran v této oblasti problému až k hodnotám podobným trénovatelnému detektoru. V případě klasického detektoru se ideální parametry pro každý snímek mírně lišili, což lze považovat za nevýhodu. Přístup založený na trénování také vykazoval větší robustnost. Pokud zanedbáme tyto nevýhody, potom použití trénovatelného detektoru nepřinášelo významný užitek.

Kapitola 7

Závěr

Práce představuje možné způsoby využití trénovatelného detektoru hran Structured forest for fast edge detection. Bylo prováděno množství experimentů nad vlastními datasety s cílem zjistit možnosti tohoto nového algoritmu detekce hran. Experimenty se skládaly z konkrétních úloh. První část se zaměřovala na detekci hran v historických mapových podkladech. Bylo ukázáno, že použitím trénovatelného hranového detektoru lze s určitou úspěšností najít linie odpovídající pouze cestám, železnicím, případně vodním tokům. Druhá část experimentů byla zaměřená na CT snímky a zvýraznění pouze některých významových hran odpovídající kostní tkáni.

Byl zkoumán vliv různých stylů anotací na výsledek detekce hran, včetně stanovení parametrů nejvíce ovlivňující kvalitu hran a jejich minimální požadované hodnoty. V určité míře lze část této práce chápat jako návod pro stanovení základních parametrů a způsobu tvorby anotovaného datasetu.

Ke zhotovení vlastního datasetu anotovaných hran pro trénování a vyhodnocování úspěšnosti byl vytvořen anotační nástroj. Formát anotovaných dat je ve shodě s formátem použitým v datasetu Berkeley. Program tedy umožňuje zobrazovat a upravovat tento známý dataset.

V rámci práce vznikla implementace některých klasických metod detekce hran: Marr-Hildreth, Cannyho, Derichův hranový detektor a filtrace anizotropickou difuzí. Pro Derichův detektor byly nalezeny kombinace parametrů, při kterých se výstup klasického detektoru přibližuje k výstupům trénovatelného detektoru. V případě detekce hran kostní tkáně klasické detektory dosahovaly podobné úspěšnosti. Naopak v mapových podkladech trénovatelný detektor dosahoval mnohem větších úspěšností.

Další pokračování práce by mohlo směřovat ke zvýšování úspěšnosti detekce železnic a cest v mapových podkladech.

Literatura

[1] Canny, J.: A Computational Approach to Edge Detection. *PAMI*, ročník 8, č. 6, 1986: s. 679–698.

[2] Dollár, P.: Piotr's Computer Vision Matlab Toolbox (PMT). http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.

[3] Dollár, P.; Zitnick, C. L.: Structured Forests for Fast Edge Detection. 2013.

[4] Forsyth, D. A.; Ponce, J.: *Computer Vision A Modern Approach*. Alan Apt, 2003, ISBN 0-13-191193-7.

[5] Lim, J.; Dollár, P.; Zitnick, C. L.: Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection. 2013.

[6] Ma, W.-Y.; Manjunath, B. S.: EdgeFlow: A Technique for Boundary Detection and Image Segmentation. 2000.

[7] Martin, D.; Fowlkes, C.; Tal, D.; aj.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc. 8th Int'l Conf. Computer Vision*, ročník 2, July 2001, s. 416–423.

[8] Microsoft: Structured Edge Detection Toolbox [online]. http://research.microsoft.com/en-us/downloads/389109f6-b4e8-404c-84bf-239f7cbf4e3d/.

[9] Perona, P.; Malik, J.: Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 12, 1990: s. 629–639.

[10] Sirdey, R.: A Gentle Introduction to the Deriche Optimal Edge Detector. 1998.

[11] Sumengen, B.; Manjunath, B. S.: Multi-scale edge detection and image segmentation. 2005.

Příloha A

Ovládání anotačního nástroje

Používání aplikace spočívá v obtahování kontur (levé tlačítko myši) objektů a následném obarvování souvislých oblastí (pravé tlačítko myši) aktuální barvou. Při vyplňování se za souvislou oblast považují pixely stejného typu třídy ve čtyřokolí pixelu. Pozice kurzoru, rozměry obrázku a informace o zvetšení obrázku jsou zobrazeny ve stavovém řádku aplikace.

- *CTRL + levé tlačítko myši*: načtení aktuální barvy z již anotované oblasti pod kurzorem myši.
- *ALT + levé tlačítko myši*: změna barvy pod kurzorem myši v celém obrázku.
- Stisknuté *prostřední tlačítko myši*, nebo šipky *vpravo*, *vlevo*, *nahoru*, *dolů*: posouvání výřezu obrázku.
- *skrolování*, *SHIFT + skrolování*: posouvání výřezu obrázku nahoru/dolů, respektive doleva/doprava.
- *CTRL + skrolování*: zvětšení/zmenšení obrázku se zaměřením na oblast, nad kterou se nachází kurzor myši.

Příloha B

Plakát