Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

Zpracování obrazu pomocí metod teorie grafů

Bakalářská práce

Plzeň, 2013

Tomáš Filandr

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem předloženou bakalářskou práci vypracoval samostatně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne: . . . . . podpis: . . . . . .

Poděkování

Rád bych poděkoval Ing. Tomáši Rybovi za odborné vedení bakalářské práce a mnoho cenných rad a připomínek při její tvorbě.

Anotace

Tématem práce je zpracování obrazu pomocí teorie grafů. Zvolenou metodou pro implementaci jsou inteligentní nůžky. Pro správnou práci této metody je nutné obraz upravit. Nejprve se použije hranový detektor, poté se obraz vhodně naprahuje a nakonec se použije vzdálenostní transformace. Upravený obraz se převede do grafu. Pro segmentaci se používá Dijsktrův algoritmus pro hledání nejkratších cest grafem.

Klíčová slova: segmentace, grafové metody, inteligentní nůžky, hranové detektory, binární dilatace, vzdálenostní transformace

Abstract

The topic of this thesis is image processing using graph theory. The chosen method for implementation is the intelligent scissors. To ensure correct operation of this method it is necessary to modify the image. At first edge detector shall be used, then proper threshold is applied and finally distance transform is used. The modified image is converted into a graph. For segmentation is used Dijkstra's algorithm for finding the shortest paths in graph.

Key words: segmentation, graph algorithms, intelligent scissors, edge detectors, binary dilatation, distance transform

Obsah

Kapitola 1

Úvod

Cílem této práce je shrnutí poznatků z oblasti nižší úrovně zpracování obrazu, především segmentace obrazu, což je činnost, při které jsou automaticky či za pomoci člověka rozlišovány objekty zájmu v obraze. Segmentace obrazu předchází vyšší úrovni zpracování obrazu souhrnně nazývané počítačové vidění. Jelikož je kvalita počítačového vidění úměrně závislá na kvalitě nižší úrovně zpracování obrazu, a hlavně pak segmentaci obrazu, jsou požadavky na tuto nižší úroveň zpracování vysoké. A z tohoto důvodu je tento obor zpracování obrazu velmi rozvinutou a stále se rozvíjející vědní disciplínou.

Zpracování obrazu pomocí metod teorie grafů jsem zvolil z důvodu výhodnosti využití grafů jako reprezentace obrazových dat. Hlavním pozitivem tohoto přístupu je množství a kvalita algoritmů vymyšlená pro práci s grafy, které jsou upraveny do podoby použitelné pro zpracování obrazové informace, zvláště pak segmentaci obrazu.

Jelikož zrak je nejdůležitějším smyslem člověka, alespoň co se množství získané informace týče, je zde velká snaha o jeho napodobení v počítačové technice. Počítačové vidění má velké pole použitelnosti od zabezpečovací techniky přes medicínská data po ovládání počítače gesty. Vzhledem k po-

dobnosti mezi počítačovým a biologickým procesem vidění, je přenášení poznatků mezi těmito dvěma disciplínami klíčové pro rozvoj obou.

Práce se skládá z teoretické části, zde jsou popsány různé metody segmentace obrazu grafové i negrafové. Zvláštní prostor je věnován metodě zvané inteligentní nůžky, jejíž implementace se nachází v praktické části. Následují experimenty porovnávající úspěšnost segmentace inteligentními nůžkami a segmentaci prováděnou ručně člověkem. V závěru práce jsou shrnuty získané poznatky.

Kapitola 2

Teoretická část

2.1 Zpracování digitalizovaného obrazu

Před tím, než vůbec můžeme obraz v počítači zpracovávat, je nutná jeho vhodná reprezentace. Při zpracování obrazu pro potřeby počítačového vidění je vhodné vnímat obraz jako matici čísel resp. vektorů, kde tato čísla odpovídají intenzitě barvy resp. barev jednoho bodu obrazu.

Jelikož práci zpracovávám v programovacím jazyce Python, použil jsem pro práci z maticemi knihovnu numpy, která napodobuje prostředí Matlab. Na obraz ve formě matice lze snadno aplikovat algoritmy pro zpracování obrazu a také se obraz snadno převede do formy grafu, což je klíčové pro mnou zvolenou implementaci algoritmu inteligentní nůžky.

První provedenou úpravou obrazu je převedení obrazu na šedotónový formát, což zajistí značné snížení obsahu dat, ale zůstane zachována rozlišovací schopnost jak segmentačních metod, tak člověka. Ze zmenšení obsahu dat vyplývá rychlejší a jednodušší implementace dalších úprav, prováděných na obrazových datech.

Obraz pořízený z fotoaparátu či videokamery je často nekvalitní a za-

šuměný, proto je většinou nutné předzpracovat ho pomocí filtrů pro odstranění šumu. Rozeznáváme dva základní typy šumu:

- Náhodný šum jinak zvaný nezávislý šum. Ten bývá způsoben snímacím zařízením konkrétně vadnými CCD elementy ve snímacím zařízení. Šum většinou postihuje ojedinělé body obrazu, kde dochází k extrémním hodnotám jasu. Tomuto typu šumu se říká sůl a pepř. Pro odstranění se používají nelineární filtry, mezi něž patří mediánový filtr, či konzervativní vyhlazení.

- Gaussův šum také zvaný závislý šum. Jedná se o šum postihující celoplošně. Jeho výskyt se zvyšuje se zvýšením citlivosti snímače u fotoaparátu, jedná se vlastně o parazitní šum vzniklý z důvodu používání elektrických veličin. Pro jeho odstranění se používají lineární filtry, jejichž zástupcem je Gaussův filtr.

2.1.1 Mediánový filtr

Vhodným filtrem pro odstranění náhodného šumu je mediánový filtr. Pracuje tak, že prochází obraz bod po bodu a změní hodnotu intenzity každého bodu na medián vypočtený z bodů okolních. Za okolní body jsou považovány body ležící v matici 3×3, pokud je střed matice umístěn na zpracovávaném bodě. Medián je taková hodnota ze souboru čísel, pro niž platí, že 50% čísel leží nad mediánem, či je mu rovna a 50% leží pod mediánem, či je mu rovna.

2.1.2 Konzervativní vyhlazení

Jedná se o rychlou metodu vhodnou pro odstranění šumu typu sůl a pepř, neboli šumu způsobujícímu ojediněle se vyskytující extrémně vysoké, či nízké hodnoty jasu v obrazu. Filtr stejně jako mediánový filtr pracuje s okolím

bodu. V okolí zkoumaného bodu vždy nalezne minimální a maximální hodnotu intenzity. Pokud hodnota intenzity zkoumaného bodu leží mezi tímto minimem a maximem, zůstane bod nezměněn. Pokud však intenzita neleží v tomto rozmezí, změní se na hodnotu minima resp. maxima, podle toho, jestli je menší než minimum resp. větší než maximum.

2.1.3 Gaussův filtr

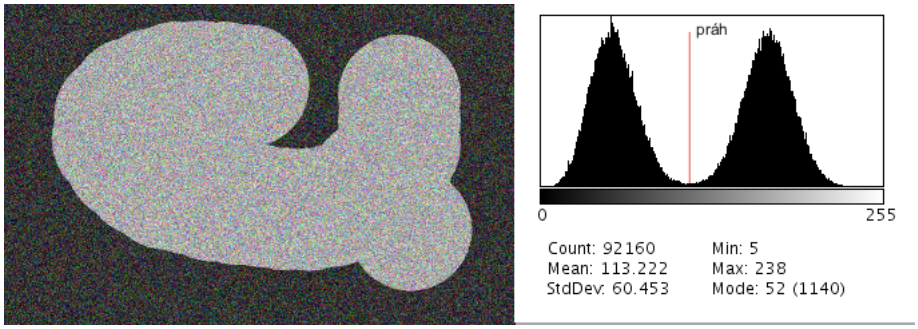
Gaussův filtr je zástupcem lineárních fitrů vhodných pro odstraňování závislého šumu. Gaussův fitr pracuje na základě konvoluce s maskou. Jedná se o matematickou metodu, při níž se výsledná hodnota jasu zkoumaného bodu vypočte jako suma intenzit okolních bodů přenásobených maskou. Maskou je myšlena matice. V případě Gaussova filtru se používá matice s elementy vypočítanými Gaussovou funkcí:

$$G(x, y) = \frac{1}{2\pi\sigma} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

Bohužel použití tohoto filtru vede k rozmazání obrazu, což zhoršuje následnou detekci hran.

2.2 Segmentace obrazu

Segmentace obrazu je skupina metod, sloužící k rozdělení obrazu na logické části, na oblasti se společnými vlastnostmi. Mezi tyto vlastnosti patří barva a vzor. Toto rozdělení vede k analýze obsahu zpracovaných obrazových dat. Jedná se o poslední krok prováděný v nižší úrovni zpracování obrazových dat, po němž již nastupují metody počítačového vidění, které na základě segmentovaného obrazu rozpoznávají, co se vlastně na snímku nachází. Z toho důvodu je kvalita segmentace velmi důležitá pro další práci s obrazovými daty. Mezi segmentační metody patří:



Obrázek 2.1: Segmentace prahováním. Převzato z: http://commons. wikimedia.org/wiki/File:Tr_dobry.png

2.2.1 Segmentace prahováním

Jedná se o základní a jednoduchou metodu segmentace. Z histogramu obrazu se určí vhodný práh jasu a to tak, aby všechny hodnoty jasu nižší než práh odpovídaly pozadí obrazu a hodnoty jasu vyšší než zvolený práh zase popředí obrazu. V případě více objektů s různou hodnotou jasu lze zvolit prahů více. Problém nastává v případě, že se rozložení jasu popření a pozadí výrazně překrývá, potom je tato metoda prakticky nepoužitelná. Na obrázku č. 2.1 je vidět obraz s histogramem a zvoleným prahem.

2.2.2 Segmentace spojováním oblastí

Tato metoda segmentace může být použita jako interaktivní, ale nemusí. To záleží na tom, jestli ponecháme volbu startovacích bodů segmentace uživateli, či tyto body stanovíme, například náhodně. Zvolené body tvoří na počátku různé segmenty. Tyto segmenty se zvyšují nabalováním dalších bodů v okolí na základě kritéria podobnosti, například mají-li podobný jas. Během tohoto algoritmu vzniknou v obraze oddělené segmenty.

2.2.3 Segmentace štěpením oblastí

Průběh této metody je do jisté míry opakem segmentace spojováním oblastí. Při startu metody se nejdříve označí obraz jako jeden segment. Tento segment je následně rozdělen na čtyři stejně velké segmenty, například čtvercové. Takto vzniklý segment je testován na homogenitu. Jde o to, jestli body v segmentu mají shodný jas. Je-li takto vniklý segment homogenní, již se dále nedělí. Pokud homogenní není, rozdělí se opět na čtyři shodné oblasti, které se znovu otestují. Algoritmus takto pokračuje do doby, něž je každý segment v obraze homogenní.

2.2.4 Segmentace rozvodí

Segmentace rozvodí, či povodí nebo v originále watershed je metoda vycházející z geografie. Obraz je chápán jako terén, jehož nadmořskou výšku v jednotlivých bodech určuje hodnota jasu. Algoritmus začíná tím, že najde nejnižší místa v obraze a od těchto míst začne zaplavovat obraz. V místech, kde by mohlo dojít ke spojení dvou oblastí, je vystavěna hráz, která tomu zabrání. Takto algoritmus zaplavuje stále vyšší místa, až dojde do maxima obrazu. Výsledkem je obraz rozdělený do regionů zaplavených vodou oddělených hrázemi. Nevýhodou této metody je, že produkuje zbytečně velké množství segmentů, a ty je tedy nutno spojit.

2.3 Grafové segmentační metody

Segmentace využívající teorii grafů je výhodná ať už z pohledu snadné a přehledné reprezentace, tak z pohledu algoritmů a metod, kterých teorie grafů má nepřeberné množství. Při segmentaci obrazu reprezentovaného grafem se skoro vždy jedná o snahu rozdělit graf na podgrafy tak, aby každý podgraf

odpovídal jednomu objektu zachycenému na obraze. Dále v této kapitole jsou popsány nejznámější používané metody.

2.3.1 Segmentace minimální kostrou grafu

Kostrou grafu se rozumí graf obsahující všechny vrcholy původního grafu. Tyto vrcholy musí být navzájem spojené hranami a v grafu se nesmí nacházet žádná kružnice. To znamená, že mezi libovolnými dvěma vrcholy existuje vždy jen jedna cesta. Kostra grafu se dá také nazývat stromem.

Minimální kostra grafu angl. minimal spanning tree (MST) je kostra ohodnoceného grafu, jejíž součet vah hran je nejnižší ze všech možných koster. Pro vytvoření MST se využívá Kruskalův algoritmus [6].

Kruskalův algoritmus postupuje tak, že vezme všechny hrany grafu a vytvoří z nich samostatné stromy. Poté ze seznamu hran najde nejlevnější hranu a pokud tato hrana spojuje dva různé stromy, použije ji a spojené stromy změní v jeden. Pokud hrana nespojuje dva různé stromy, vyhodí ji. Tímto postupem bychom nakonec dostali MST, ale to není v segmentaci žádaný výsledek.

V roce 2004 představil Felzenszwalb algoritmus [7] vycházející z Kruskalova algoritmu. Jedinou změnou oproti Kruskalovu algoritmu je podmínka přidaná při připojování dvou stromů. A to taková, že ke spojení dvou stromů je navíc zapotřebí aby si jejich body byly podobné. Za podobnost lze brát například podobnou hodnotu průměrného jasu bodů v daném stromě. Aplikací tohoto algoritmu vzniknou oddělené stromy reprezentující jednotlivé segmenty obrazu. Algoritmus má lineárnělogaritmickou náročnost. Ukázky segmentace obrázek č. 2.2

Obrázek 2.2: MST segmentace. Převzato z: http://origin-ars.els-cdn.com/content/ image/1-s2.0-S0031320312004219-gr2.jpg

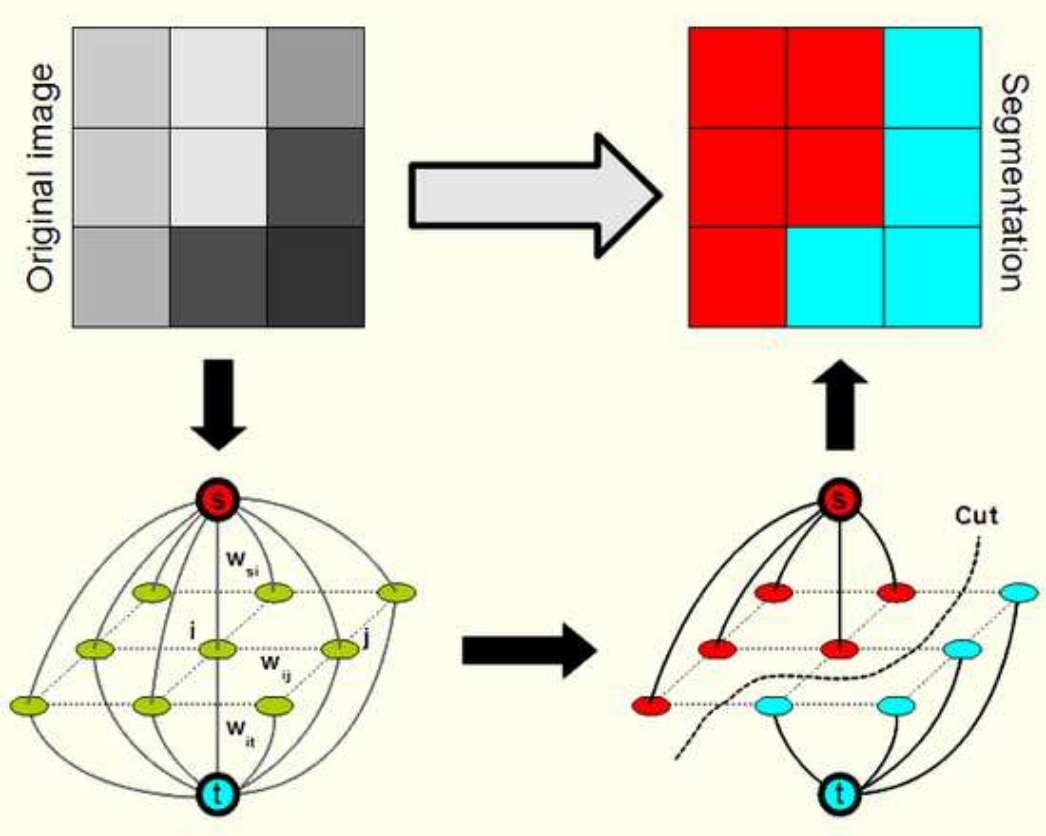
2.3.2 Segmentace minimálním řezem grafu

Segmentace minimálním řezem patří do skupiny interaktivních metod segmentace, jejichž běh může být ovlivněn zásahem člověka. V případě této metody člověk určí, co na obraze je popředí, čili předmět zájmu a která část patří do pozadí obrazu. Algoritmus tyto body využije při konstrukci sítě. A to tak, že z bodů popředí udělá vrcholy zdroje a z bodů pozadí vrcholy spotřebiče nebo stoku.

Takto vzniklý zdroj a spotřebič se napojí na všechny ostatní vrcholy grafu. Cena takto vzniklých hran se určí podle rozdílu jasu mezi spojenými body a také je zde zohledněna vzdálenost mezi zdrojem popř. spotřebičem a vrcholem grafu.

Úkolem algoritmu je poté nalézt řez grafem tak, že zdroj a spotřebič budou v rozdílných podgrafech a řez bude proveden hranami s nejmenším součtem vah. K tomuto účelu lze použít Fordův-Fulkersonův algoritmus [8], který v grafu nalezne maximální tok a provede minimální řez, čímž rozdělí graf na dva podgrafy.

Ideálně tyto podgrafy vymezují popředí a pozadí obrazu. Úspěšnost metody segmentace minimálním řezem grafu lze zvýšit tím, že uživatel vybere



Obrázek 2.3: Segmentace minimálním řezem grafu. Převzato z: http://www.ondrej-danek.net/images/vyzkum/graph.png

další body a určí u nich, zda patří do popředí či pozadí obrazu. Algoritmus je poté možno spustit znovu. Ukázka práce segmentace minimální řezem grafu na obrázku č. 2.3

2.3.3 Segmentace náhodnou procházkou

Jedná se o nejnovější ze zde představených metod. Metoda náhodnou procházkou angl. random walker patří do rodiny interaktivních metod segmentace. Tato metoda využívá matematickou myšlenku náhodné procházky. Jedná se o posloupnost náhodných kroků, kdy každý další krok je proveden náhod-

ným směrem. Tato myšlenka je také někdy nazývána chůzí opilce.

Segmentační metoda tedy začíná zásahem uživatele, který označí jednotlivé objekty v obraze, kterých může být více. Poté se pro každý bod v obraze spočte pravděpodobnost, že bychom při náhodné procházce z tohoto bodu došli do označeného bodu. Tato pravděpodobnost se spočte pro každý označený objekt. Příslušnost zkoumaného bodu je dána nejvyšší z pravděpodobností doražení do některého z označených bodů vzhledem k ostatním označeným bodům.

Díky tomu, že algoritmus pracuje s ohodnoceným grafem, je do výsledku zahrnuta i váha hran a tak random walker překročí výraznou hranu s velmi malou pravděpodobností.

Výhodou této metody je snadná implementace a rychlý výpočet jednotlivých pravděpodobností. Mezi další výhody patří i to, že při vhodně zvolených počátečních bodech je algoritmus schopen rozdělit i objekty u nichž není hrana čitelná změnou jasu.

2.4 Segmentace inteligentními nůžkami

Jedná se o metodu, kterou jsem si zvolil k implementaci. Segmentační metoda inteligentní nůžky nebo magické laso, jak je pojmenována v grafickém editoru Photoshop, patří do skupiny interaktivních metod.

Metoda inteligentních nůžek se poprvé objevila v publikaci [5]. Inteligentní nůžky vznikly jako nástroj pro zrychlení segmentace prováděné člověkem.

Pro samotnou segmentaci se používají algoritmy pro vyhledávání nejkratší cesty grafem. Tento graf je však nejdříve nutno sestavit, jelikož pro správnou funkci nestačí převést obraz na graf jako u ostatních metod.

Obraz musí být upraven do podoby, ve které z něj bude patrná vzdálenost od hran objektů. Toho se docílí několika kroky úprav.

Prvním krokem, pokud již máme obraz uložen ve formě matice a převeden do šedotónové stupnice, je detekce hran. Detekce hran vyhledává v obraze místa, kde dochází k výrazné změně intenzity jasu. Hranu si lze také představit jako přechod mezi nízkofrekvenční a vysokofrekvenční oblasti signálu. Pro detekci hran se používají hranové detektory. Různé druhy detektorů pracují z obrazem jinak. Detektory mohou být založeny na:

- první derivaci,
- druhé derivaci,
- nebo nemusí pracovat s derivacemi vůbec.

Pro účely inteligentních nůžek jsem použil detektory pracující s první derivací, tedy Sobelův, Cannyho a Prewittové hranový detektor. Všechny tři postupují v podstatě stejně. Zderivují obraz a provedou konvoluci pomocí masky. Masek je celkem osm aby reagovaly na všechny směry hran. Pro počítaný bod se tedy použijí všechny masky a ponechá se ta s největším výsledkem. Masky jsou definované různě pro každou metodu. Dále uvedu masky pro jednotlivé metody. Nebudu je uvádět všechny, protože vznikají pouhým pootočením.

- Sobelův detektor:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix} \tag{2.2}$$

- Prewittové detektor:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} \tag{2.3}$$

Cannyho detektor využívá při své práci jádro Sobelova detektoru. Vypočtením konvoluce s jádrem tedy vznikne obraz derivací, ve kterém mají hrany nejvyšší čísla.

Dalším krokem při úpravě obrazu je prahování pro odstranění nepodstatných hran. Jelikož hranový detektor rozezná jakoukoliv změnu jasu v obraze jako hranu, je nutno zvolit vhodný práh, kterým se odfiltrují slabé hrany. Tento práh bohužel nelze spočítat a musí se zjistit experimentálně.

Po odfitrovaní nepodstatných hran se může stát, že prahování způsobilo diskontinuity i v hranách, které považujeme za podstatné. Tento problém je řešitelný metodou zvanou binární dilatace. Binární dilatace pracuje tak, že prozkoumá postupně každý bod obrazu, a když narazí na bod označený jako hrana, položí na něj střed strukturního elementu. Strukturní element je čtvercová matice s lichým rozměrem hrany, obsahující pouze jedničky a nuly. Algoritmus binární dilatace po přiložení strukturního elementu středem na bod hrany určí i všechna místa, která strukturní element označí jedničkou jako hrany. Tímto postupem dojde k vyplnění případných nespojitostí v hranách. Na obrázcích č. 3.7 a 3.8 je vidět rozdíl mezi hranovým obrazem bez binární dilatace a hranovým obrazem s binární dilatací

V momentě, kdy již máme obraz hran, se pro další postup použije vzdálenostní transformace. Jedná se o metodu, která hranám přiřadí číslo nula a všem ostatním bodům přiřadí číslo odpovídající vzdálenosti k nejbližší hraně. Tímto postupem se zařídí vhodný obraz pro algoritmus hledající nejkratší

cesty, jelikož se tento algoritmus bude držet převážně hran a prostor mezi nimi bude překlenovat tak, aby byl co nejblíže hraně.

Obraz je tedy připraven. Dále se převede na graf, který umožní použití algoritmu pro hledání nejkratších cest. Za tento algoritmus jsem zvolil Dijkstrův algoritmus. Jeho pseudokód je uvedený dále:

```
1 funkce Dijkstra(Graf, zdroj):
2    pro každý uzel v z Graf:
3       d[v] := infinity ;   // nekonečná vzdálenost
                                ze zdroje do v
4       p[v] := undefined ;  // předchozí uzel na ideální
                                cestě ze zdroje
6    d[z] := 0 ;             // vzdálenost ze zdroje do zdroje
7    N := Graf ;             // seznam všech nenavštívených
                                vrcholů
8    dokud N není prázdné:   // hlavní smyčka algoritmu
9       u := uzel z N s nejmenší d[u] ;   // při prvním průchodu
                                             se jedná o zdroj
10      smazat u z N ;
11      pokud d[u] = infinity:
12         break ;           // nastane v případě kdy jsou všechny
                                uzly z N nedostupné z uzlu u
13      pro každé v které je sousedem u:   // kde v se stále
                                              nachází v N
14         alt := d[u] + d(u, v) ;
15         pokud alt < d[v]:   // testuje zda je cesta ze zdroje
                                   do u + cesta z u do v levnější
                                   než cesta ze zdroje do v
```

```
16            d[v] := alt ;   // nastaví cenu cesty ze zdroje
                                 do v na tu levnější přes u
17            p[v] := u ;    // předchozí uzel na ideální cestě
                                ze zdroje do v je u
```

Takto zadaný algoritmu hledá nejlevnější cestu ze zdroje do všech ostatních propojených uzlů, pro účely nalezení nejkratší cesty mezi dvěma body se přidá zastavovací podmínka na řádku devět pokud je u cílovým uzlem. Cesta se poté určí zpětnou iterací za pomoci zásobníku:

```
1 Z := prázdný zásobník       // zde bude uložena cesta
2 u := cíl
3 dokud p[u] je definováno:
4    vlož u na začátek Z
5    u := p[u]                // takto se vrací od cíle ke zdroji
```

Teď již je v zásobníku Z uložena nejkratší cesta ze zdroje do cíle. Tato cesta je použita jako hranice mezi segmenty v obraze.

Kapitola 3

Praktická část

3.1 Dílčí úlohy

V této části práce představím jednotlivé úkony prováděné při úpravě obrazu a posléze grafu pro použití segmentace inteligentními nůžkami.

3.1.1 Načtení obrazu

Pro načtení obrazu do prostředí programovacího jazyka python jsem použil knihovnu *OpenCV*[1], která umožňuje rychlé načtení obrazových formátů jako jsou Windows bitmapy, JPEG, PNG, TIFF a další, do formátu dvourozměrných polí, která jsou pro další zpracování výhodná. Tato knihovna také umožňuje okamžité převedení barevného obrazu do šedotónového, což usnadňuje další práci.

[1] http://docs.opencv.org/

3.1.2 Detekce hran

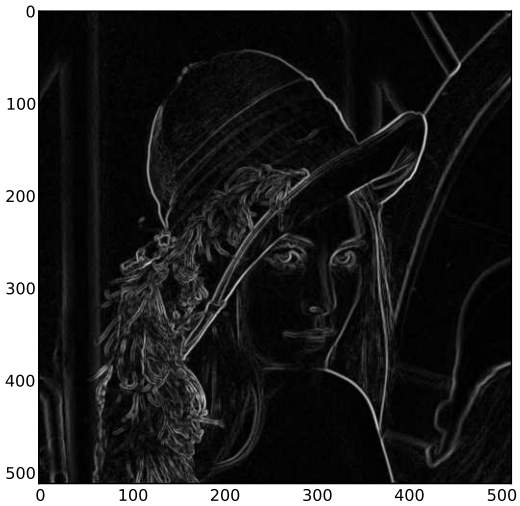
Dalším krokem při zpracování obrazové informace je detekce hran.Pro hranové detektory byla použita knihovna *scikit-image*[2], konkrétně její modul *filter*, který obsahuje implementace různých hranových detektorů. V této práci jsem testoval tři různé detektory hran a to Sobelův detektor, Prewittové detektor a Cannyho detektor. Jediným rozdílem mezi Sobelovým a Prewittové detektorem je konvoluční jádro.

Cannyho detektor [9] je o něco komplexnější něž předchozí dva. Pro detekci gradientů používá stejný postup jako předchozí detektory, dokonce nejčastěji využívá i konvoluční jádro ze Sobelova detektoru. Avšak před samotnou detekcí nejprve upravuje obraz Gaussovým filtrem pro redukci šumu. Poté použije Sobelův detektor pro detekci hran. V dalším kroku odfiltruje nepodstatné hrany a to tak, že má nastaveny dva prahy. Pokud hrany projdou tím přísnějším, jsou automaticky označeny jako důležité. Pokud neprojdou ani méně přísným prahem, jsou automaticky odfiltrovány a pokud se nacházejí mezi oběma prahy, jsou označeny jako důležité, a to v momentě kdy sousedí s bodem označeným jako hrana, jinak jsou jsou odfiltrovány. Na obrázcích (č. 3.1 - 3.4) jsou ukázky práce jednotlivých hranových detektorů.

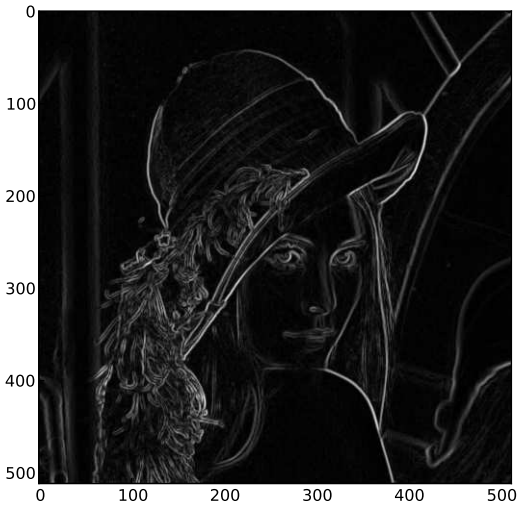
Pro další práci jsem použil Sobelův detektor z důvodu snadnějšího nastavení a uspokojivých výsledků. Bylo možno použít i Prewittové detektor a výsledky by byly k nerozeznání od Sobelova detektoru. Cannyho detektor má zbytečně komplexní nastavení, které se musí stejně jako u všech zbylých detektorů nastavovat experimentálně, a proto je nastavení Cannyho detektoru nevhodné pro další použití v této práci.

[2]http://scikit-image.org/

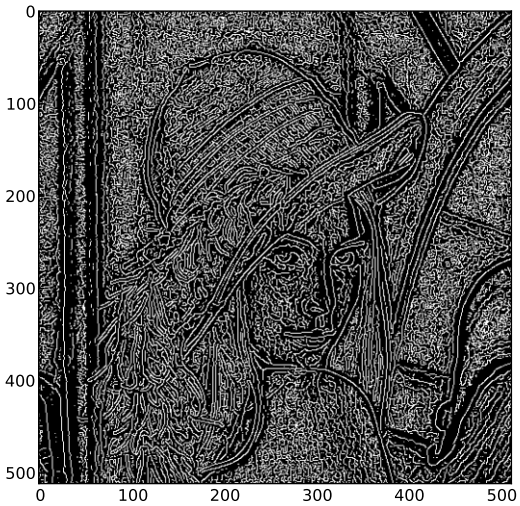
Obrázek 3.1: Obraz převedený do šedotónové stupnice



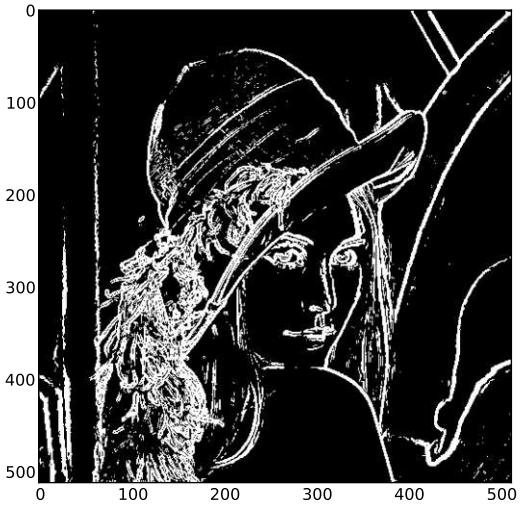
Obrázek 3.2: Hrany nalezené Sobelovým hranovým detektorem



Obrázek 3.3: Hrany nalezené Prewittové hranovým detektorem



Obrázek 3.4: Hrany nalezené Cannyho hranovým detektorem



Obrázek 3.5: Hrany nalezené Sobelovým hranovým detektorem, práh 8%

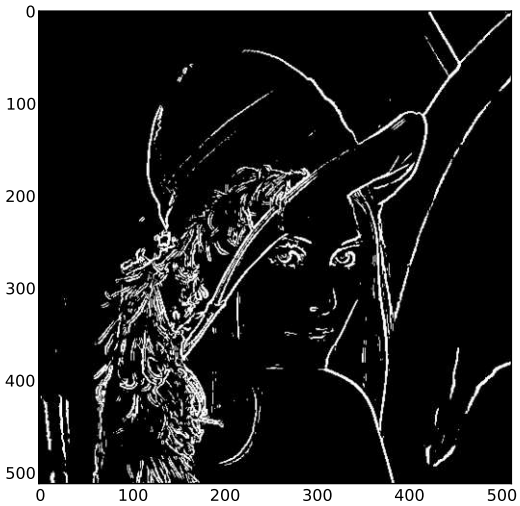
3.1.3 Prahování

Při použití Sobelova detektoru je dalším krokem nastavení vhodného prahu pro odfiltrování nepodstatných hran v obraze. Vhodná hodnota prahu je různá pro každý obraz, a je nutno ji zjistit experimentálně. Během testování jsem zjistil, že tato hodnota se pohybuje v rozmezí od 8% do 20%. Na obrázcích (č. 3.5 - 3.7) je ukázán vliv hodnoty prahu na hranový obrázek.

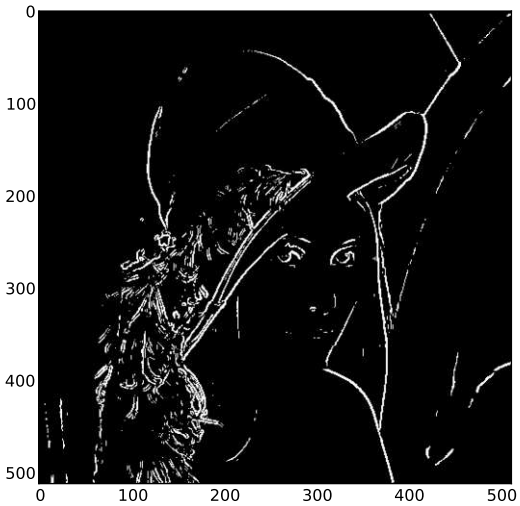
3.1.4 Binární dilatace

Dalším možným krokem ve zpracování hranového obrazu je binární dilatace. Její implementace se nachází taktéž v knihovně *scikit-image* v modulu *morphology*. Binární dilatace se používá pro zvýraznění hran v hranovém obraze. Součástí binární dilatace je strukturní element, což je čtvercová matice s lichou velikostí hrany obsahující pouze jedničky a nuly.

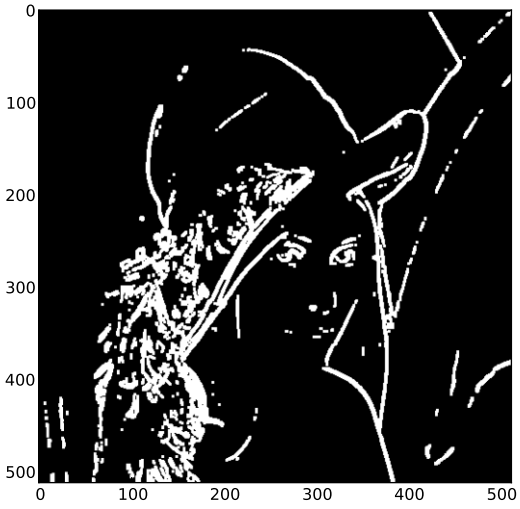
Tento strukturní element se přikládá středem na každou pozici v matici a



Obrázek 3.6: Hrany nalezené Sobelovým hranovým detektorem, práh 15%



Obrázek 3.7: Hrany nalezené Sobelovým hranovým detektorem, práh 20%

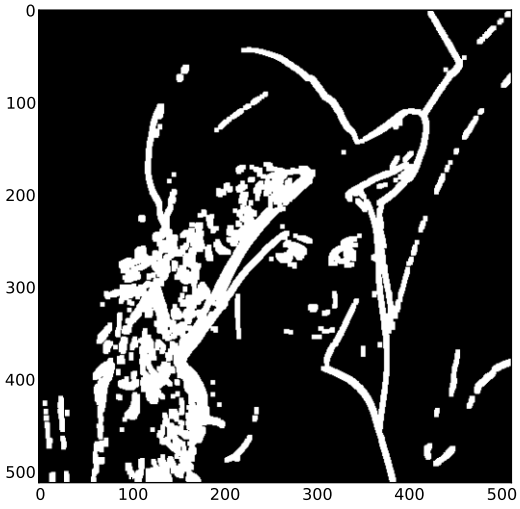


Obrázek 3.8: Binární dilatace čtvercovým strukturním elementem o hraně 3

pokud se přiloží na pozici označující hranu, rozšíří se tato hrana i na ostatní místa ve strukturním elementu označená právě jedničkou. Takto se sice dosáhne spojení hran, které jsou neúplné, ale také dojde k jejich nežádoucímu rozšíření do tloušťky, což v mnohých případech vede ke spojení hran, které jsou blízko sebe i když spojeny být nemají. Toto spojení a rozšíření hran vede k nepřesnostem při pozdějším použití inteligentních nůžek, a proto jsem tuto úpravu v práci otestoval, ale nepoužil k dalšímu zpracování. Ukázka binární dilatace hranového obrazu při různé velikosti strukturního elementu je na obrázcích č. 3.8 a 3.9.

3.1.5 Vzdálenostní transformace

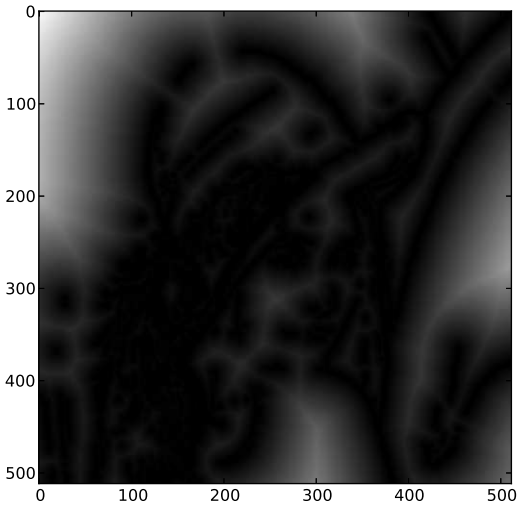
Posledním krokem při úpravě hranového obrazu je vzdálenostní transformace. Pro její implementaci používám metodu *medial axis* v modulu *morphology* knihovny *scikit-image*, jejíž vedlejším produktem je matice vzdálenostní



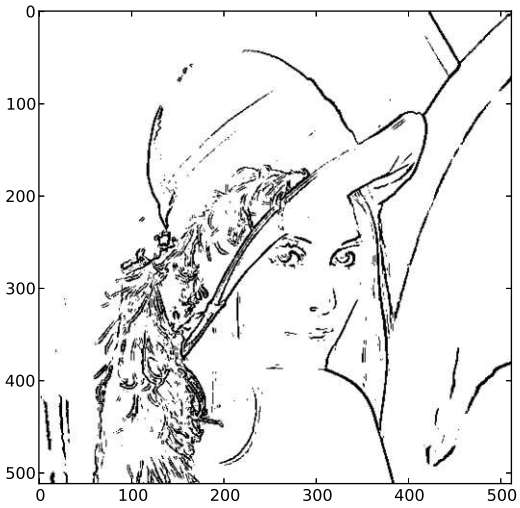
Obrázek 3.9: Binární dilatace čtvercovým strukturním elementem o hraně 5

transformace, kde každý bod má nastavenu hodnotu odpovídající jeho vzdálenosti od hrany (viz obr. 3.10). Tato matice mi umožní vytvořit graf, kde uzly odpovídající hraně budou mít nejmenší hodnotu a čím dále se bude uzel nacházet od hrany, tím jeho hodnota poroste.

Druhou možností je nepoužívat vzdálenostní transformaci, ale použít obrázek hran jako takový, kde se hraně nastaví malá hodnota v mém případě nula a všem ostatním bodům, které nejsou hranou, se nastaví hodnota velká v mém případě 100 (viz obr. 3.11). Tento obrázek je také možno použít pro konstrukci grafu. Avšak jak je vidět na obrázku č. 3.12, použití tohoto přístupu způsobuje problémy v konvexních oblastech obrazu, pokud při zadání počáteční pozice netrefíme přímo hranu.



Obrázek 3.10: Vzdálenostní transformace použitá na hranový obraz



Obrázek 3.11: Inverzní hranový obraz

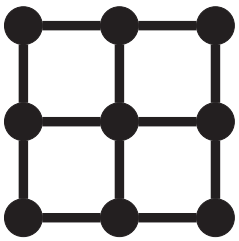
Obrázek 3.12: Vlevo: obrázek segmentovaný s použitím inverzních hran, vpravo: použití vzdálenostní transformace

3.1.6 Graf

Pro reprezentaci grafu v jazyce python jsem použil knihovnu *NetworkX*[3], která má implementovány všechny potřebné metody pro práci s grafem i další metody, například pro výpočet nejkratší cesty, kterou jsem také použil. Graf jsem vytvořil z obrazu vzdálenostní transformace. Každý bod obrazu, resp. každý bod matice jsem převedl na uzel grafu, tomuto uzlu je také uložena hodnota odpovídající vzdálenosti od hrany. Uzly grafu se dále spojí neorientovanými hranami podle tzv. čtyřokolí (viz obr. 3.13), kdy je každý uzel napojen na uzly nacházející se nad, pod, vlevo a vpravo od napojovaného uzlu.

Tyto hrany byly dále ohodnoceny buď lineární funkcí, kde se váha hrany počítá jako absolutní hodnota rozdílu spojených uzlů ke které jsem přičetl malé číslo 0.05 aby jsem při hledání nejkratší cesty donutil algoritmus nejen

[3]http://networkx.github.io



Obrázek 3.13: Spojení uzlů grafu podle čtyřokolí

hledat nejlevnější cestu, ale také tu nejkratší.

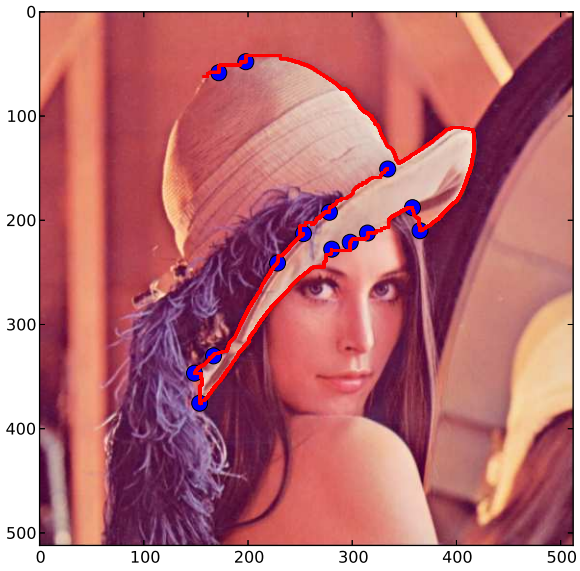
Další možností je použít exponenciální funkci pro výpočet váhy hran a to ve tvaru:

$$w(a,b) = e^{|a-b|} \tag{3.1}$$

V takto vytvořeném grafu je možné pomocí algoritmu pro hledání nejkratších cest provádět segmentaci obrazu. Ukázka segmentace na obrázku č. 3.14.

3.2 Kontrola stability segmentace

Jako zajímavý zlepšovací prvek byla navrhnuta metoda pro kontrolu stability segmentace. Jedná se o algoritmus zajišťující automatické přidávání průběžných bodů při segmentaci. Algoritmus kontroluje nově spočtenou cestu průchodu obrazem s tou, která byla spočtena o krok dříve, a pokud se nová cesta od staré liší pouze v přidaném úseku, je uznána jako stabilní. Stabilní cesta se uloží, aby byla použita při kontrole další nově spočítané cesty. Avšak pokud nově spočítaná cesta obsahuje rozdíly, tzn. že algoritmus najednou zvolil jiný směr průchodu obrazem, je tato cesta smazána a použije se poslední stabilní cesta. Tímto krokem vznikne průběžný bod v místě ukončení stabilní cesty a algoritmus pokračuje z toho místa. Ukázka práce algoritmu



Obrázek 3.14: Ukázka segmentace

je na obrázku č. 3.15. Vlevo se nachází zatím stabilní segmentace, na obrázku uprostřed je vidět že segmentace "ustřelila", algoritmus ji prohlásí za nestabilní a vloží průběžný bod na konec poslední stabilní segmentace a od tohoto budu pokračuje to je na obrázku vpravo.

3.3 Srovnávací test

Jako experiment pro otestování správnosti segmentace pomocí nástroje inteligentní nůžky jsem zvolil porovnání segmentace inteligentními nůžkami se segmentací provedenou člověkem. Segmentované obrazy jsem získal z on-line knihovny *Boundary Detection Benchmark*[4].

Zvolil jsem si deset obrazů (viz příloha A, obr. č. A.1 - A.10) u kterých jsem postupně prováděl segmentaci inteligentními nůžkami a testoval jsem, na kolik je hrana segmentace shodná s hranou segmentace provedené člově-

[4]http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

Obrázek 3.15: Kontrola stability segmentace. Ilustrativní obrázek.

kem.

V testu jsem použil segmentaci pomocí dvou bodů, to jest označil jsem pouze počátek a konec oblasti segmentace. Tento test šel provést pouze u čtyř z deseti obrazů, jelikož u zbytku nezvolil trasu segmentace ani podobným směrem, jakým jsem zamýšlel, a proto by výsledek takovéto segmentace o ničem nevypovídal.

Dalším testem byla segmentace pomocí pěti kliknutí. To znamená, že mimo počátečního a koncového bodu jsem zvolil i tři body průběžné přes které segmentace probíhá.

Předposledním testem byla segmentace za použití deseti bodů.

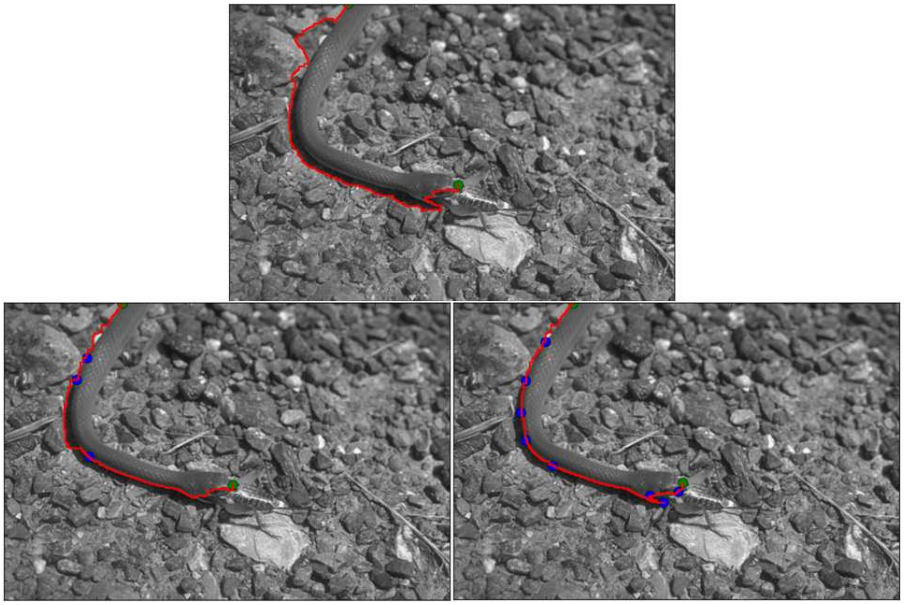
Jako poslední byl použit algoritmus pro kontrolu stability segmentace. Tento test podle očekávání měl přinést nejlepší výsledky a nejpřesnější sledování objektu. Výsledky testů jsou zaznamenány v tabulce.

Příklad segmentace inteligentními nůžkami za použití různého počtu bodů je na obrázcích č. 3.16 - 3.18. Na obrázcích č. A.1 - A.10 umístěných v příloze A je porovnání mezi šedotónovým obrázkem, jeho ruční segmentací, segmen-

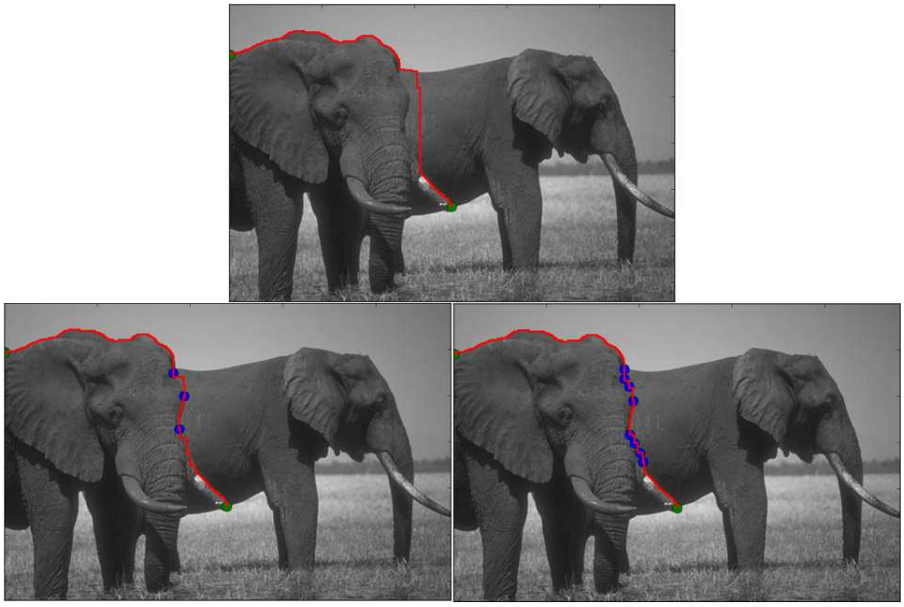
Obrázek 3.16: Segmentace obrázku 06. Nahoře 2 body, dole 5 bodů a 10 bodů.

tací pomocí deseti bodů a nakonec segmentací při použití kontroly stability.

obrázek č.	použitý práh	2 body	5 bodů	10 bodů	stabilita
1	15%	x	94.55%	96.82%	97.96%
2	20%	x	87.96%	83.11%	96.41%
3	15%	x	92.14%	99.41%	98.97%
4	8%	x	91.60%	92.30%	97.29%
5	8%	x	83.71%	91.47%	98.63%
6	8%	70.05%	89.84%	91.38%	98.14%
7	8%	52.58%	70.46%	83.30%	99.49%
8	15%	79.09%	95.86%	98.35%	99.63%
9	8%	x	88.78%	96.90%	99.33%
10	15%	86.41%	91.96%	96.77%	99.26%



Obrázek 3.17: Segmentace obrázku 07. Nahoře 2 body, dole 5 bodů a 10 bodů.



Obrázek 3.18: Segmentace obrázku 08. Nahoře 2 body, dole 5 bodů a 10 bodů.

Kapitola 4

Závěr

Hlavním cílem práce bylo shrnout poznatky o segmentaci obrazu, dále pak implementovat a otestovat jednu s grafových metod segmentace obrazu. Pro implementaci jsem zvolil metodu segmentace inteligentní nůžky. Pro implementaci bylo nutné převést obrazovou informaci do požadovaného tvaru, k tomuto účelu bylo použito několik metod.

Nejdříve byl obraz převeden do formátu matice, aby s ním bylo možno v počítači pracovat. Dále byl použit hranový detektor. Zde byl ze tří testovaných detektorů a to Sobelova, Cannyho a Prewittové detektoru, vybrán Sobelův detektor z důvodů dostatečné přesnosti a snadného nastavení.

Dalším krokem při zpracování hranového obrazu bylo experimentální nastavení prahu pro odfiltrování nepodstatných hran. Bylo zjištěno, že se tento práh u testovaných obrazů pohyboval mezi 8% až 20%.

Následujícím navrhovaným krokem byla binární dilatace hran za účelem odstranění nespojitostí v hranách. Tento krok byl zavržen z důvodu neblahého efektu ztluštění hrany, což vedlo k nepřesnostem při segmentaci, obzvláště pokud se nacházely v obraze hrany blízko u sebe.

Posledním krokem při zpracování hranového obrazu byla vzdálenostní

transformace, která převedla obraz z formátu, ve kterém pouze určoval, zda daný bod je hrana, či nikoliv, do formátu, který měl pro každý obrazový bod spočtenu vzdálenost k nejbližší hraně.

V dalším kroku se hranový obraz převedl na graf, kde vrcholy byly spojeny čtyřokolím a byla spočtena váha jednotlivých hran jako absolutní hodnota rozdílu uzlů které spojuje.

Pro samotnou segmentaci, byl použit Dijkstrův algoritmus pro hledání nejkratší cesty mezi dvěma body.

S takto funkční segmentací byl proveden experiment za použití obrazů a jejich ruční segmentace z internetové databáze Boundary Detection Benchmark. Test byl prováděn na deseti obrazech. Měřítkem úspěšnosti bylo procentuální zastoupení hrany vytvořené segmentací pomocí inteligentních nůžek na hraně vytvořené člověkem. Testoval jsem segmentaci při použití pouze dvou bodů, počátečního a koncového. Tato segmentace šla použít jen u čtyř z deseti obrazů. U zbylých šesti vedl algoritmus hranu jinou než zamýšlenou cestou. Průměrný úspěch této segmentace byl 64.6%. Dalším testem byla segmentace za použití pěti bodů, počátečního, koncového a tří průběžných. Průměrný úspěch této segmentace byl 88.7%. Předposledním testem byla segmentace za pomoci deseti bodů, jejíž průměrný výsledek byl 93.0%. Nakonec jsem použil algoritmus pro stabilitu segmentace. Její výsledek byl dle očekávání nejlepší, jelikož se vlastně jedná již z velké části o ruční segmentaci. Průměrný výsledek tohoto pokusu byl 98.5%.

Jak se dalo očekávat segmentační nástroj inteligentní nůžky, je možno použít k velmi přesné segmentaci při použití dostatečného počtu bodů.

Literatura

[1] Milan Šonka, Václav Hlaváč, Roger Boyle. *Image Processing, Analysis and Machine Vision, International Student Edition.* Thomson Learning EMEA, Limited, 2008.

[2] Miloš Železný. *Zpracování digitalizovaného obrazu.* Západočeská univerzita v Plzni, Katedra kybernetiky, 2006.

[3] Dieter Jungnickel. *Graphs, Networks and Algorithms.* Springer, 2007.

[4] J. A. Bondy, U. S. R. Murty. *Graph theory with applications.* Elsevier Science Publishing Co., Inc., Fifth Printing, 1982.

[5] Eric N. Mortensen, William A. Barrett. *Interactive Segmentation with Intelligent Scissors.* Graphical Models and Image Processing, Volume 60, Issue 5, September 1998, strany 349–384.

[6] A. Kershenbaum, R. Van Slyke. *Proceedings of the ACM annual conference-Volume 1.* ACM, 1972.

[7] P. Felzenszwalb, D. Huttenlocher. *Efficient Graph-Based Image Segmentation.* IJCV 59(2), September 2004.

[8] Ford, Lester R., Delbert R. Fulkerson. *Management Science* 5.1 (1958), strany 97-101.

[9] Ding, Lijun, and Ardeshir Goshtasby. *Pattern Recognition.* 34.3 (2001), strany 721-725.

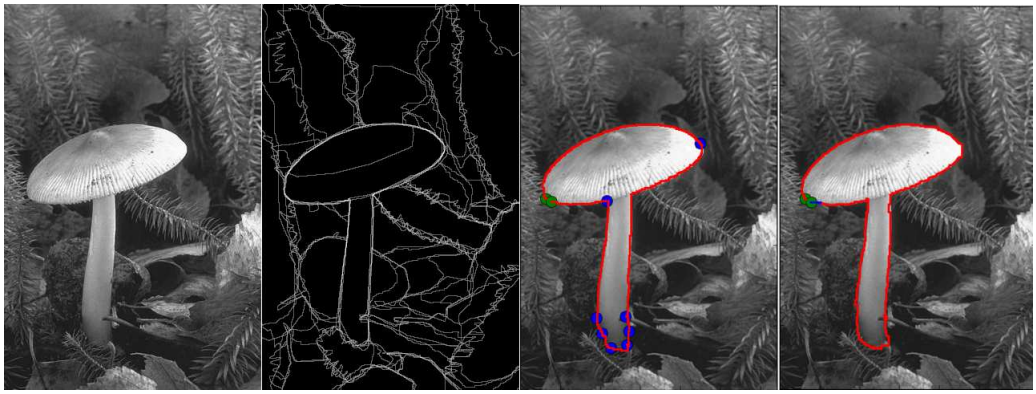
Seznam obrázků

Příloha A

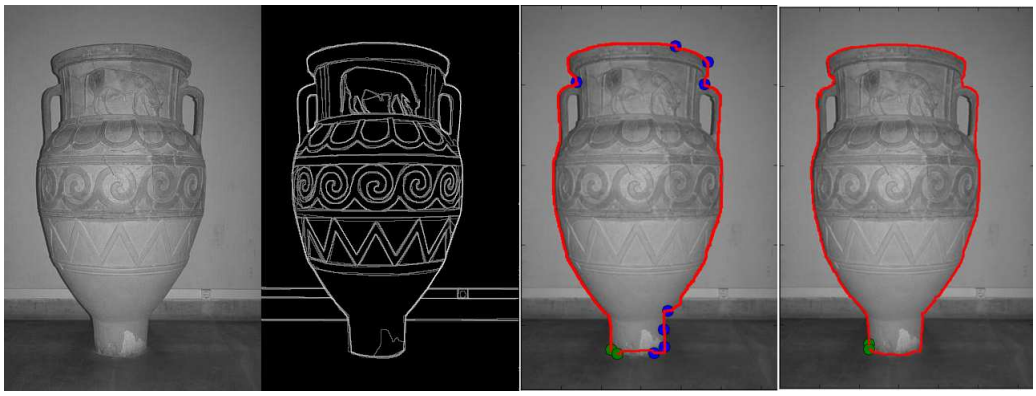
Obrázky srovnávací test

Obrázek A.1: Obr. 01. Šedotón, ruční segmentace, 10 bodů, stabilita.

Obrázek A.2: Obr. 02. Šedotón, ruční segmentace, 10 bodů, stabilita.

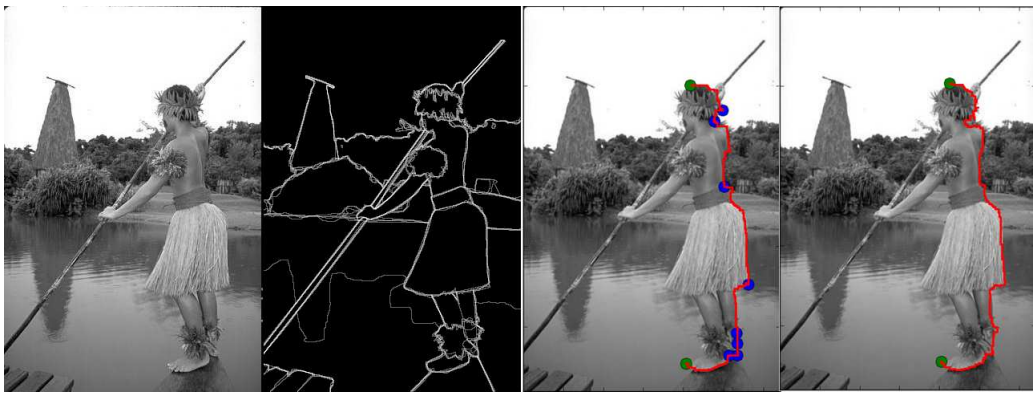


Obrázek A.3: Obr. 03. Šedotón, ruční segmentace, 10 bodů, stabilita.

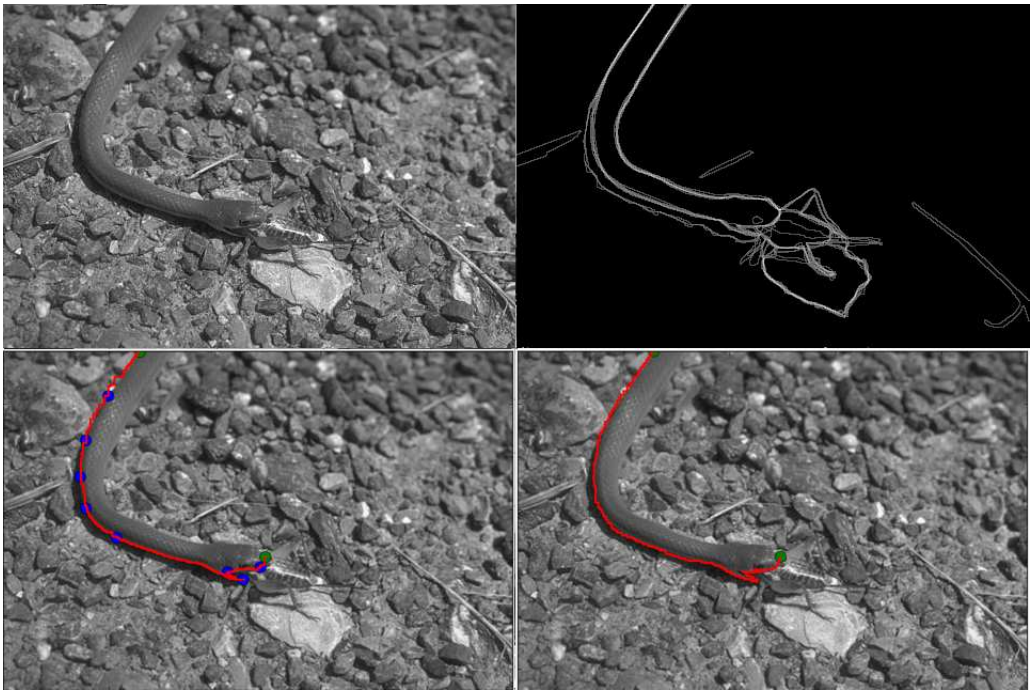


Obrázek A.4: Obr. 04. Šedotón, ruční segmentace, 10 bodů, stabilita.

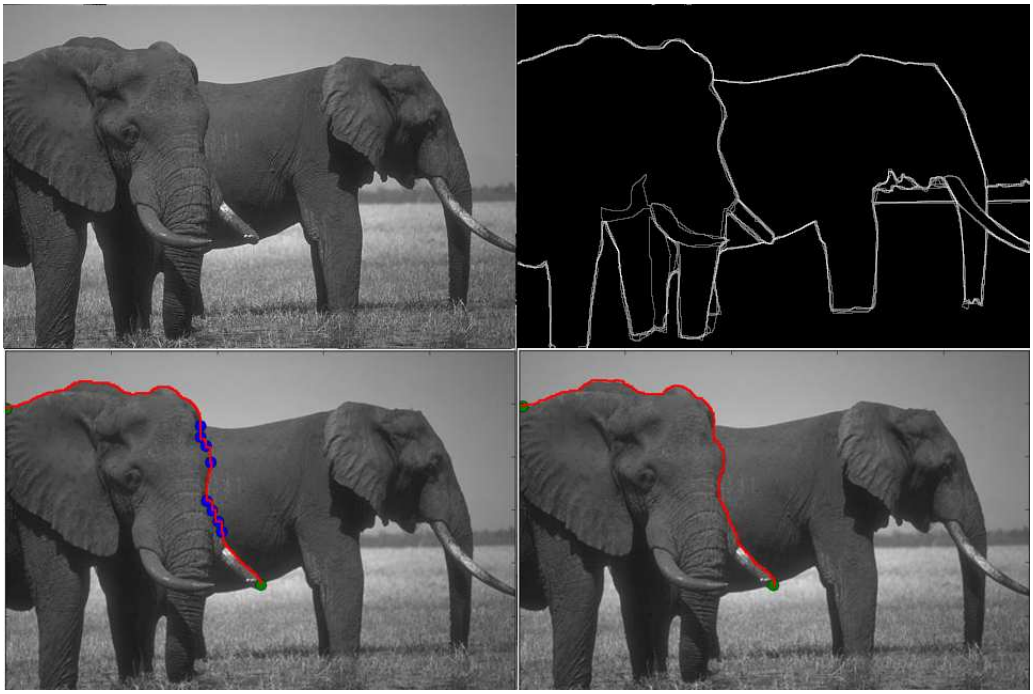
Obrázek A.5: Obr. 05. Šedotón, ruční segmentace, 10 bodů, stabilita.



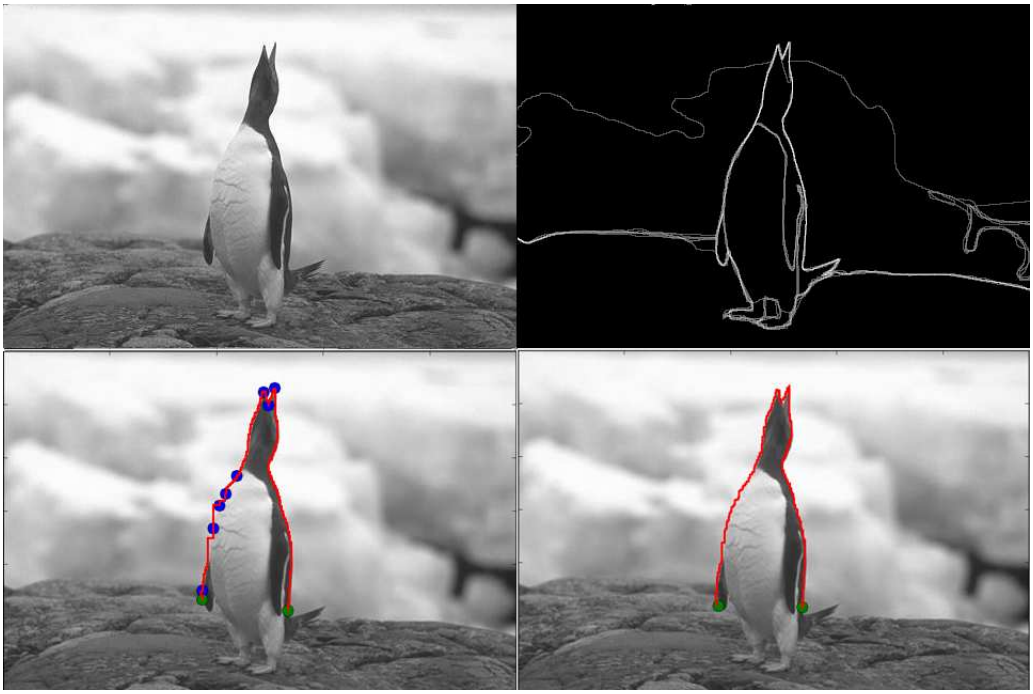
Obrázek A.6: Obr. 06. Šedotón, ruční segmentace, 10 bodů, stabilita.



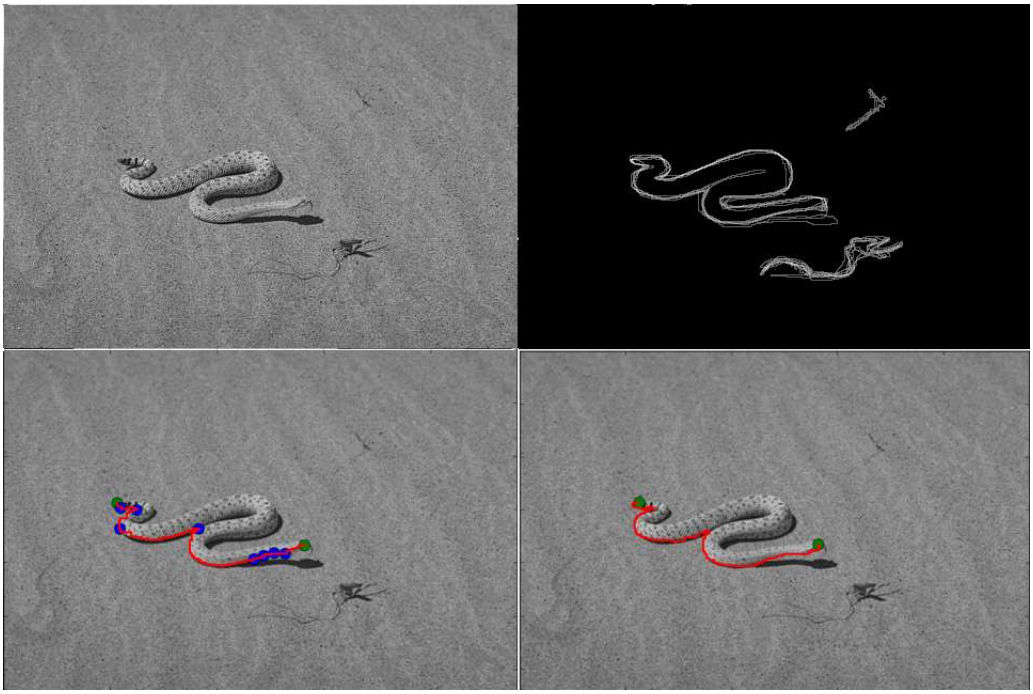
Obrázek A.7: Obr. 07. Šedotón, ruční segmentace, 10 bodů, stabilita.



Obrázek A.8: Obr. 08. Šedotón, ruční segmentace, 10 bodů, stabilita.



Obrázek A.9: Obr. 09. Šedotón, ruční segmentace, 10 bodů, stabilita.



Obrázek A.10: Obr. 10. Šedotón, ruční segmentace, 10 bodů, stabilita.