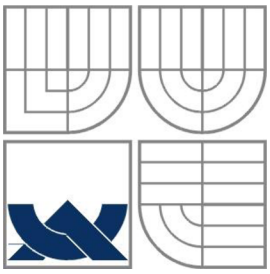
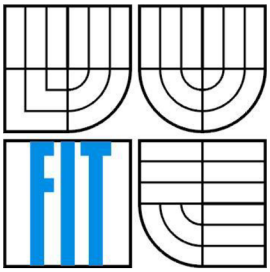


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAREK NOVÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL ŠPANĚL, PH.D.

BRNO 2012

Abstrakt

Detekce hran v obraze je jedním fundamentálních postupů užitých v oblasti zpracování obrazu a počítačového vidění. Cílem této práce je implementace a porovnání zvolených konvenčních a pokročilých metod, včetně porovnání jejich výkonnosti. Práce popisuje implementaci a evaluaci metody *Linked Edges as Stable Region Boundaries*. Úspěšnost metod je vyhodnocena za použití *Berkeley Segmentation Data Set and Benchmarks 500*.

Abstract

Edge detection is one of the fundamental techniques used in the fields of image processing and computer vision. Goal of this thesis is an implementation and evaluation of chosen basic and advanced edge detection methods, including performance evaluation. Thesis describes implementation and performance evaluation of *Linked Edges as Stable Region Boundaries* method. Performance is evaluated using *Berkeley Segmentation Data Set and Benchmarks 500*.

Klíčová slova

Detekce hran, Zpracování obrazu, Počítačové vidění, Edge linking, Canny edge detector, Segmentace obrazu

Keywords

Edge detection, Image processing, Computer vision, Edge linking, Canny edge detector, Image segmentation

Citace

Novák Marek: Pokročilé metody detekce hran v obraze, bakalářská práce, Brno, FIT VUT v Brně, 2012

Pokročilé metody detekce hran v obraze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Španěla, Ph.D.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Marek Novák

16. 5. 2012

Obsah

1 Úvod

Metody pro detekci hran patří mezi stěžejní postupy, využité v oblasti zpracování obrazu a počítačového vidění. Detekce hran umožňuje extrakci významných rysů obrazu a omezení jejich počtu před zpracováním. Hrany lze využít k rozpoznávání a klasifikaci objektů v obraze, jejich polohy, velikosti a jiných vlastností. Nalezené hrany mohou být využity ke zvýšení čitelnosti obrazu či jeho vylepšení, např. zaostřením obrazu [1].

Základním atributem sledovaným detektory hran jsou změny intenzity jasu v obraze. Metody pro detekci hran tato místa nalézají, upřesňují jejich pozici, omezují je a staví do logických celků. Často využitými metodami jsou filtrování obrazu, vyřazení krátkých či nevýznačných hran, hystereze výsledků a metody předzpracování vstupního obrazu.

Tématem této práce je implementace a porovnání základní a pokročilé metody detekce hran v obraze a vyhodnocení jejich úspěšnosti. Zvolenou pokročilou metodou je *Linked Edges as Stable Region Boundaries* (dále pod zkratkou *LESRB*), publikovanou v roce 2010 Michaelem Donoserem, Hayko Riemenschneiderem and Horstem Bischofem Institute for Computer Graphics and Vision Graz University of Technology [2]. Pro srovnání je využita starší, základní metoda, vytvořená J. F. Cannym, známá jako *Canny edge detector*, publikovaná v roce 1986 [3].

Výsledkem této práce je ukázková realizace obou metod v jazyce C++ za použití knihoven OpenCV a získání porovnatelného ohodnocení úspěšnosti těchto metod pomocí množiny lidsky anotovaných ukázkových dat. Zdrojem těchto dat a evaluační funkcionality je *Berkeley Segmentation Data Set and Benchmarks 500* [4] (ke stažení na webu [5]; dále pod zkratkou *BSDSB*), obsahující 500 ručně anotovaných obrázků určených, mimo jiné, k účelu porovnávání výkonnosti detekce hranic regionů obrazu.

Druhá kapitola této práce rozebírá principy a teoretické pozadí problematiky detekce hran a evaluace výkonnosti detektorů. Kapitola popisuje užitou terminologii a její význam.

Třetí kapitola obsahuje náhled do současných pokročilých metod detekce hran v obraze. Kapitola obsahuje shrnutí nejpoužívanějších přístupů a jejich specifických vlastností.

Čtvrtá a pátá kapitola popisují návrh a implementaci zvolených detekčních metod a testovacího rozhraní. Obsahem čtvrté kapitoly je i popis použité metodiky tréninku a evaluace metod.

Šestá kapitola prezentuje výsledky, kterých bylo dosaženo nad evaluačními daty testovací sady BSDSB. Kapitola rozebírá zjištěný vliv parametrů metod, použitý postup získání nejlepších výsledků a celkové vzájemné srovnání metod.

Závěr provádí diskuzi dosažených výsledků a možných budoucích uplatnění provedených implementací.

2 Detekce hran v obraze

Tato kapitola popisuje teoretické podklady k tématu a představuje zvolenou základní metodu *Canny Edge Detector*. Další podkapitoly se věnují problematice měření úspěšnosti metod a metodice jejich evaluace.

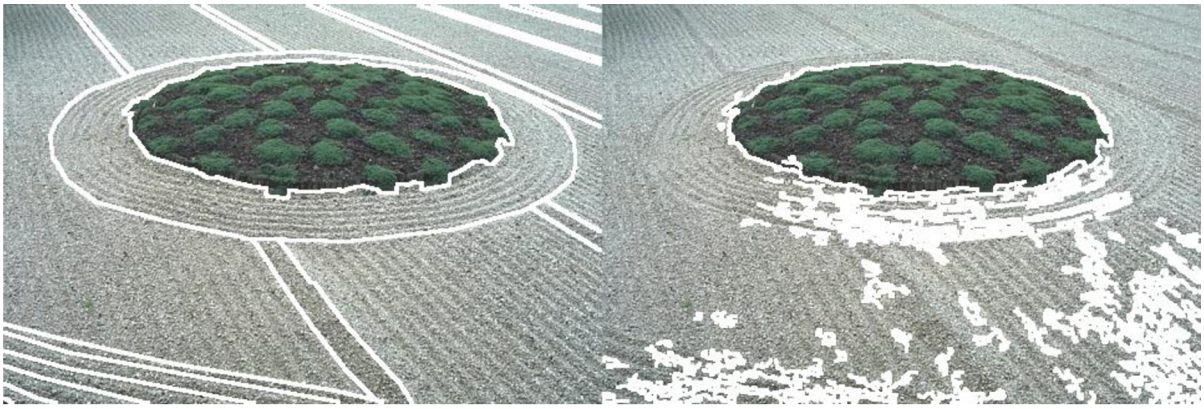
2.1 Fundamenty detekce hran v obraze

Podkapitola popisuje základní terminologii a definice, použité pro detekci hran.

2.1.1 Definice a klasifikace hran

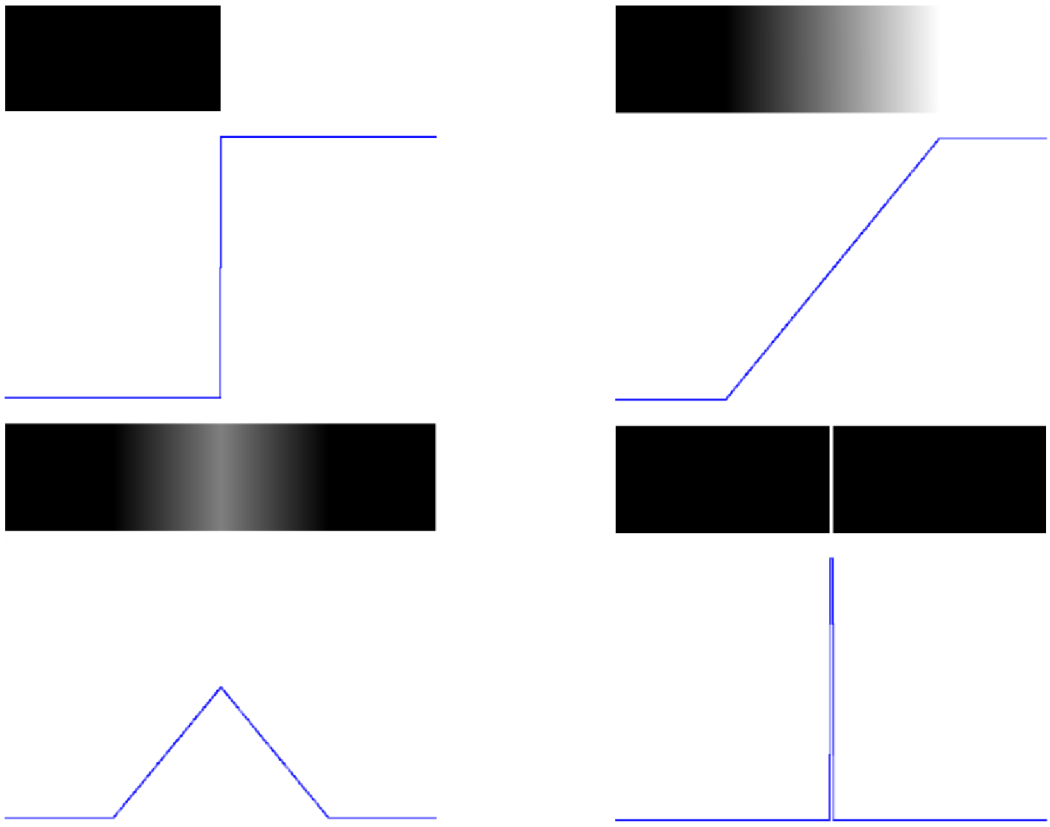
Formální definice hran v obraze je často předmětem diskuzí a kontextu. Existují dva základní koncepty, které se týkají hran. Koncept *hrany* (*edge*) je nejčastěji definován jako nespojitost intenzitě obrazu (v praxi nejčastěji jasu). Hrany jsou „lokálního“ významu a samy o sobě nevyjadřují nějakou vyšší strukturu.

Naopak *hranice* (*boundary*) jsou významu „globálního“ a korespondují ohraničení jednotlivých regionů v obraze. Regiony jsou množiny sousedících pixelů, spojených do celku na základě společného parametru, jako např. barvy nebo úrovně intenzity. Hranice regionů jsou uzavřené a jsou definovány jak pro vnější ohraničení regionů, tak pro jeho dutiny. Formálně lze hraniční pixely určit jako množinu pixelů, které sousedí s jinými pixely, než s těmi příslušejícím regionu. Hrany mohou být spojovány do větších celků, které nakonec mohou korespondovat s hranicemi regionů, ovšem toto nemusí být pravidlem.



Obrázek 1 – Hranice regionů (vlevo) a hrany (vpravo) (převzato z [5])

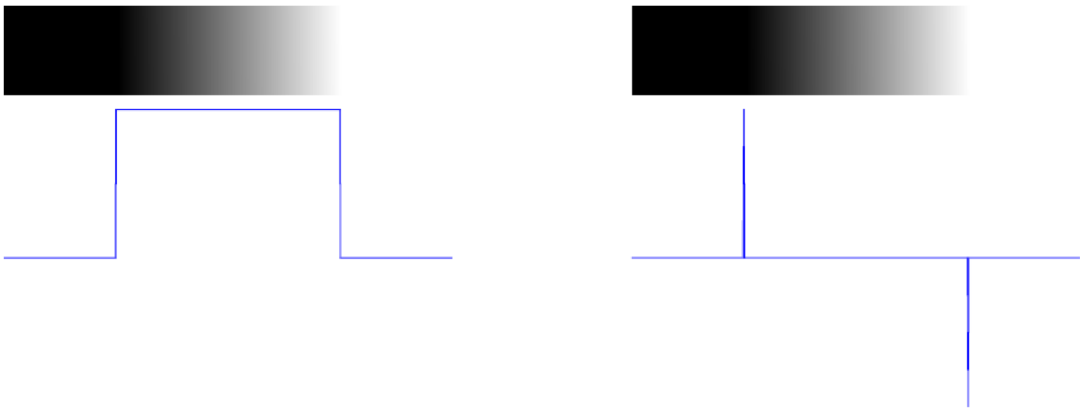
Existují tři základní modely hran, které se mohou v obraze vyskytnout. Prvním jsou *skokové* hrany (*step*), tvořené náhlou změnou v úrovni intenzity. Tyto hrany jsou nazývány binárními nebo také ideálními, díky jednoznačné detekci a lokalizaci. Druhým modelem jsou hrany *náběžné* (*ramp*). Náběžné hrany se více vyskytují v reálných obrazech a jejich délka je určena úrovní rozmazání obrazu. *Střechové* (*roof*) hrany jsou tvořeny nástupnou a sestupnou tendencí signálu. Speciální skupinu tvoří většinou *impulzní* (*impulse*) hrany, tvořené skokovým zvýšením intenzity, setrvání po délce jeden pixel a změnu zpět. Tyto hrany jsou klasifikovány zvlášť, protože jejich klasifikace pomocí dvojice hran skokových může vést k nechtěné duplikaci hran.



Obrázek 2 – Průbeh intenzity obrazu a odpovídající model hrany (zleva doprava, shora dolů: step, ramp, roof, impulse)

2.1.2 Detekce nespojitostí intenzity

Detekce nespojitostí je dosahováno detekcí změn intenzitě obrazu (zpravidla jasu černobílého obrazu). Metodou detekce změn průběhu intenzity je derivace zmíněného průběhu. Obrázek ilustruje první a druhou derivaci ukázkové ramp edge. Z průběhů funkcí lze vyčíst, že nenulovou hodnotu první derivace lze využít k detekci přítomnosti změny intenzity v obraze. Pozorováním druhé derivace si můžeme všimnout, že pro každou hranu tvoří dvojici extrémních hodnot, jejichž znaménko určuje směr změny intenzity obrazu. Tato duplikace není žádoucí, avšak spojením extrémních hodnot imaginární úsečkou získáme průsečíkem s nulou střed hrany, což je velice užitečné při omezování šířky hran.



Obrázek 3 – Průběh první (vlevo) a druhé derivace (vpravo) ramp edge

Ilustrované postupy v jednorozměrném prostoru lze aplikovat i na dvourozměrný prostor intenzity běžného obrazu. Tohoto se dosáhne aplikováním funkce odděleně ve směrech obou dimenzí a podobnou interpretací výsledků. Tímto získáme dvojici map změn intenzity obrazu pro oba směry.

Pro získání derivací diskrétního obrazu jsou použity rozličné aproximace 2D gradientu. Gradient obrazu $f(x, y)$ v bodě (x, y) je definován jako vektor:

$$\boldsymbol{\nabla f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix}. \quad (1)$$

Důležitou vlastností tohoto vektoru je jeho síla (magnitude, alternativně velikost) značená ∇f, kde:

$$\nabla f = mag(\boldsymbol{\nabla f}) = [G_x^2 + G_y^2]^{1/2}. \quad (2)$$

Hodnota ∇f se zpravidla v oblasti detekce hran označuje jako *gradient* (dále označován pod tímto názvem). Další důležitou kvantitou je *směr gradientu*. Směr gradientu $\alpha(x, y)$ v bodě (x, y):

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right), \quad (3)$$

je důležitý v pozdějších fázích hledání lokálních maxim gradientu a omezování šířky nalezených hran.

K získání hodnot derivací je využito *gradientních operátorů*. Gradientní operátory jsou konvoluční jádra, která se vyskytují zpravidla ve dvou (transponovaných) tvarech – pro oba směry derivace. Následující matice demonstruje jedny z primitivnějších operátorů, *Roberts cross-gradient operators*:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \quad (4)$$

Pro jednodušší implementaci jsou většinou použity operátory lichých rozměrů. Existuje více operátorů, avšak pro účely této práce jsou využity následující: *Sobelův* a *Gaussův*. Sobelův operátor šířky 3 má pro získání derivací ve směrech x a y následující tvar:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \quad (5)$$

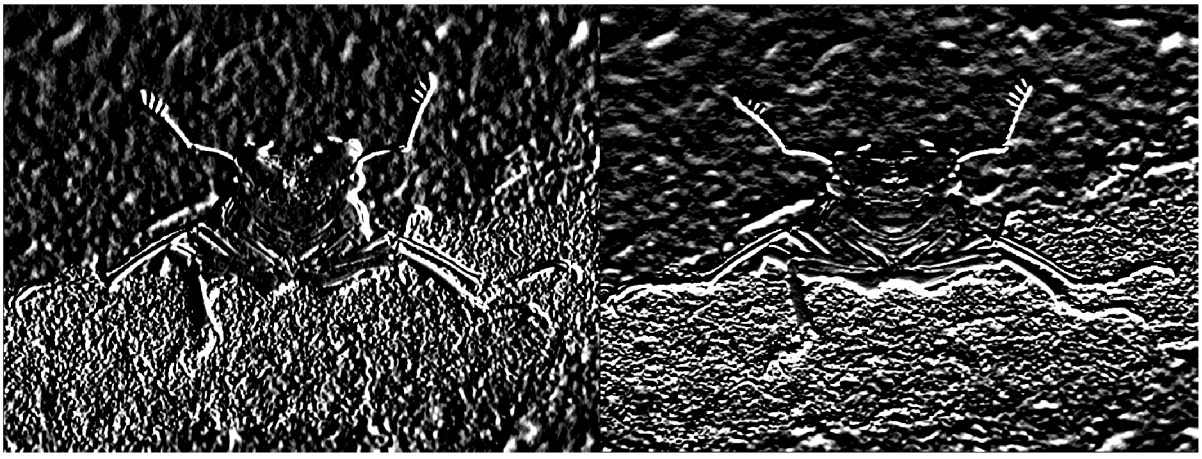
Gaussův operátor lze získat pomocí následujícího vztahu (první derivace dvourozměrné Gaussovy funkce s rozptylem σ):

$$G_x(x, y) = \frac{-x}{\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (6)$$

kde hodnoty x a y jsou vzdálenosti od středu matice, s minimem vlevo nahoře, ilustrovány na následujícím příkladě matice s šířkou 5:

$$\begin{bmatrix} (-2,-2) & (-1,-2) & (0,-2) & (1,-2) & (2,-2) \\ (-2,-1) & (-1,-1) & (0,-1) & (1,-1) & (2,-1) \\ (-2,0) & (-1,0) & (0,0) & (1,0) & (2,0) \\ (-2,1) & (-1,1) & (0,1) & (1,1) & (2,1) \\ (-2,2) & (-1,2) & (0,2) & (1,2) & (2,2) \end{bmatrix}. \quad (7)$$

Tento operátor je aproximací optimálního filtru, vyvíjeného J. F. Cannym [3]. Pro získání operátoru kolmého směru matici operátoru transponujeme. Z obrazu tak získáme hodnoty derivací ve dvou směrech, které můžeme použít k následnému vytvoření gradientu.



Obrázek 4 – Derivace černobílého obrazu v ose x (vlevo) a y (vpravo)

Získáním těchto komponent (výše označené jako G_x a G_y) můžeme pokročit k tvorbě samotného gradientu (∇f). Nejpřesnější metodou je použití *Eukleidovské metriky*, dle vztahu uvedeného výše (viz Rovnice č.(2)). Místy využívanou aproximací je použití *Manhattanské metriky*. Manhattanská metrika d_1 (taktéž rektilineární vzdálenost, L_1 vzdálenost, city block distance) dvojice bodů P a Q je definována jako součet absolutních hodnot rozdílů jejich jednotlivých dimenzí:

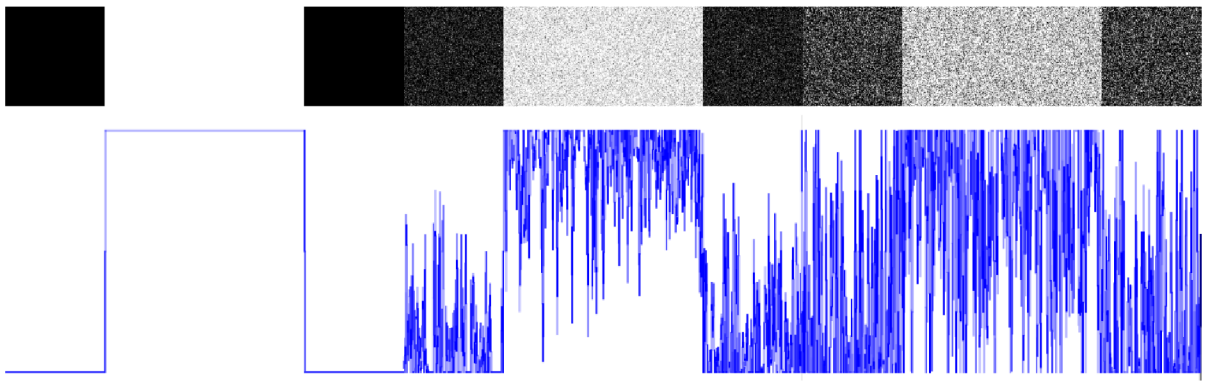
$$d_1(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{n} |p_i - q_i|, \quad (8)$$

kde n je počet dimenzí a vektory $\boldsymbol{p}$ a $\boldsymbol{q}$ jsou jednotlivé souřadnice všech dimenzí bodů P a Q.

Obrázek 5 – Výchozí obraz (vlevo) a gradient obrazu získaný euklidovskou metrikou (vpravo)

2.1.3 Problémy detekce

Modely hran zmíněné v úvodní části této podkapitoly, jsou demonstrovány na ideálních obrazech, které netrpí šumem a jinými vadami. Šum má velmi negativní vliv na výsledný průběh intenzity obrazu, což se odráží v hodnotách první a druhé derivace. Projevy šumu ve výsledných nalezených hranách zvyšuje počet falešných hran (false positives), stejně jako zhoršuje výsledky detekce skutečných hran (false negatives).



Obrázek 6 – Intenzita obrazu (nad křivkou) po aplikaci Gaussova šumu (zleva: bez šumu, σ=0,1, σ=0,5)

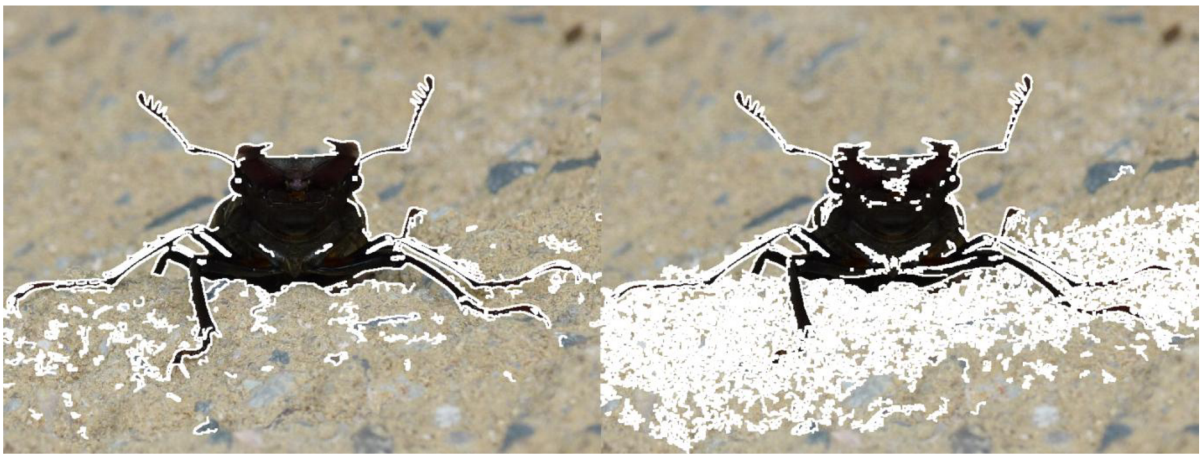
Řešením, které potlačí šum a volitelně sníží vliv příliš nevýznamných hran, je aplikace rozostření. Často používanou metodou je *Gaussovo rozostření*. Gaussovo rozostření spočívá v aplikaci filtru, který odpovídá dvojdimenzionální Gaussově funkci:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}, \qquad (9)$$

kde σ je rozptylem hodnot. Alternativně lze využít separovatelnosti této funkce a aplikovat filtr jednodimenzionální Gaussovy funkce pro oba směry os dimenzí (v jednom směru transponovanou):

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{x^2}{2\sigma^2}}. \qquad (10)$$

Vzhledem k tomu, že rozostření nerozlišuje mezi šumem a signálem v obraze, aplikací rozostření můžeme ztratit i část relevantních výsledků. Tato vlastnost je občas žádaná, jelikož omezí počet málo významných hran.



Obrázek 7 – Ukázka vlivu Gaussova rozostření na detekci hran; vlevo rozostřeno, vpravo bez rozostření (detekce: Canny edge detector; rozostření: σ=1,0 šířka filtru 9); bílé křivky jsou detekované hrany

2.2 Canny edge detector

Canny edge detector byl vytvořen J. F. Cannym a publikován v roce 1986 [3]. Tato metoda je považována za jedny ze standardních přístupů k detekci hran, a přestože jde o starší metodu, je stále využívána v praxi i v oblasti výzkumu (např. [6] [7]).

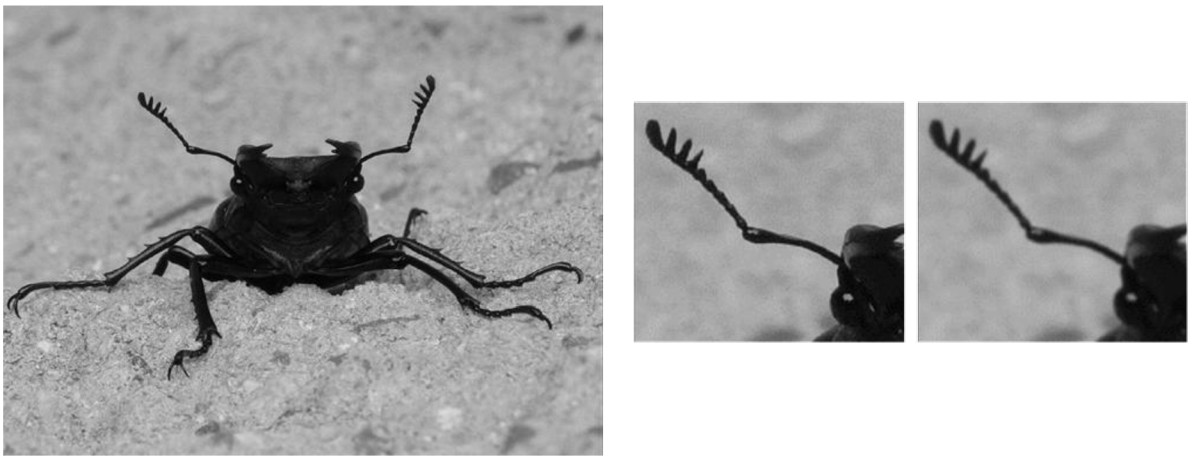
Cíle deklarované při vývoji této metody jsou následující:

- Maximalizovat pravděpodobnost detekce skutečných bodu hran, při minimalizaci chybně detekovaných pixelů.
- Lokalizovat nalezené hrany co nejblíže skutečným.
- Jednotlivé skutečné hrany by měly odpovídat pouze jediné detekované hraně (diskutabilně implikováno prvním bodem).

Cannyho metoda je díky své formulaci optimální při detekci step edges. Algoritmus lze rozdělit do pěti navazujících kroků.

V prvním kroku dochází k předzpracování vstupního obrazu, za účelem odstranění šumu. Tohoto se dosáhne rozmazáním obrazu Gaussovým rozostřením. Následující příklad demonstruje typickou konvoluční matici Gaussova rozostřujícího filtru s rozptylem $\sigma = 1{,}4$ a šířky 5:

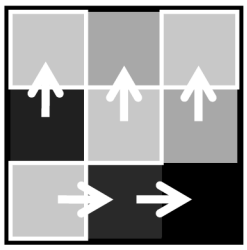
$$\frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}. \qquad (11)$$



Obrázek 8 – Gaussovo rozostření aplikováno na obraz (originál vlevo, výřez uprostřed, rozostřený výřez vpravo)

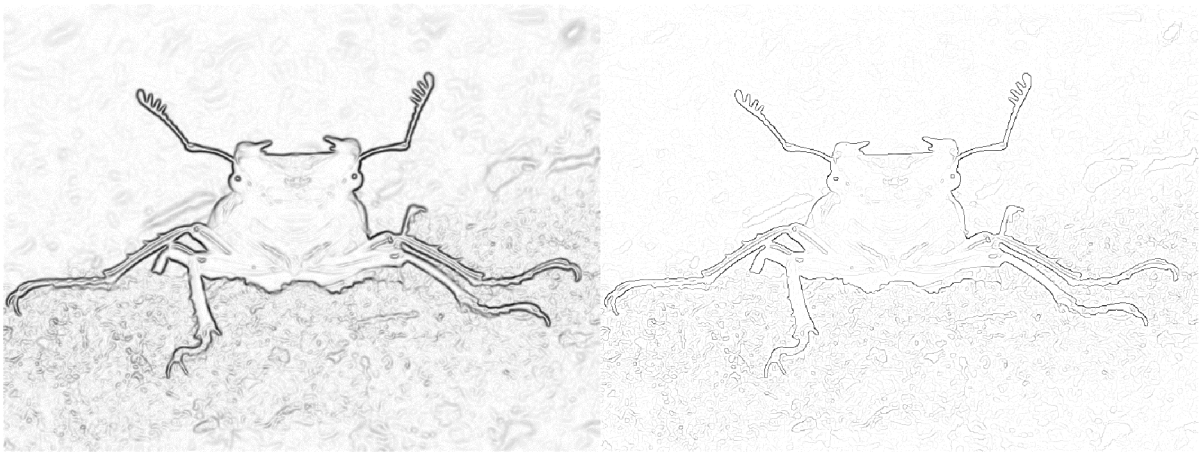
Následujícím krokem je nalezení gradientu. Pomocí Sobelova operátoru jsou nalezeny směrové derivace G_x a G_y, následné získání sil gradientů je provedeno jejich kombinací Euklidovskou metrikou. Některé mutace metody používají efektivnější Manhattanské metriky. Zároveň je získána mapa směrů gradientu (dle rovnice č. (3)).

Získané směry jsou využity v dalším kroku metody – potlačení nemaximálních hodnot. Směry jsou zaokrouhleny na násobky 45° pro rozmezí 0°- 135°. Tyto hodnoty udávají který ze sousedních bodů osmiokolí bude použit k potlačení, kdy 0° je pixel napravo od zkoumaného a směr otáčení je proti směru hodinových ručiček. Potlačení je provedeno porovnáním příslušných sousedících pixelů a potlačením těch, které mají menší hodnotu.



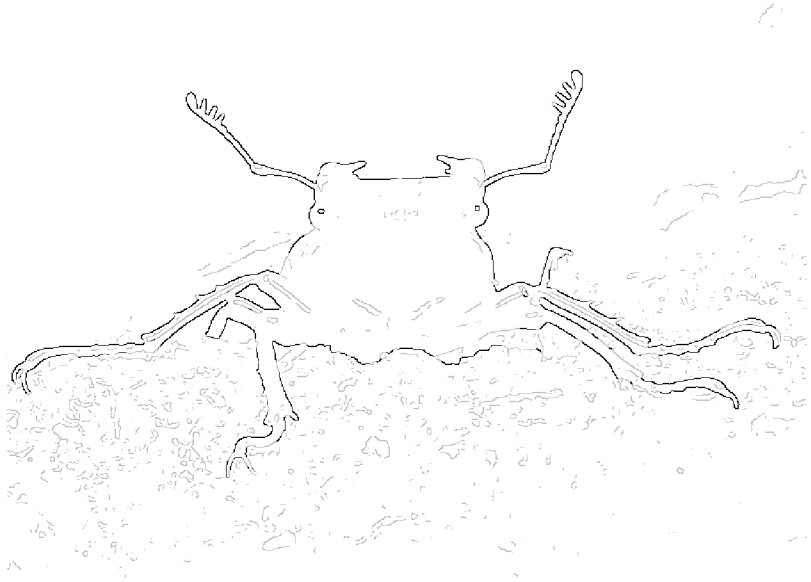
Obrázek 9 – Ukázka potlačení nemaximálních hodnot; šipky ukazují směr gradientu; bíle orámované pixely jsou nalezenými maximálními hodnotami

Výsledkem operace jsou zúžené hrany gradientu, označované jako *kandidátní hrany*.



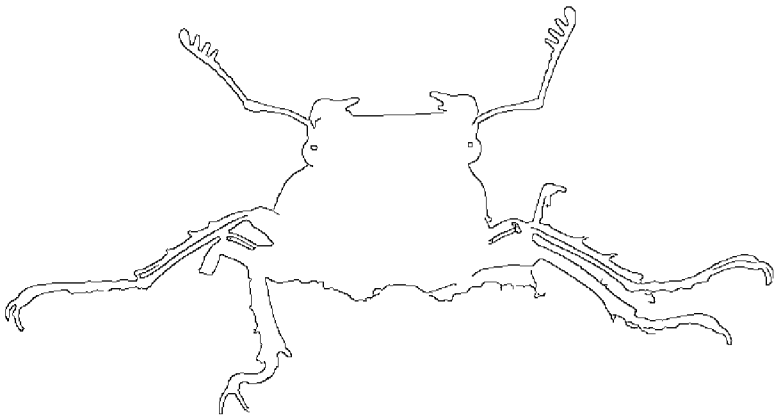
Obrázek 10 – Invertovaný gradient před (vlevo) a po potlačení nemaximálních hodnot (vpravo)

Ve čtvrtém kroku dochází k prahování obrazu dvojicí prahů. *Spodní práh* určuje maximální hodnotu pixelů kandidátních hran, které budou zahozeny. Jinými slovy, jakýkoliv pixel gradientu hodnoty pod tímto prahem je potlačen. *Vyšší práh* určuje, které hrany budou označeny jako *silné hrany*. Silné hrany, které jsou vždy obsaženy ve výsledném obraze, mají pixely s hodnotou nad tímto prahem. Zbývající interval hodnot mezi těmito prahy (větší hodnota jako nižší práh a zároveň menší jako vyšší práh) je použit pro označení *slabých hran*. Slabé hrany se vyskytují ve výsledném obraze pouze, pokud jsou napojeny na silné hrany. Nastavení prahů je typickým parametrem metody, který určuje počet a kvalitu navrácených hran metodou.



Obrázek 11 – Silné hrany obrazu (černé) a slabé hrany obrazu (šedé) při nastavení prahů na hodnoty 60 a 180

Slabé hrany jsou zpracovány ve finálním kroku metody – *hysterezi hran*. Použitím libovolné iterační metody jsou všechny slabé hrany prozkoumány, a pokud jsou napojeny na jakoukoliv silnou hranu, jsou zakomponovány do výsledného obrazu. Pokud slabé hrany leží samostatně (např. ve spodní části ilustrace), jsou zahozeny.



Obrázek 12 – Výsledná mapa hran po aplikaci hystereze

2.3 Porovnávání výkonnosti metod

Tato podkapitola popisuje metriky a terminologii použitou při evaluaci výkonnosti metod.

2.3.1 Kritéria

Ke klasifikaci výsledků detektorů hran při porovnání s referenčními daty je používána kategorizace do čtyř tříd, dle rozdělení na chybné versus správné a nalezené versus nenalezené výsledky (viz Tabulka 1).

		Skutečný výsledek	
		Nalezena hrana	Nenalezena hrana
Předpokládaný výsledek	Nalezena hrana	*True positive*, správný výsledek	*False negative*, absence výsledku
	Nenalezena hrana	*False positive*, nepředpokládaný výsledek	*True negative*, správná absence výsledku

Tabulka 1 – Klasifikace úspěšnosti výsledků detekce

Tyto kategorie slouží ke kvantifikaci úspěšnosti a chybovosti detektorů a pro výpočet dalších metrik. V kontextu statistiky mluvíme v případě false positives a false negatives jako o chybách typu I a typu II.

K porovnání výkonnosti metod je nutné stanovit měřená kritéria a jejich význam. Typickými kritérii pro ohodnocení výkonnosti metod mohou být (převzato z [8]):

1. pravděpodobnost false positive (kolik existuje chybně nalezených ne-hran),
2. pravděpodobnost false negative (kolik existuje chybně nalezených hran),
3. chyba při určení úhlu hrany,
4. střední kvadratická vzdálenost nalezených hran od skutečných,
5. tolerance metody ke zkresleným segmentům hran, jako zakončení a rozpolcení či zkřížení.

Avšak měření těchto kritérií je v mnoha případech komplikované. První komplikací je otázka, zda je vůbec možné sestavit referenční mapu „správných“ hran a jak by měly tyto hrany být určeny. Třetí a čtvrté kritérium tedy nejsou pro účely porovnání velmi užitečné. Zakončení hran a křížení hran taktéž nemusí být příliš užitečné a ohodnocení jejich zpracování detektorem je sporné.

Nejdůležitějšími kritérii jsou tedy kritérium první a druhé. Chybné označení hrany může vést k mnohem výraznějším chybám při následném použití detektorů. Většinou je považováno za více blízké „ideálu“, pokud jsou jednotlivé hrany detekovány, než zda jsou detekovány v naprosto správném formátu.

2.3.2 F-score

Pro účely generování numerického hodnocení detekce ze zmíněných kritérií je použito F-score (také známo jako F_1 score nebo F-measure). F-score je tvořeno kombinací dvou metrik, *precision* a *recall*. Výsledkem F-score je hodnota z rozsahu 0 až 1, kterou lze použít k přímému porovnání výsledků (vyšší hodnota znamená lepší výsledek).

Precision udává kolik z nalezených elementů (v našem případě pixelů hran) je skutečně nalezeno správně. Užitím hodnot tp, určujících počet true positives a fp, určujícím počet false positives můžeme získat precision použitím následujícího vztahu:

$$Precision = \frac{tp}{tp+fp}. \tag{12}$$

Recall udává kolik z elementů, které očekáváme, že budou nalezeny, je skutečně nalezeno. Za použití dříve definované hodnoty fp a hodnoty false negatives fn lze získat recall následujícím vztahem:

$$Recall = \frac{tp}{tp+fn}. \tag{13}$$

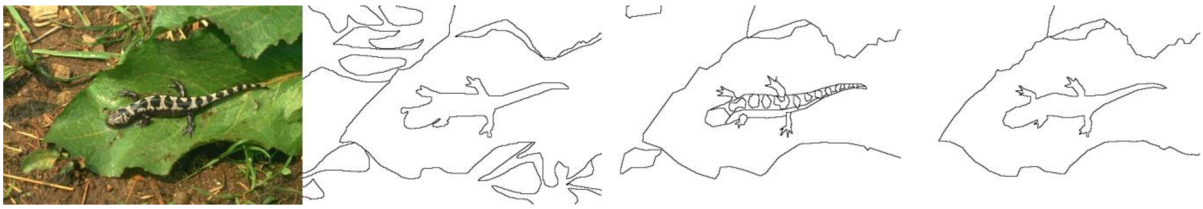
Výsledné F-score F lze získat pomocí následujícího vztahu:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}. \tag{14}$$

Tento vztah používá shodnou váhu pro oba parametry precision a recall. Váha může být libovolná, mezi v praxi používané patří např. $F_{0,5}$ score, které dává větší význam precision. Pro účely vlastního zhodnocení čtenářem je publikováno tedy nejen F-score, ale i parametry použité pro jeho výpočet.

2.4 Berkeley Segmentation Data Set and Benchmarks

Berkeley Segmentation Data Set and Benchmarks (dále *BSDSB*) je kombinací 500 anotovaných obrázků (v původní verzi 300) a evaluačních metod pro empirické ověření úspěšnosti segmentace obrazu a detekce hranic regionů, vyvíjený Computer Vision Group Berkeley University of California. Obrazy jsou anotované lidmi, kdy každá předloha (*ground truth*) je složena z jednotlivých segmentací provedených několika osobami.



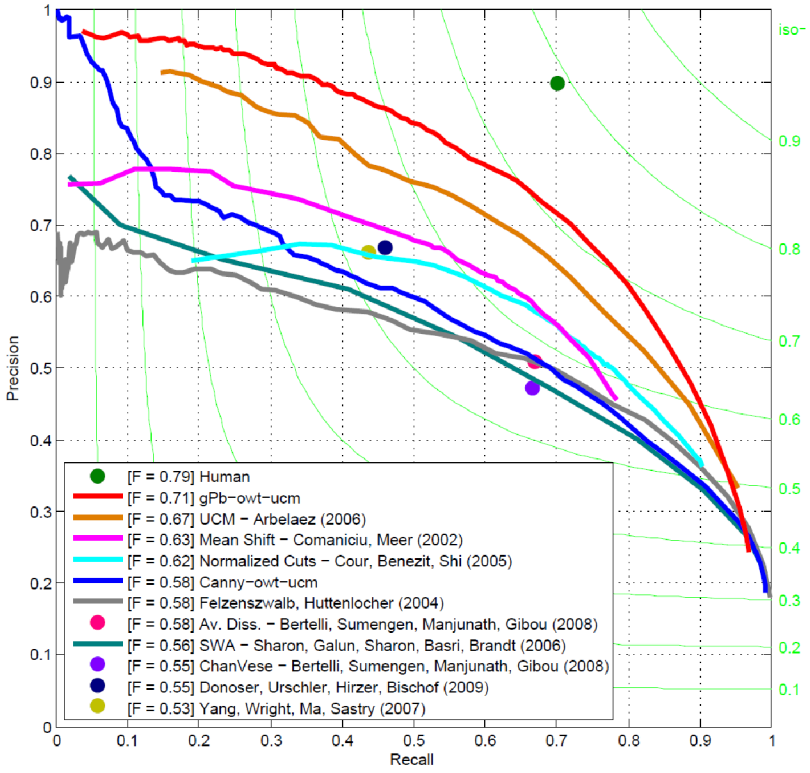
Obrázek 13 – Testovací obraz BSDSB a trojice lidských anotací (převzato z [5])

Z funkcionality tohoto balíku je využit *boundary benchmark*, sloužící k evaluaci úspěšnosti nalezení hranic regionů v obraze. Vzhledem k tomu, že detektory hran jsou často používány právě k vyhledávání hranic regionů, je toto měření vypovídající, pokud je cílem detekce najít co nejvýznamnější hranice v obraze. Vstupem boundary benchmark můžou být jak „měkké“ hranice z rozpětí šedi nebo „tvrdé“ hranice, tvořené pouze bílými hranami a černými pozadím.

Boundary benchmark provádí následující zjednodušený postup evaluace:

- Každý vstupní obraz hran je prahován dle předem zvoleného počtu prahů (výchozí hodnota je 99). Prahy jsou lineárně určovány po celém rozsahu hodnot obrazu. Prahování má význam pouze pro měkké mapy hran, u tvrdých map jsou výsledky všech prahů shodné.
- Pro každý prahovaný obraz jsou hrany standardizovány zeštíhlením a nastavením pixelů hran na shodnou hodnotu.
- Pro každou referenční segmentaci je získáno ohodnocení korespondence pixelů s prahovaným obrazem.
- Ohodnocení jsou zkombinována a je vygenerováno precision a recall pro každý obraz.

Výsledné ohodnocení pro každý soubor a práh jsou zkombinovány a je získáno trojí ohodnocení. Ohodnocení pomocí *optimálního prahu množiny dat* (*optimal dataset scale – ODS*) používá nejoptimálnější práh shodný pro všechny obrazy tak, aby bylo dosaženo co nejlepšího výsledku. Ohodnocení *optimálním prahem obrazů* (*optimal image scale – OIS*) navrací typicky lepší výsledek, použitím různého nejlepšího prahu pro každý obraz zvlášť. Třetím výstupem je *precision-recall křivka*, která udává výsledky průměrných hodnot všech zkoumaných prahů obrazů. V případě použití



Obrázek 14 – Precision-recall křivka výsledků různých metod (převzato z [4])

tvrdých dat je výstupem jediný bod (je použit jediný práh).

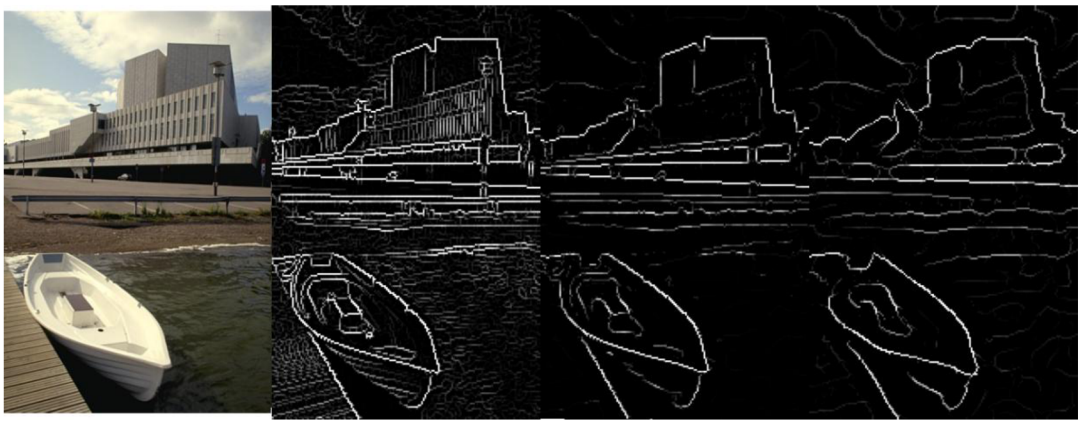
Množina dat BSDSB je dělena na tři oddělené sady, tréninková, validační a testovací. Uvedené „Best Practice Guidelines“ definují doporučené použití sad pro přípravu a samotné měření. Před samotným měřením je vhodné provést trénink parametrů, tj. nalezení optimálních parametrů evaluace. Trénování by mělo být prováděno pouze nad tréninkovou a validační sadou. Finální testování za použití natrénované konfigurace je prováděné jediným spuštěním nad testovací sadou. Tréninková a testovací sada obsahují každá 200 obrázků, validační 100.

Celý balík je implementován pro prostředí MATLAB za použití vnitřního jazyka a C++. Dostupné je sestavení pro Linux (MATLAB Executable files) i zdrojové kódy. Evaluace může trvat dle nastavení i několik hodin.

3 Současné metody detekce

Využití detektorů hran v oblastech rozpoznávání a klasifikace (např. [9], [10] nebo [11]) se vyznačuje vyšším důrazem na „globální“ význam hran, především pro účely segmentace obrazu. Z tohoto důvodu vzniklo množství modifikovaných a pokročilých metod, vnímajícím význačnost hran vzhledem k jiným parametrům, než lokálním změnám intenzity obrazu.

Jedním z pokročilých přístupů detektorů je detekce za použití více na více škál obrazu – *multi-scale edge detection* (např. [12], [13]nebo [14]). Tento postup umožňuje určit význačnost hran na základě jejich výskytu ve více škálách, zpravidla získaných půlením dimenzí obrazu. Detekce na obrazu vyšší škály sebou přináší problém horší lokalizace hran, avšak přináší redukci nevýznamných hran. Používané metody jsou postaveny na rozličných přístupech ke kombinaci škál



Obrázek 15 – Multi-scale edge detection; zleva: originál, detekce na základní škále, detekce na vyšší škále, kombinace škálovaného a neškálovaného obrazu (převzato z [12])

Dalším používaným postupem je spojování hran (*edge linking*), které funguje na základě spojování nezávisle na sobě nalezených hran. Jednoduchým příkladem spojování hran je nalézání hran ve více předdefinovaných směrech a jejich kompozice. Některé metody používají postupy vyhledávání regionů a spojování jejich hranic [2]. Metrikou kvalitní hrany může být rozsah parametrů vyhledávání, při kterých se projeví – stabilita hrany. Jednoduchá implementace postupu spojování hran je i hystereze, používaná v metodě Canny edge detector [3], kdy jinak vyloučené hrany jsou zpracovány, díky napojení na silné hrany.

Běžnou praxí metod detekce hran je použití datových struktur s vyšší mírou abstrakce, jako např. stromové struktury [13] [2] nebo hexagonální spirální struktury [15].

Obrázek 16 – Příklad spirální architektury (převzato z [15])

4 Návrh detektoru hran a metodika evaluace

Kapitola popisuje návrh realizace testovaných metod pomocí jazyka C++ a knihoven OpenCV nad platformou Windows. Zároveň popisuje metodiku tréninku a evaluace detektorů pomocí boundary benchmark BSDSB. Návrh se snaží o modularitu a čistotu řešení, stejně jako o zjednodušení případné portace na jinou platformu.

4.1 Testovací rozhraní a formát metod

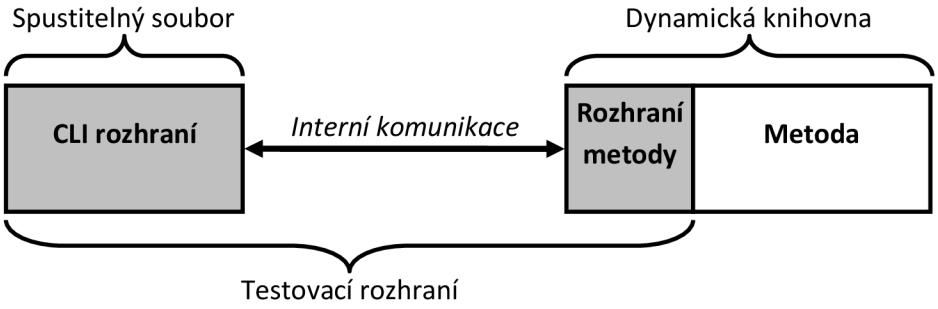
Pro sjednocení procesu spouštění a testování metod bylo přistoupeno k vytvoření izolovaného testovacího rozhraní. Vzhledem k nízkým nárokům na uživatelskou interakci a prakticky nulových nároků na interaktivou je rozhraní čistě textové. Analýza parametrů metod přinesla následující požadavky na funkcionalitu rozhraní:

- Rozhraní musí být schopno přijímat vstupní soubory (ideálně v dávkách) a zpracovat je pro potřeby metod.
- Uživatel musí být schopen přesně deklarovat, jaká metoda má být použita.
- Rozhraní musí být schopno delegovat proměnnou množinu parametrů metod zadaných uživatelem.
- Rozhraní musí ošetřit jakékoliv výjimečné stavy, které mohou vzniknout při běhu metod, neplatnými vstupy nebo z jiných důvodů.
- Rozhraní musí zajistit uložení výstupních souborů a jejich správné formátování.
- Formát vstupního souboru by měl být nastavitelný uživatelem.
- Pokud je to možné, rozhraní by mělo zajistit optimalizaci běhu pomocí paralelní exekuce nad dávkou souborů.

Vzhledem k požadavku změny použité metody během běhu, zároveň kvůli omezení zátěže při sestavování programu a zvýšení modularity řešení bylo přistoupeno k rozdělení sestavení na spustitelné textové rozhraní a jednotlivé metody, implementované jako dynamicky připojované knihovny (DLL, shared objects). Knihovny mají shodné rozhraní, používané spustitelnou aplikací. Rozhraní metod je specifikováno v jazyce C/C++ následovně:

```
extern "C" void __cdecl EdgeDetection(
	cv::InputArray input, cv::OutputArray output, std::string configFile = "");
```

Nárok na modularitu řešení vznesl problém s řešením propagace nastavení metod. Každá metoda může specifikovat množinu parametrů libovolného počtu či typu. Tyto parametry musí být zároveň přívětivě uživatelsky nastavitelné dle potřeb. Tento požadavek způsobil přenesení části rozhraní přímo do knihoven metod a zvolení externího souboru jako úložiště nastavení. Tato změna přenáší definici formátu a zpracování konfiguračního souboru na autora metody, resp. autora jejího kompatibilního rozhraní. Dynamické připojování knihoven a fixní formát jejich komunikace umožňuje připojení jakékoliv jiné metody k již existujícímu testovacímu rozhraní.



Obrázek 17 – Rozsah testovacího rozhraní

4.2 Linked Edges as Stable Region Boundaries

Výhodou metody Linked Edges as Stable Region Boundaries (dále *LESRB*) je možnost použití výstupních hran ve formátu strukturované kolekce souřadnic a jejich význačnosti. Takový výstup je v mnoha případech vhodnější pro další zpracování než bitmapa se zakreslenými hranami, jaká je výstupem jiných detektorů, jako např. Canny edge detector. Cílem návrhu je tedy tuto funkcionalitu zachovat, stejně jako poskytnout export ve formátu bitmapy měkkých a tvrdých hran. Předlohou pro implementaci bylo demo, publikované na webu autorů metody [16] implementované pomocí MATLAB a C++. Běh předlohy byl rozčleněn na čtveřici kroků.

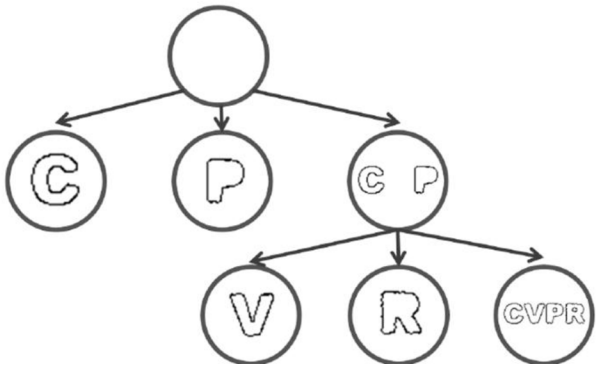
V první kroku byl získán gradient obrazu pomocí Gaussova gradientního operátoru, včetně aplikace preprocesu ve formě rozmazání obrazu. Metoda provádí adaptivní určení šířky filtrů jejich generováním, dokud hodnoty hraničních buněk konvolučních matic neklesnou pod minimální práh.

V následujících dvou krocích je provedena konstrukce a analýza *stromu komponent*. Strom komponent je abstraktní datová struktura, která je jádrem LESRB a je tvořen předem daným počtem úrovní.

Pro konstrukci jednotlivých *úrovní stromu komponent* je použitý obraz vytvořený prahováním gradientu. Použité prahy jsou získány lineárním rozdělením intervalu prahů. Limitní hodnoty prahů jsou zadány parametricky nebo jsou získány podle histogramu gradientu. Obraz získaný prahováním gradientu je použit k vyhledání regionů příslušejících ke zpracovávané úrovni stromu komponent. Nalézané regiony jsou tvořeny souvislými plochami nižších hodnot gradientu než je použitý práh a musí splňovat parametry minimální plochy a minimální délky hranice regionu. Použitý práh je označován jako *význačnost* (*saliency*) a v pozdějších fázích určuje ohodnocení získaných hran. Před zpracováním regionů je u důvodů omezení množství malých hran a vylepšení významu rozlehlých regionů provedeno vyplnění „děr“ v prahovaném obraze, které plochou nepřesahují parametr minimální plochy nalezeného regionu (v demonstračním obrázku na straně 22 si můžete všimnout později vyplněných děr regionů „P“ a „R“). Získané upravené regiony jsou označeny unikátní identifikací, a včetně výčtu jejich vnějších a vnitřních hranic zařazeny do tvořené úrovně.

Obrázek 18 – Regiony nalezené postupným prahováním gradientu (převzato z [2])

Ke každému regionu – *uzlu úrovně* je následně nalezen příslušející *otcovský uzel*. Jelikož tvorba stromu postupuje od nejvyšší úrovně význačnosti po nejnižší, validní uzel stromu je vždy plně obsažen uvnitř plochy otcovského regionu s vyšší význačností. Například na demonstračním příkladu regiony nižší význačnosti (úrovně t+1) obsahující písmena „V“ a „R“ patří do regionu tvořícího „pozadí“ předchozí úrovně (t). Kořenový uzel je tvořen celým obrazem bez jakýchkoliv hranic (hranice obrazu se mezi hranicemi žádného regionu nevyskytují).



Obrázek 19 – Vizualizace hranic regionů jednotlivých uzlů stromu komponent (převzato z [2])

Po dokončení konstrukce stromu komponent je vyvolána analýza. Analýza provádí porovnávání jednotlivých hran uzlů s jejich příslušejícími otci ležící o n úrovní výše, dle zadaného parametru. Pixely, které splní požadavky stability dané parametry metody, jsou označeny za validní a použity k zápisu do *obrazu význačných gradientů* (*saliency image*). Obraz význačných gradientů je tvořen hodnotami gradientu, ekvivalentního umístění jako pixely validních hran a jejich nejbližšího okolí do vzdálenosti 3px, vynásobených význačností úrovně, které nalezená validní hrana přísluší.

Samotné porovnávání využívá simplifikovanou verzi *chamfer matching* za pomoci *distanční mapy* (získané *distance transform*) hran regionu. Distanční mapa je porovnávána s jednotlivými otcovskými hranami. Pro každý pixel hrany je z distanční mapy získána jeho vzdálenost k nejbližší hraně „syna“, pokud tato vzdálenost neklesne pod parametrický práh, je pixel považován za validní. Na vzniklý řetězec je aplikována heuristika v podobě spojení nesouvislých segmentů, které jsou odděleny

maximálně pěti nevalidními pixely. Validita celého řetězce hran, vzniklých popsaným způsobem z hranic regionu je určena parametrem poměru validních a nevalidních pixelů. Pokud řetězec splňuje tento parametr, je úspěšně rozdělen na jednotlivé segmenty – stabilní hrany a spolu s jejich význačností zapsán do seznamu navrácených hran.

Obrázek 20 – Získání stabilních hran; hrany regionu na úrovni 22 (vlevo nahoře), hrany otcovského regionu na úrovni 10 (vpravo nahoře), odpovídající segment validní hrany a jeho okolí (vlevo dole), ekvivalentní hodnoty zapsané do saliency image (vpravo dole)

Po úspěšném provedení analýzy je základní běh metody ukončen. Následující zpracování závisí na využití výstupu metody a zvoleném formátu. V úvahu přichází trojice formátů, které může metoda poskytovat:

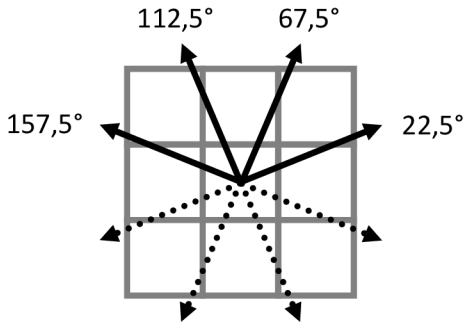
- Vnitřní formát tvořený seznamy bodů jednotlivých hran a ohodnocení jejich význačnosti.
- Mapa tvrdých hran získaných potlačením lokálních maxim a prahováním.
- Mapa měkkých hran získaných potlačením lokálních maxim a ekvalizací.

První dva formáty nejsou pro účely této práce použity, avšak pro účely znovupoužití kódu jsou přítomny. Výstup vnitřního formátu je zajištěn ukládáním výstupu analýzy a jeho zpřístupněním.

Předloha, ze které návrh vychází, používala export do formátu tvrdých hran, tato funkcionalita byla tedy adaptována bez významných změn. Formát tvrdých hran používá histogramu obrazu význačných gradientů a konstantního očekávaného poměru množství pixelů hran a pixelů pozadí k určení prahu, který je použit pro získání obrazu výstupních hran. Výstupní hrany jsou lokalizovány a prahovány způsobem ne nepodobným tomu použitému v Canny edge detector. Využita je tedy dvojice prahů pro určení silných a slabých hran. Následně je provedena obdobná hystereze a omezení délek hran podle parametrického minima.

Tvorba měkkých hran je podobná, avšak k prahování zde nedochází vůbec. Aby byla omezena chyba tvořená zaokrouhlením hodnot do výstupního formátu, obraz je ekvalizován tak, aby 20% pixelů hran nejvyšší hodnoty bylo označeno maximem.

Výstupní hrany jsou lokalizovány pomocí vyhledání lokálních maxim, využitím principu získání směru gradientu, popsaném v podkapitole 2.1.2. Algoritmus využívá směru gradientu získaného pomocí směrových derivací zaokrouhleného na čtveřici hodnot s rozestupem 45°, první hodnoty 22,5° (viz Obrázek 21). Směry opačné lze díky symetrii tohoto modelu zanedbat (nalezená lokální maxima by byly stejné).



Obrázek 21 – Směry detekce lokálních maxim použité LESRB

Uživatelsky konfigurovatelné parametry metody byly převzaty z demonstrační předlohy. Jako formát konfiguračního souboru byly zvoleny YAML a XML, z důvodu dobré podpory serializace v OpenCV. Výčet parametrů určuje tabulka níže.

Název	Popis	Typ
`gradientLowThreshold`	Minimální hodnota prahu použitého pro získání regionů tvořících strom komponent. *Hodnota z rozsahu 0–1 nižší jako maximální hodnota nebo hodnota -1 pro automatické zjištění gradientů.*	`double`
`gradientHighThreshold`	Maximální hodnota prahu použitého pro získání regionů tvořících strom komponent. *Hodnota z rozsahu 0–1 vyšší jako minimální hodnota nebo hodnota -1 pro automatické zjištění gradientů.*	`double`
`numberOfLevels`	Počet úrovní stromu komponent. *Kladné číslo.*	`int`
`delta`	Počet úrovní zanoření do stromu komponent při získávání korespondujícího otcovského regionu při porovnání stability. *Kladné číslo.*	`int`
`threshold`	Maximální vzdálenost validního pixelu od hrany otce. *Kladné číslo.*	`int`
`minimalMatchedLength`	Minimální poměr validních pixelů pro označení hrany jako stabilní. *Nenulová hodnota rozsahu 0–1.*	`double`
`minimalAreaSize`	Minimální velikost regionů tvořících strom komponent. *Kladné číslo.*	`int`
`minimalBoundaryLength`	Minimální délka hranice regionu tvořícího strom komponent. *Kladné číslo.*	`int`

Tabulka 2 – Parametry metody Linked Edges as Stable Region Boundaries

Mezi parametry metody platí několik vztahů, které by měl mít uživatel na paměti. V nastavení prahů nelze kombinovat automatické zjištění prahu a ruční hodnoty, protože tato konfigurace by mohla

vést k nekonzistentnímu stavu, kdy by byl vyšší práh menší nebo roven nižšímu. Úroveň zanoření při porovnávání stability by měla být v souladu s počtem úrovní stromu, vyšší poměr klade větší nároky na stabilitu. Pokud bude skutečná hranice regionu o zadané minimální velikosti regionu větší, jako zadaná minimální délka hranice, dojde k nalezení mnoha zbytečných regionů bez validních hran.

4.3 Canny edge detector

Vnitřní funkcionalita Canny edge detector je v knihovnách OpenCV již implementována. Vzhledem k tomu, že není důvod tuto implementaci měnit, je možné ji použít přímo. Odlišností od výchozího popisu metody v podkapitole 2.2 je nepřítomnost Gaussova rozostření pro účely preprocesu. Za pomocí příslušné funkcionality dostupných knihoven však lze tuto funkcionalitu doplnit. Tabulka 3 popisuje uživatelsky konfigurovatelné parametry metody. Zvolený formát konfiguračního souboru je stejně jako u LESRB libovolně YAML nebo XML.

Název	Popis	Typ
`lowThreshold`	Nižší práh pro vyřazení příliš slabých hran.	`double`
`highThreshold`	Vyšší práh pro označení silných hran.	`double`
`apertureSize`	Šířka použitého Sobelova operátoru (*liché kladné číslo*).	`int`
`blurKernelSize`	Šířka použitého Gaussova rozostřovacího filtru (*liché kladné číslo*).	`int`

Tabulka 3 – Parametry metody Canny edge detector

4.4 Metodika evaluace

K získání nejpřesnějších výsledků s nejvyšší výpovědní hodnotou je nutné maximalizovat skóre metody při ohodnocení. Za tímto účelem je v souladu s „Best Practice Guidelines“ BSDSB proveden trénink na „train“ (tréninková) a „validation“ (validační) sadách. Vytvořená postup trénování je následující:

1. Je zvolena výchozí konfigurace dle doporučení autorů nebo dle zdrojů metody.
2. Parametry jsou upraveny, za účelem zjištění pozitivního nebo negativního vlivu na výsledek.
3. Je provedena testovací evaluace nad tréninkovou sadou. V případě měkkých hran je použit jen poloviční počet bodů precision-recall křivky (50), pro zvýšení rychlosti evaluace.
4. Výsledky evaluace jsou zpracovány, je zjištěn vliv parametrů. Konfigurace se slibnými výsledky jsou využity v další iteraci od bodu 2. Pokud se již výsledky nevylepšují, pokročí se na bod 5.
5. Omezené množství nejlépe ohodnocených konfigurací je ohodnoceno nad validační sadou. Konfigurace s nejlepším výsledkem je označena jako *natrénovaná konfigurace*. V případě shodných výsledků je vybrána konfigurace náhodně nebo podle jiných pozitivních vlastností (např. nižší nároky na výkon).

Po úspěšném natrénování konfigurace metody je natrénovaná konfigurace použita pro evaluaci nad sadou „test“ (testovací). Evaluace nad touto sadou se provádí jen jednou, výsledky evaluace jsou použity jako výstupní ohodnocení výkonnosti metody. Pro porovnání jsou použity výsledky s optimálním prahem obrazů (OIS).

5 Implementace

Tato kapitola popisuje implementační detaily detektoru hran pomocí LESRB a přidruženého rozhraní pomocí Microsoft Visual Studio 11 Beta, OpenCV 2.3.1 a Parallel Patterns Library v jazyce C++11.

5.1 Testovací rozhraní

Testovací rozhraní je děleno na tři základní bloky. Prvním je rozhraní pro připojování dynamických knihoven a interakci s testovanými metodami deklarované v hlavičkovém souboru `EdgeDetectionLibrary`. Deklarace určuje formát ukazatele na funkci a postup připojení knihovny, zajištěného pomocí třídy `EdgeDetectionLibrary`. Účelem této třídy je reprezentovat připojenou funkci v aplikaci a zajistit bezpečné a unikátní připojení ke knihovně (jedná se tedy o thread-safe singleton). Třída využívá principu počítání referencí pro automatické odpojení knihovny, při destrukci poslední instance třídy. Třída neumožňuje připojení více knihoven v rámci jedné aplikace, protože tato funkcionalita nebyla vyžadována, avšak její implementaci by bylo možné upravit bez změn rozhraní. Po konstrukci třída umožní programátorovi přístup k získanému ukazateli na knihovní funkci.

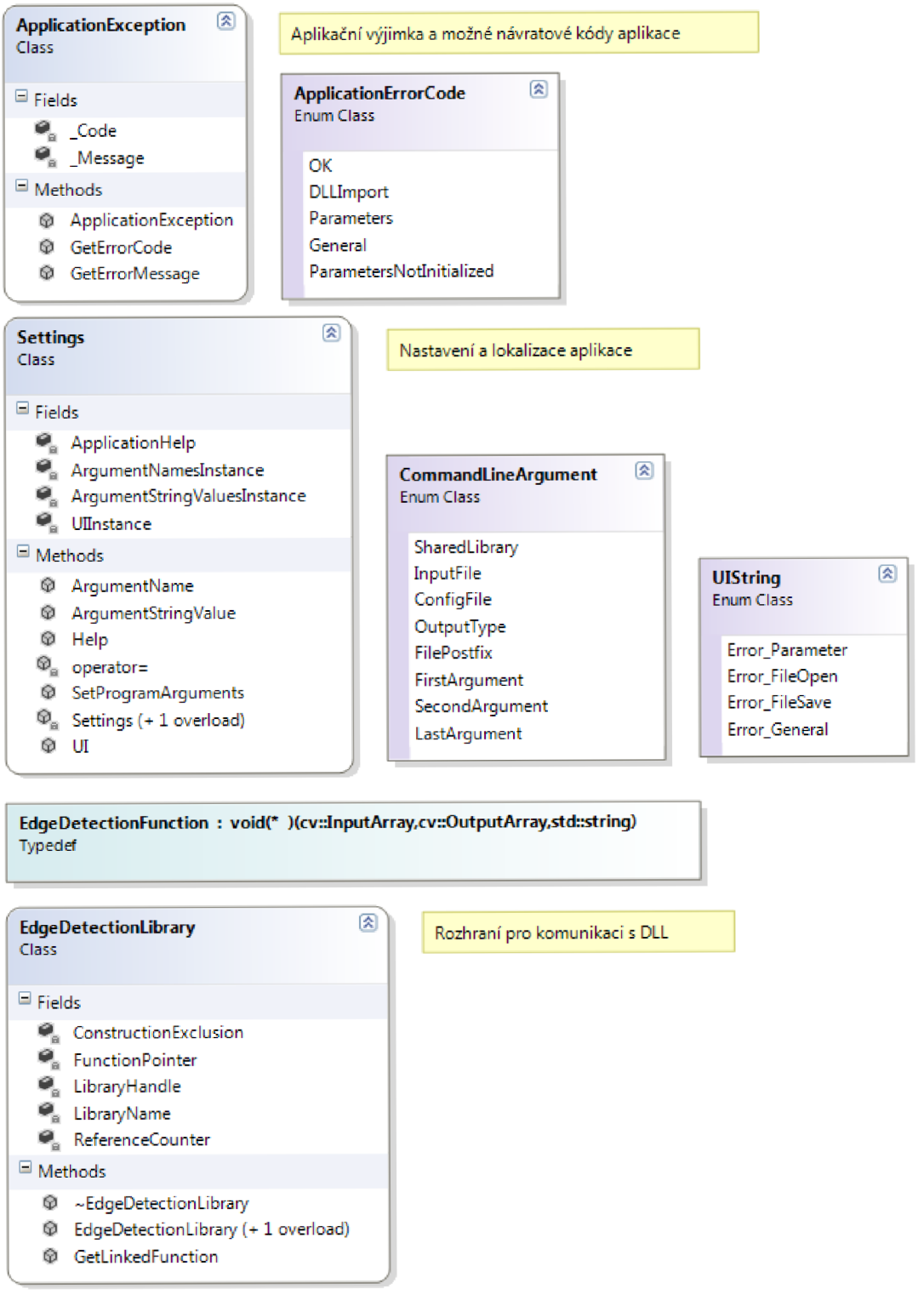
Druhým funkcionálním blokem je lokalizace, nastavení a systém šíření chyb aplikace. Třída `ApplicationException` definuje formát popisu výjimek, předávaných při výskytu výjimečných stavů běhu aplikace, obsahující jeden z definovaných chybových kódů aplikace `ApplicationErrorCode` a zprávu, která je eventuálně použita při interakci s uživatelem (chybové výpisy). V hlavičkovém souboru `Settings`, obsahujícím stejnojmennou třídu, je umístěna funkcionalita pro přístup k aplikačnímu nastavení, lokalizaci a uživatelským parametrům, jejichž rozbor je taktéž poskytován třídou. Tato třída je implementována jako statická (C++11 sice neumožňuje deklarovat třídu jako statickou, avšak při její deklaraci jsou použity stejné principy), čímž umožňuje přístup datům odkudkoliv z rozsahu aplikace. Přidružené výčty slouží jako vnitřní identifikace jednotlivých datových polí, které jsou inicializovány při prvním použití.

Třetím blokem je samotná „implementace“ aplikace – její vstupní bod a podpůrná funkcionalita. Kód vstupního bodu je relativně krátký, proto byl jen minimálně exportován do externích funkcí. Postupnými kroky běhu aplikace jsou inicializace parametrů, připojení knihovní funkce a iterativní zpracování všech vstupních souborů. Aplikaci lze použít více způsoby, postavena je ale na konceptu textového „filtru“, konvertujícím seznam jmen vstupních souborů na seznam výstupních souborů. V případě chyby metody dojde k jejímu ohlášení a pokračování dalším vstupem.

Pro urychlení procesu detekce hran je možné provést její urychlení pomocí paralelního zpracování. Tohoto je dosaženo za použití *Parallel Patterns Library*, což je jediná externí součást aplikace, která není součástí STL C++11 nebo OpenCV. Pro zachování multiplatformosti kódu je použití paralelizace parametricky nastavitelné při překladu aplikace (`/D "PARALLEL"`).

Spouštěcími parametry aplikace byly zvoleny: cesta k připojené knihovně, vstupní soubor (pokud není použit standardní vstup pro vložení dávky souborů), cesta ke konfiguraci metody, typ výstupního souboru a rozlišující postfix výstupního souboru. Výstupní soubory jsou umístěny do stejného umístění jako vstupní.

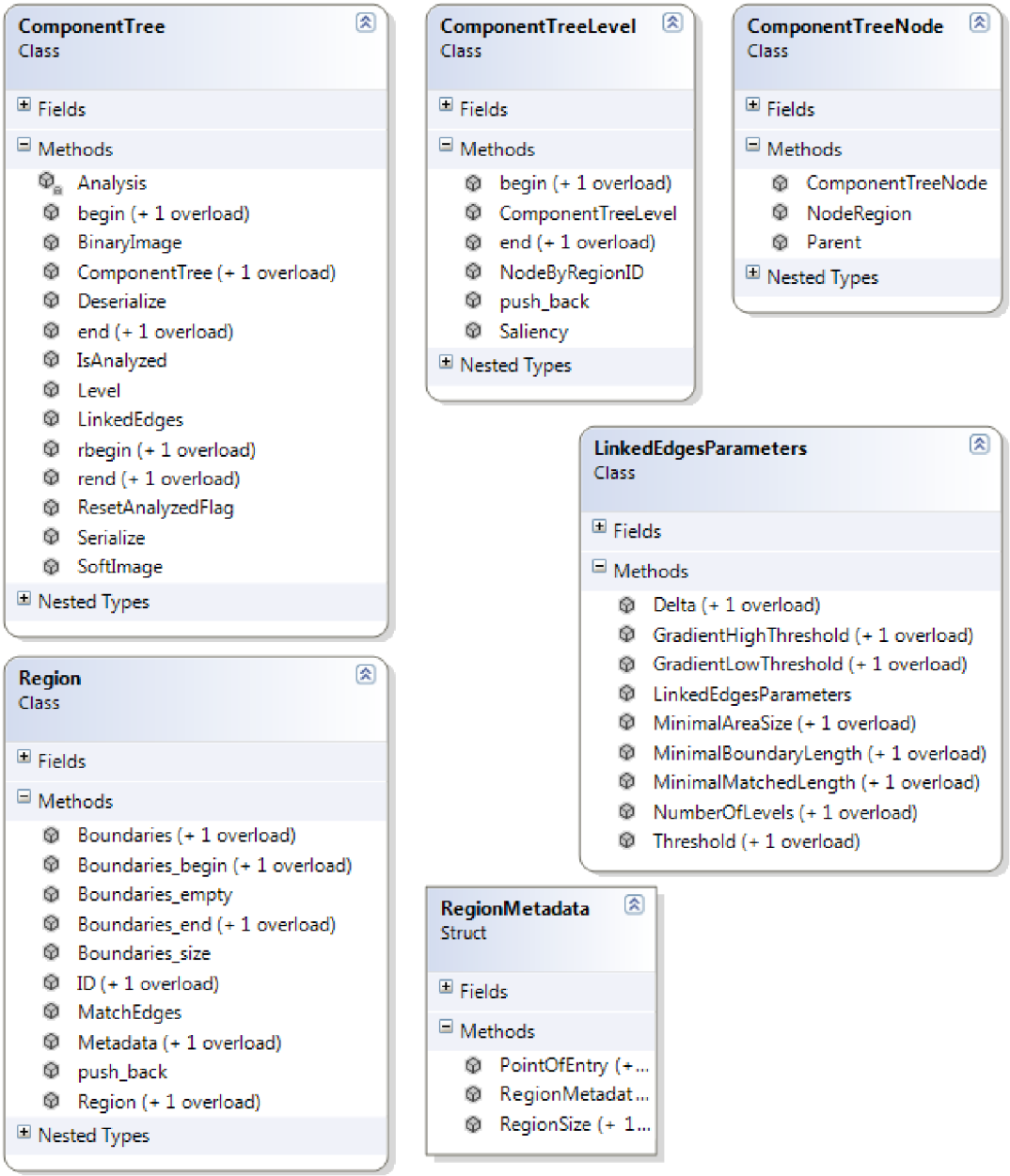
Kromě zmíněného přepínače umožňujícím paralelizaci je použito dalších dvou: `"CASE_INSENSITIVE"`, který potlačí citlivost na velikost písmen parametrů a `"DEBUG"`, který povolí ladící funkcionalitu a výstupy.



Obrázek 22 – Class diagram testovacího rozhraní

5.2 Linked Edges as Stable Region Boundaries

Pro zachování nativního přístupu k detekovaným datům není metoda LESRB implementována jako funkce, nýbrž jako hierarchie tříd, umožňujících přístup k datům stromu komponent v příslušném kontextu. Výchozí třídou pro použití metody je `ComponentTree`, reprezentující celý strom komponent.



Obrázek 23 – Základní hierarchie tříd LESRB

Konstrukce metody přijímá černobílý vstupní obraz a parametry metody, definované instancí `LinkedEdgesParameters`. Uložení parametrů do speciální třídy ulehčuje jejich předávání a validaci. Metody `BinaryImage`, `SoftImage` a `LinkedEdges` slouží jako rozhraní pro získání obrazu tvrdých a měkkých hran, nebo k přístupu k analyzovaným hranám ve vnitřním formátu. Analýza je provedena automaticky, v případě potřeby. Třída umožňuje přímý přístup k jednotlivým

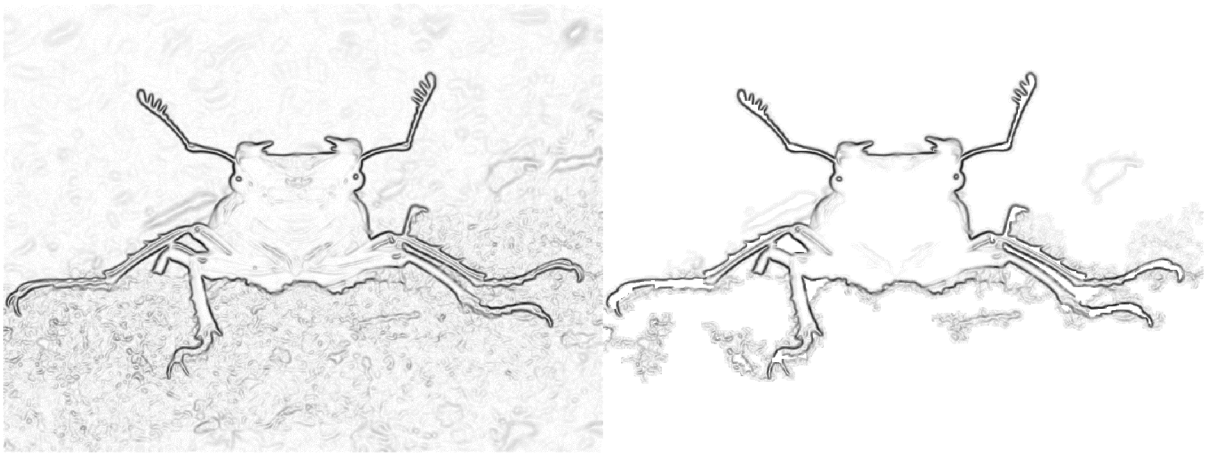
úrovním, pro účely specializovaného použití za pomocí typického rozhraní iterátorů (metody `begin`, `end` a `Level` pro přímý přístup). Úrovně stromu jsou tvořeny instancemi třídy `ComponentTreeLevel`, obsahující seznam uzlů a informaci o jejich význačnosti (`Saliency`). Přístup k uzlům je opět řešen pomocí kombinace iterátorů a přímého přístupu dle ID odpovídajícího regionu uzlu. Samotné uzly jsou tvořené jednoduchou třídou `ComponentTreeNode`, která ukládá informaci o odpovídajícím otcovském uzlu (případně speciální hodnotu pro kořenový uzel bez otce) a příslušející region prahovaného gradientu. Regiony jsou tvořeny třídou `Region`, obsahující hranice regionu (`Boundaries_*` uložené jako řetězce souřadnic pixelů), dále metadata, obsahující plochu regionu a vstupní bod pro potřeby lokalizace regionu (`Metadata`), unikátní identifikaci v rámci úrovně stromu pro účely provázání otcovských uzlů (`ID`) a metodu pro porovnání souladu hran s hranami jiného regionu (`MatchEdges`).

Konstrukce stromu komponent probíhá během konstrukce třídy. Nejprve je získán gradient obrazu pomocí třídy `ImageGradient`, která poskytuje jak samotný gradient, tak jeho komponenty, směrové derivace obrazu. Třída je implementována jako šablona, pro možnost jednoduché změny typu gradientu dle potřeb implementace stromu komponent. Tato vlastnost je pro celou implementaci LESRB specifická a většina z podpůrných funkcí jsou deklarovány jako šablony, což umožňuje snadnější znovupoužití a úpravu. V následujícím kroku dochází k případnému nalezení automatických limitů použitých prahů gradientu podle jeho histogramu. Funkce `AutomaticThresholds` provede konstrukci histogramu s 64 sloupci a získá maximum intervalu z odpovídající hodnoty sloupce, ve kterém je dosaženo kumulativního součtu pixelů hranice 85% všech pixelů. Minimum intervalu je získáno jako 0,4 násobek maxima. Lineárním rozdělením je pak získáno tolik hodnot, kolik je potřeba dle počtu úrovní konstruovaného stromu. Do stromu je zařazena kořenová úroveň, obsahující jeden uzel, reprezentující region o rozměrech obrazu. Pro každou úroveň stromu je pak provedeno nalezení platných regionů. Je provedeno prahování dle příslušné hodnoty získané dříve a mapa prahů je upravena pomocí funkce `FillSmallHoles`, která vylepšuje výsledky nalezených hran vyplněním příliš malých děr v obraze. Tato funkce používá princip analýzy regionů, který se později opakuje. Každý pixel je kontrolován na hledanou hodnotu (hodnotu díry), pokud této hodnoty dosahuje, dochází k označení regionu a tím získání jeho plochy pomocí iterativního *flood fill* algoritmu, používajícího osmiokolí bodu. Označení regionu odlišnou hodnotou zajišťuje, že nebude analyzován vícekrát. V tomto případě jsou příliš malé regiony „obarveny“ znovu, barvou popředí. Tento postup zanechává analyzované regiony vyplněné odlišnou barvou než na počátku, což lze odstranit iterativním přebarvením, ovšem této vlastnosti se často využívá jinak.

Obrázek 24 – Regiony vniklé prahováním (bílé) před (vlevo) a po vyplnění děr a detekci regionů (vpravo)

Po vyplnění malých regionů dochází k detekci regionů tvořících úroveň stromu. Regiony jsou „obarveny“ dle unikátně generovaného ID a během jejich detekce jsou označeny speciální hodnotou i jejich hrany, analýzou okolí pixelů regionu. Body vzniklých hran, označené specifickou hodnotou, jsou pak získány obdobným způsobem, jako jsou nalézány regiony. Regiony a hrany, které splní požadavky parametrů metody, jsou zařazeny do úrovně stromu. ID otcovského regionu je získáno z hodnoty dočasně uschovaného obrazu označených regionů předchozí úrovně, na souřadnicích vstupního bodu regionu (teoreticky libovolných souřadnicích regionu).

Dalším krokem, který již není prováděn během konstrukce, je analýza. K analýze stromu dochází dle potřeby při prvním zavolání některé ze tří metod, popsaných dříve jako uživatelské rozhraní stromu. Výsledkem analýzy je obraz význačných gradientů a seznam validních hran včetně jejich význačnosti. Při analýze jsou příslušné regiony uzlů každé úrovně (krok lze pro účely optimalizace zvýšit) porovnávány s asociovanými otcovskými regiony s vyšší význačností, umístěné o parametricky určený počet úrovní výše. Během porovnávání (metodou `MatchEdges`) jsou zkoumány vzdálenosti pixelů jednotlivých hran otcovského regionu pomocí distance transform obrazu hran „syna“. Porovnáním vznikne maska validity jednotlivých pixelů hran, tvořící jednotlivé validní řetězce – stabilní hrany. Na masku je aplikována metoda `FillSmallHoles`, aby došlo k vylepšení výsledků omezením zanedbatelně oddělených řetězců. Použitá hodnota velikosti maximální vyplněné mezery je 5 pixelů hrany. Na masku je následně aplikováno *run-length encoding*, které slouží ke zjištění délky jednotlivých validních řetězců vytvořených z hranice. Výsledkem porovnávání je seznam validních hran, splňujících opět parametr minimální délky. Pomocí dilatovaného obrazu nalezených hran jsou do obrazu význačných gradientů zapsány hodnoty gradientu do vzdálenosti 3px od nalezených hran, vynásobeny význačností analyzované úrovně. Po ukončení analýzy je saliency image normalizován na rozsah hodnot $< 0{,}0; 1{,}0 >$.



Obrázek 25 – Gradient obrazu (vlevo) a saliency image (vpravo) (invertováno)

V případě, že si uživatel vyžádá jednu z map hran, je provedeno další zpracování. Pro přístup ke kolekci analyzovaných hran ve vnitřním formátu již není potřeba žádného kroku. Pro tvorbu obou typů hran je použita metoda `FindLocalMaxima`, nalézající lokální maxima saliency image po jednotlivých směrech. Odlišností implementace od demonstrační předlohy je nepoužití normalizace hran pomocí morfologického zúžení tak, aby pixely hran měly minimum sousedů. Tato funkcionalita byla shledána nepotřebnou pro účely evaluace, protože boundary benchmark BSDSB provádí normalizaci automaticky.

V případě `BinaryImage` pak dochází k nalezení tvrdých hran pomocí prahování. Prahy jsou opět zvoleny relativně, pomocí funkce `AutomaticThresholds`. Vyšší práh je zjištěn tak, aby 70% saliency image bylo označeno jako pixely pozadí a zbylé jako pixely hran. Nižší práh je určen jako 0,4 násobek vyššího. Po následném nalezení lokálních maxim a prahování je provedena hystereze na stejném principu jako Canny za využití dosavadních nástrojů pro analýzu regionů. Příliš krátké hrany jsou opět odstraněny.

Metoda `SoftImage` používá podobných postupů. Pro omezení délek hran je vytvořena dočasná binární maska, která je poté aplikována na výsledný obraz. Pokusy s výstupy ukázaly, že mnoho nevýznačných hran je konverzí do výstupního formátu znehodnoceno, proto byla implementována ekvalizace obrazu, kdy 20% nejvýznačnějších hran je označeno maximální hodnotou a zbytek je ekvivalentně škálován. Tato metoda není součástí demonstrace a byla implementována za účelem získání precision-recall křivky při evaluaci jednotlivých prahů obrazu.

Metoda opět poskytuje několik přepínačů pro podmíněnou kompilaci zvolené funkcionality. Přepínač `"HIGH_PRECISION"` je použit pro aktivaci náročnější a přesnější analýzy a jeho vypuštěním dojde k rychlejšímu běhu. Podobný účel má přepínač `"EDGES_OUTPUT"`, který kontroluje přístupnost nativního formátu analyzovaných hran. Jeho vypuštěním opět dojde ke zvýšení výkonu. `"MANHATTAN_GRADIENT"` způsobí použití Manhattanské metriky při získání gradientu obrazu, což vede k větší chybovosti a lepšímu výkonu. Přepínač `"DEBUG"` povoluje širší kontrolu a generování ladících výpisů a obrazů jednotlivých mezikroků. Použité sestavení používá pouze přepínače `"HIGH_PRECISION"`.

Vlastní implementovanou funkcionalitou je možnost serializace a deserializace stromu jako soubor formátu XML nebo YAML. Motivací k tomuto kroku bylo zefektivnění jinak náročného ladění pozdějších fází metody načtením předzpracovaných hodnot, avšak funkcionality je možné využít libovolně, včetně procesu konstrukce třídy stromu.

Obrázek 26 – Výsledná mapa měkkých hran

5.3 Canny edge detector

Rozhraní knihovny implementující Canny edge detector následuje stejné postupy jako LESRB. Metoda zajišťuje thread-safe zpracování konfigurace, za použití předchozího nastavení nebo výchozího v případě, že nastavení není možné načíst nebo není poskytnuto. Jádro metody využívá funkce `cv::GaussianBlur` pro aplikaci preprocesu na vstupní obraz, následně provádí detekci pomocí funkce `cv::Canny`.

Podmíněnou kompilací, řízenou přepínači při sestavení, je možné upravit chování metody následovně: `"MANHATTAN_GRADIENT"` způsobí použití Manhattanské metriky při získání gradientu obrazu, což vede k větší chybovosti a lepšímu výkonu a přepínač `"DEBUG"` povoluje generování ladících výpisů a obrazů jednotlivých mezikroků. Použité sestavení nepoužívá žádný z těchto přepínačů.

6 Evaluace úspěšnosti metod

Kapitola popisuje výsledky evaluace zvolených metod pomocí boundary benchmark BSDSB nad přiloženou množinou dat.

6.1 Canny edge detector

Tato podkapitola popisuje trénink a vyhodnocení metody Canny edge detector.

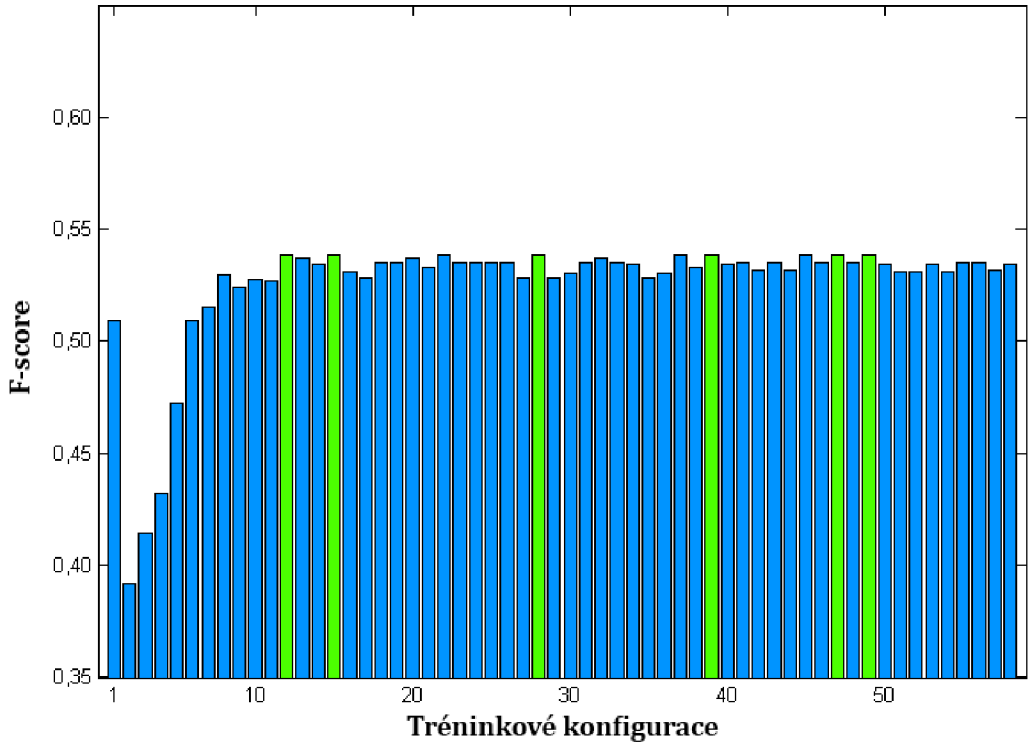
6.1.1 Trénink

Vzhledem k tomu, že doporučení nastavení parametrů metody je omezeno na ideální poměr prahů mezi 2:1 a 3:1 byly inicializační parametry zvoleny autorem, a to následovně:

Název	Hodnota
`lowThreshold`	`60`
`highThreshold`	`180`
`apertureSize`	`3`
`blurKernelSize`	`3`

Tabulka 4 – Výchozí hodnoty tréninku Canny edge detector

Tato konfigurace dosahovala na tréninkové sadě $F = 0{,}51$. Mutace této konfigurace odhalily pozitivní přínos preprocesu při zpracování přirozených obrazů, kdy nejlepších výsledků dosahoval preproces za použití matic šířky 9 a 11. Změna velikosti Sobelova operátoru se na lepších výsledcích neprojevila. Následně bylo provedeno 46 experimentů s různými kombinacemi prahů za použití Gaussovy matice šířky 9 a 11. Po provedení celkem 58 experimentů bylo získáno celkem 43 konfigurací s výsledným F-score mezi hodnotami $0{,}53$ a $0{,}54$.



Obrázek 27 – Výsledky trénovacích konfigurací (nejlepší zeleně)

Z experimentálních konfigurací bylo zvoleno 7 nejlépe ohodnocených (shodně $F = 0{,}538739$) a tyto byly evaluovány nad validační sadou. Evaluace omezila počet nejlépe ohodnocených konfigurací na jednu, s ohodnocením $F = 0{,}53734$.

6.1.2 Testování

Natrénovaná konfigurace metody použitá pro evaluaci nad testovací sadou je následující:

Název	Hodnota
`lowThreshold`	`75`
`highThreshold`	`160`
`apertureSize`	`3`
`blurKernelSize`	`11`

Tabulka 5 – Natrénovaná konfigurace Canny edge detector

Zjištěné hodnoty prahů potvrzují premisu ideálního poměru. Jejich poměr 160:75 (2,13:1) leží v intervalu určeném J. F. Cannym jako ideálním. Dosažené hodnoty F-score jsou následující:

$$F(Precision, Recall) = F(0{,}70; 0{,}46) = 0{,}55$$

6.2 Linked Edges as Stable Region Boundaries

Tato podkapitola popisuje trénink a vyhodnocení metody Linked Edges as Stable Region Boundaries.

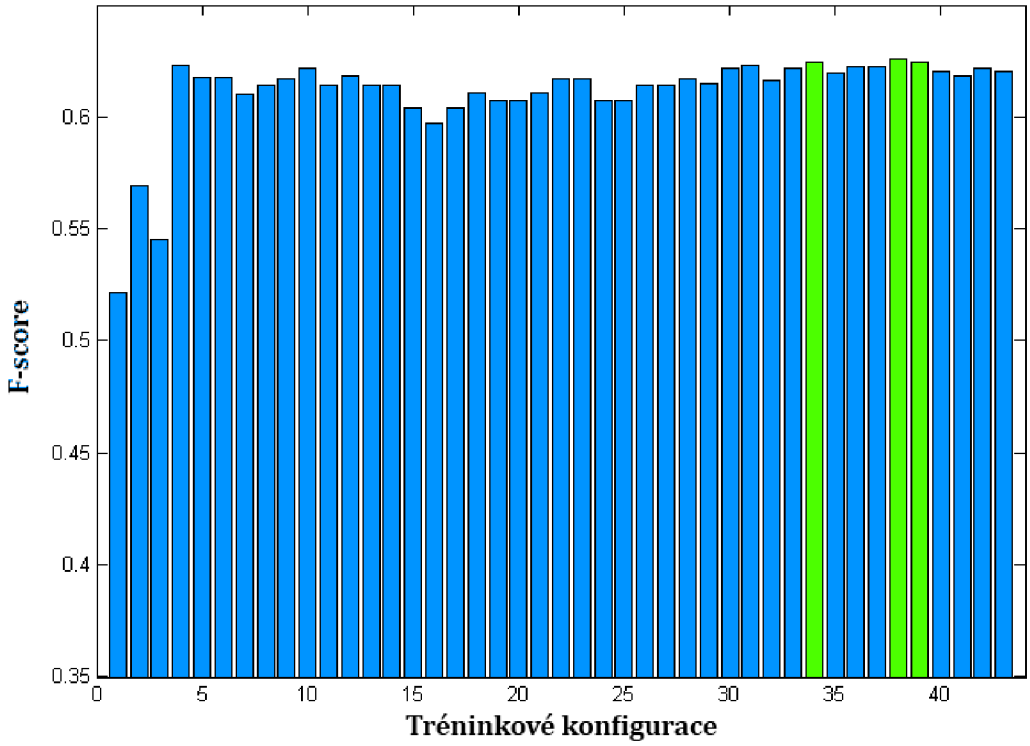
6.2.1 Trénink

Jako inicializační parametry použity hodnoty použité v demonstračním příkladě LESRB:

Název	Hodnota
`gradientLowThreshold`	`-1`
`gradientHighThreshold`	`-1`
`numberOfLevels`	`40`
`delta`	`12`
`threshold`	`2`
`minimalMatchedLength`	`0,3`
`minimalAreaSize`	`1250`
`minimalBoundaryLength`	`70`

Tabulka 6 – Výchozí hodnoty tréninku LESRB

Tato konfigurace dosahovala překvapivě nízkého $OIS\ F = 0{,}52$. Experimenty ukázaly, že tato nízká hodnota byla dosažena z důvodu velmi vysokého nastavení minimální plochy regionů a délky hran (hodnota byla nadsazena i kvůli většímu rozlišení demonstračních obrazů). Experimenty se snižováním plochy přinesly nejlepší hodnoty s nastavením velikosti 50–250 px. Experimentace se zvyšováním počtu úrovní přinesla nulové nebo jen zanedbatelné zlepšení. Vyšší úroveň `delta` přineslo podobné výsledky (hodnoty F-score mezi $0{,}61$ a $0{,}62$). Lepších výsledků bylo dosaženo nastavením vyššího prahu `threshold` a přísnějšího poměru validních pixelů (`minimalMatchedLength`). Nejlepší konfigurace dosahovala ohodnocení na tréninkové sadě $OIS\ F = 0{,}63$.



Obrázek 28 – Výsledky trénovacích konfigurací (nejlepší zeleně)

Z tréninkových konfigurací byly vybrány tři s nejlepším ohodnocením, které byly evaluovány na validační sadě. Výsledky určily nejlepší konfiguraci s ohodnocením $OIS\ F = 0{,}61$.

6.2.2 Testování

Natrénovaná konfigurace metody použitá pro evaluaci nad testovací sadou je následující:

Název	Hodnota
gradientLowThreshold	-1
gradientHighThreshold	-1
numberOfLevels	40
delta	12
threshold	3
minimalMatchedLength	0,3
minimalAreaSize	200
minimalBoundaryLength	28

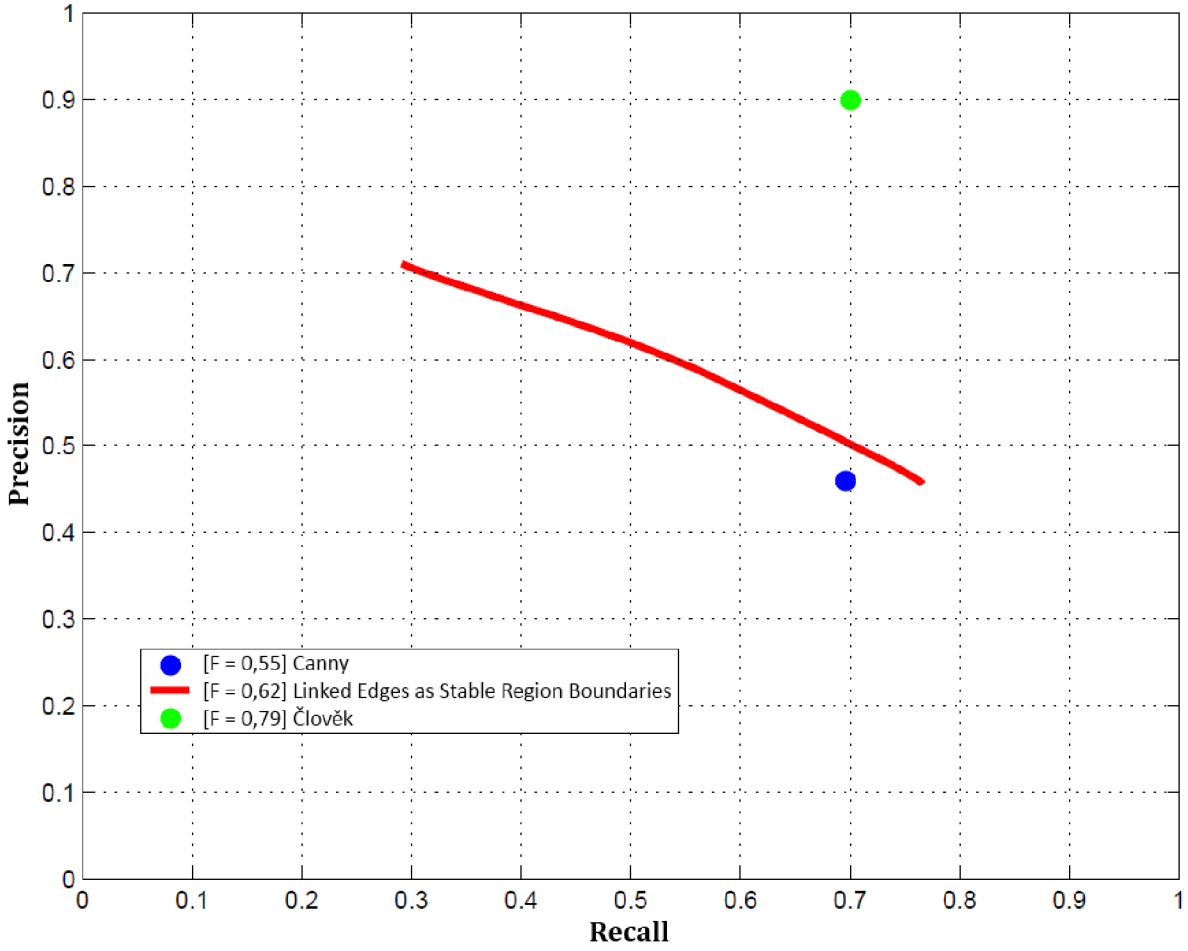
Tabulka 7 – Natrénovaná konfigurace LESRB

Natrénovaná konfigurace se příliš neliší od demonstrační, lepších výsledků bylo dosaženo pouze snížením požadavku na stabilitu hran a zvýšením výskytu hran úpravou minimální velikosti regionů a hran. Dosažené hodnoty F-score jsou následující:

$$OIS\ F(Precision, Recall) = F(0{,}68; 0{,}57) = 0{,}62$$

$$ODS\ F(Precision, Recall) = F(0{,}67; 0{,}52) = 0{,}59$$

Společný práh pro optimal data set scale je $0{,}27$.



Obrázek 29 – Precission-recall křivka testovaných metod

Z precision-recall křivky můžeme dojít k závěru, že metoda je relativně stabilní a nepřibližuje se mezním hodnotám precision ani recall. Zda je toto výhodné chování, bude záviset na použití metody.

7 Závěr

V této práci byly úspěšně popsány, implementovány a vzájemně empiricky porovnány základní a pokročilá metoda detekce hran. V kapitolách 4 a 5 jsou popsány návrh a implementace použité metody Linked Edges as Stable Region Boundaries a proces její evaluace. Následující kapitola prezentovala konkrétní postup tréninku a měření výkonnosti zkoumaných metod.

Výsledky evaluace ukázaly přednosti implementované metody v dosažení přesnějších výsledků, především ve formě méně nenalezených očekávaných hran (vyšší recall). Zkoumáním precission-recall křivky bylo zjištěno, že metoda dosahuje relativně stabilních výsledků, které mohou být zajímavé především při použití jiné metriky, dávající větší důraz na nízký počet false negatives.

Možným pokračováním této práce by mohla být optimalizace detekčního procesu, rozšíření funkcionality (např. aplikace morfologického zužování na mapu výsledných hran) nebo experimentace s parametry, preprocesem a heuristikami, použitými v implementované metodě.

Citovaná literatura

1. **Jeschke, E. R.** GIMP - "Smart" Sharpening. *GIMP - GNU Image Manipulation Program.* [Online] 2002. [Citace: 3. 5 2012.] http://www.gimp.org/tutorials/Smart_Sharpening/.

2. **Donoser, M., Riemenschneider, H. a Bischof, H.** Linked Edges as Stable Region Boundaries. 2010. http://vh.icg.tugraz.at/index.php?content=topics/edgedetect.php.

3. **Canny, J.** A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Listopad 1986, Sv. PAMI-8, 6, stránky 679-698.

4. **Arbelaez, P., a další.** Contour Detection and Hierarchical Image Segmentation. *IEEE TPAMI.* Květen 2011, Sv. 33, 5, stránky 898-916.

5. **California, Computer Vision Group of Berkeley University of.** Contour Detection and Image Segmentation. [Online] [Citace: 18. 1 2012.] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html.

6. **Azernikov, S.** Sweeping solids on manifolds. *Symposium on Solid and Physical Modeling.* 2008, stránky 249-255.

7. **Mai, F., a další.** A hierarchical approach for fast and robust ellipse. *Pattern Recognition.* Srpen 2008.

8. **Nadernejad, Ehsan.** Edge Detection Techniques: Evaluations and Comparisons. *Applied Mathematical Sciences.* 2008, Sv. 2, 31, stránky 1507-1520.

9. **Shneier, Michael.** Road Sign Detection and Recognition. [Online] 2005. http://www.isd.mel.nist.gov/documents/shneier/Road_Sign_Detection.pdf.

10. **Guo, L.** The Edge Detection Operators and Their Application in License Plate Recognition. *Computational Intelligence and Software Engineering (CiSE).* 2010, stránky 1-4.

11. **Hennings, P., Savvides, M. a Vijaya Kumar, B.V.K.** Palmprint Recognition with Multiple Correlation Filters Using Edge Detection for Class-Specific Segmentation. *Automatic Identification Advanced Technologies.* 2007, stránky 214-219.

12. **Sumengen, B. a Manjunath, B. S.** Multi-scale Edge Detection and Image Segmentation. *Proc. European Signal Processing Conference (EUSIPCO).* 2005.

13. **Suna, J., a další.** A multiscale edge detection algorithm based on wavelet domain vector hidden Markov tree model. [Online] http://www.sciencedirect.com/science/article/pii/S0031320303004254.

14. **Nes, P. G.** Fast multi-scale edge-detection in medical ultrasound signals. [Online] 2011. http://arxiv.org/pdf/1107.5186v1.pdf.

15. **Hu, Qimei, He, Xiangjian a Zhou, Jun.** Multi-scale edge detection with bilateral filtering in spiral architecture. *VIP '05 Proceedings of the Pan-Sydney area workshop on Visual information processing.* 2004.

16. Linked Edges as Stable Region Boundaries. *Virtual Habitat Graz.* [Online] [Citace: 2. 12 2011.] http://vh.icg.tugraz.at/index.php?content=topics/edgedetect.php.

Obsah přiloženého CD

- Demonstrační sestavení a zdrojové soubory metod a testovacího rozhraní.
- Elektronická verze této zprávy.
- Elektronická verze plakátu, demonstrujícího cíle a výsledky práce.