VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vyhľadávanie fráz v netextových súboroch

Phrases Search in non-text Files

2017 Andrej Gulčík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Andrej Gulčík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vyhledávání frází v netextových souborech
Phrases Search in non-text Files

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je návrh a implementace softwaru na vyhledávání libovolných frází v netextových souborech.
1. Seznamte se s problematikou detekce slov v netextových souborech se zaměřením na JPEG soubory.
2. Získejte přehled jednotlivých metodik na detekci slov za posledních 6 let.
3. Jednotlivé metody analyzujte, zhodnoťte klady a zápory.
4. Implementujte software v jazyce Java, na vyhledávání frází v JPEG souborech, pomocí nejvhodnějších metod.
5. Proveďte zhodnocení dosažených výsledků.
6. Vypracujte uživatelskou příručku a programátorskou dokumentaci.

Seznam doporučené odborné literatury:

[1] YE, Qixiang; DOERMANN, David. Text detection and recognition in imagery: A survey. IEEE transactions on pattern analysis and machine intelligence, 2015, 37.7: 1480-1500.
[2] YIN, Xu-Cheng, et al. Robust text detection in natural scene images. IEEE transactions on pattern analysis and machine intelligence, 2014, 36.5: 970-983.
[3] SHAH, Mansi; JETHAVA, Gordhan B. A literature review on hand written character recognition. Indian Streams Research Journal, 2013, 3.2: 1-19.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Hendrych**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 Gulčík.....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017 .....

Rád by som sa poďakoval svojmu vedúcemu práce Ing. Jakubovi Hendrychovi za odbornú pomoc a usmernenie pri písaní mojej práce, za cenné rady, informácie a v neposlednom rade za ochotu.

Abstrakt

Cieľom tejto bakalárskej práce bolo vytvorenie programu, schopného extrahovať text z obrazových dát vo formáte JPEG a následne v tomto texte vyhľadať zadanú frázu. Taktiež bolo nutné zoznámiť sa so základnými technikami používaými v týchto systémoch. V úvode práce je predstavená stručná história týchto systémov, ako aj súčasný výskum. V ďalších kapitolách sa venuje rôznym metódam, ktoré sú používané v týchto systémoch. Tieto kapitoly sú rozdelené podľa jednotlivých fáz systému a zoradené sú v poradí v akom sa vykonávajú. V závere je predstavená vlastná implementácia. V tejto časti sú bližšie vysvetlené postupy, ktoré boli v práci použité. Záver práce je venovaný zhrnutiu dosiahnutých výsledkov. Výsledkom tejto práce je systém pre vyhľadávanie fŕaz v obrazový dátach formátu JPEG implementovaný v jazyku Java. K tomuto systému bola vytvorená aj programátorská dokumentácia a uživatelská príručka.

Kľúčové slová: OCR, Java, JPEG, Počítačové videnie, Rozpoznávanie textu, Segmentácia, Momentový popis

Abstract

The aim of the bachelor thesis was to create the software able to extract text from visual data in JPEG form and afterwards seek out the assigned phrase. Alongside, it was also necessary to undrestand the basic techniques used in common softwares. In the opening part of the thesis is introduced the brief history of these kind of softwares as well as current development. Next chapters are focused on different metodies used in common softwares. These chapters are divided according to particular phrases of system and lined up in execution orders. In the end of the thesis is introduced the own inplementation. There are also in detail explained the used methodies and summary of achieved results. The result of the thesis is the software for searching the phrases in JPEG form implemented in JAVA. For this software was also created a program documantation and user manual.

Key Words: OCR, Java, JPEG, Computer vision, Text recognition, Segmentation, Moments descriptors

Obsah

Zoznam použitých skratiek a symbolv

DoG	–	Difference of Gaussians
IWR	–	Intelligent Word Recogniton
LoG	–	Laplacian of Gaussian
OCR	–	Optical Character Recogniton
min	–	Minúta

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

Svoju bakalársku prácu som zameral na vyhľadávanie textu v obrazových dátach. Táto problematika spadá do oblasti Počítačového videnia, ktoré sa zaoberá simuláciou schopností ľudského oka.

Počítačové videnie sa dnes uplatňuje v mnohých oblastiach, na rôzne účely ako rozpoznávanie smerovacieho čísla na poštách, rozpoznávanie poznávacích značiek či rozpoznávanie tváre.

Vďaka vývoju výkonnejšieho a cenovo dostupnejšieho hardwaru sa počítačové videnie uplatňuje v čoraz väčšej miere. To bol jeden z dôvodov prečo som si vybral práve túto oblasť, nakoľko považujem problematiku rozpoznávania objektov za zaujimavú a čoraz viac používanú v rôznych oblastiach.

Cieľom tejto práce bolo vytvorenie programu schopného vyhľadať frázu v netextovom súbore formátu JPEG. Systémy zaoberajúce sa rozpoznávanim textu označujeme ako OCR systémy. Tento systém mal byť implementovaný v jazyku Java a mal vyžadovať, čo najmenšiu interkaciu s uživateľom.

V úvodnej časti sa čitateľ oboznámi s históriou a súčasným výskumom v oblasti OCR systémov. Následne sú predstavené používané metódy OCR.

Zvyšok teoretickej časti tejto práce je venovaný opisu metód využívaných v týchto systémoch. Tieto metódy sú zoradené v poradí v akom sa používajú v samotnom systéme.

V kapitole 4 sú predstavené metódy predspracovania obrazu. Tieto metódy slúžia na odstránenie šumu a prevod farebného obrazu na šedotónový.

V nasledujúcej kapitole sú opísané základné segmentačné metódy a dve základné morfologické operácie - Dilatácia a Erózia.

Kapitola 6 je venovaná príznakom, pomocou ktorých sa klasifikuje obraz. Čitateľ sa v nej oboznámi so základnými príznakmi ako aj s príznakmi založenými na momentoch.

Samotnej klasifikáci je venovaná kapitola 7. V nej sú uvedené krátke popisy jednotlivých klasifikačných metód, ktoré sa v OCR systémoch používajú.

Záver práce je venovaný vlastnej implementáci systému v jayku Java. V tejto časti je čitateľ oboznámený s postupom riešenia a s výsledkami, ktoré systém dosiahol. Taktiež sú zobrazené výstupy jednotlivých fáz.

2 História a súčasný stav OCR

V roku 1929 Gustáv Tauschek podal patent na prvý prístroj využívajúci OCR. Tento prístroj fungoval na prekrývaní sa šablóny a obrazu. V prípade prekrytia oznámil fotosnímač zhodu. Od roku 1950 sa vývoj týchto systémov urýchlil a v roku 1954 vznikol OCR systém od Readers Digest, ktorý spracovával vytlačené finančné správy.

Medzi rokmi 1960–1965 vznikla ďalšia generácia systémov. Jednalo sa o veľmi jednoduché spracovanie znakov. Pre tieto účely vznikli špeciálne znakové sady. Taktiež sa začali objavovať systémy, ktoré boli schopné rozoznať viacero fontov. OCR systémy tejto generácie boli založené na porovnávaní obrazu znaku s obrazom uloženým v knižnici znakov.

V roku 1965 predstavila spoločnosť IBM nový OCR systém s názvom IBM 1287. Tento systém dokázal rozpoznávať bežné texty a taktiež ručne písane číslice. V rovnakom roku spoločnosť Toshiba predstavila prvý automatický stroj na triedenie pošty. Tento stroj slúžil na triedenie pošty podľa smerovacých čísel. O rok neskôr vyšli štandardizované fonty OCR-A a OCR-B určené pre Ameriku respektíve Európu. Tieto znaky boli oproti ich predchodcovi z prvej generácie prirodzenejšie a preto aj ľahšie čitateľné pre človeka.

V 70. rokoch 20. storočia vznikli systémy, ktoré boli schopné rozpoznať dokumenty s nižšou kvalitou a mali taktiež vyššiu úspešnosť v samotnom rozpoznávaní znakov. Vývoj hardwaru v tomto období viedol k znižovaniu ceny a zvýšeniu výkonu týchto systémov. Práve vďaka nižšej cene sa stali OCR systémy dostupnejšie a začali sa využívať v zdravotníctve, na poštách atď.

Vďaka nižším cenám sa OCR systémy dostávajú na osobné počítače a mobily.

Súčasný vývoj systémov je orientovaný najmä na rukou písané texty a systémy pre rozpoznávanie nelatinských znakov ako je indické písmo [1] alebo dokonca rekonštrukcia poškodených thaiských znakov [2]. Taktiež bol predstavený systém na rozpoznávanie brailovho písma [3] Dnešné systémy využívajú pokročilé techniky ako genetické algoritmy a neurónové siete. Vývoj rýchlejších a efektívnejších metód spolu s narastajúcim hardwarovým výkonom zariadení viedol k rozšíreniu na osobné počítače ale dokonca aj na mobilné platformy. V roku 2016 bola predstavená aplikácie pre Android zariadenia, ktorá prekladala odfotený text [4].

Niektoré systémy sú schopné okrem rozpoznania znaku určiť aj jeho vlastnosti, označujú sa *IWR*. Sú schopné rozpoznať slovo ako celok, prípadne frázu a v prípade detekcie chybných znakov vykonať korekciu.

3 Metódy OCR

3.1 Template matching algorithm

Táto metóda je založená na porovnávaní vstupného obrazu so šablónami, ktoré má systém uložené v databáze. Vyberá sa časť obrazu v ktorej sa predpokladá výskyt znaku a táto časť je porovnávaná so šablónami v knižnici programu. Za rozpoznaný znak sa považuje ten, ktorý mal najvyššiu, prípadne dostatočnú podobnosť so šablónou. To zároveň vyžaduje aby bol font vstupných a výstupných dát identický, prípadne veľmi podobný, aby bolo možné dosiahnuť dostatočnú zhodu znakov. Ďalším problémom tejto metódy je nutnosť presnej horizontálnej orientácie písma. To však pri fotení fotoaparátom, ručne písaných textoch prípadne nepresnom uložení dokumentu pri skenovaní vedie k neúspešnému rozpoznávaniu znakov.

3.2 Extrakcia vlastností obrazu

Táto metóda pozostáva z dvoch krokov :

1. **Samotná extrakcia** – V tomto kroku zisťujeme prítomnosť čiar a kriviek, ich pretnutia, orientáciu, umiestnenie a iné.
2. **Rozpoznávanie písmen** – Podľa získaných vlastností z extrakcie určujeme o aký znak sa jedná.

Táto metóda na rozdiel od predchádzajúcej nie je závislá na orientácií znakov.

3.3 Pomocou neurónových sieti

Táto metóda je v dnešných systémoch najpoužívanejšia najmä vďaka veľkej flexibilite a efektivite. Tieto systémy si pomocou učiaceho algoritmu trénujú rozpoznávanie na tréningovej množine. Je však potrebné vytvoriť neurónovú sieť a potrebné *learning sety*.

3.4 Pomocou Fuzzy logiky

Fuzzy logika je založená na Fuzzy množinách. Rozdiel medzi Fuzzy množinou a normálnou množinou je ten, že kým prvok x buď patrí alebo nepatrí do množiny X, tak môže prvok x patriť prípadne nepatriť do Fuzzy množiny len čiastočne. To ako veľmi daný prvok patrí do Fuzzy množiny určuje funkcia príslušnosti, ktorá tomuto prvku priradí hodnotu z intervalu $\langle 0, 1 \rangle$. Podľa [5] je možné používať túto logiku aj v OCR systémoch napríklad pri identifikácií znakov.

4 Predspracovanie obrazu

Predspracovanie obrazu je veľmi dôležitá fáza, ktorej cieľom je previesť obraz do takej podoby, ktorá je vhodná pre ďalšie rozpoznávanie textu. Tento proces zahŕňa rôzne operácie s obrazom, ako napríklad prevod farebného obrazu na šedotónový, prípadne aplikáciu rôznych filtrov za účelom úpravy obrazu pre nasledovnú segmentáciu.

4.1 Prevod na Šedotónový obraz

Pri prevode na šedotónový obraz sa získavajú pre každý pixel hodnoty jeho RGB zložiek, ktoré sú nahradené hodnotou I. Výsledná farba má teda hodnoty $R{=}G{=}B{=}I$. Podľa [6] je možné z hľadiska vnímania obrazu človekom použiť vzorec 1, avšak z hľadiska ďalšieho spracovania obrazu je vhodnejšie použiť vzorec 2, nakoľko môžeme pracovať iba s jednou dominantnou zložkou.

$$I = 0.299R + 0.587G + 0.144B \tag{1}$$

$$I = 0.33R + 0.33G + 0.3B \tag{2}$$

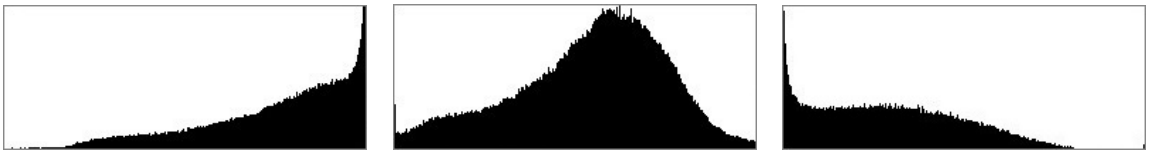
4.2 Histogram

Histogram predstavuje početnosť jednotlivých úrovní jasu. Graficky je znázornený tak, že osa y predstavuje hodnoty úrovne jasu (pri šedotónovom obraze nadobúda hodnoty 0–255) a osa x predstavuje počet pixelov s danou úrovňou jasu. Histogram je možné vytvoriť aj pre farebné obrazy, kde každá zložka má vlastný histogram. Informácie získané z histogramu môžeme využívať pri prahovani alebo vylepšeniach vzhľadu. Môžeme pomocou neho určiť podexponovanosť prípadne preexponovanosť. Príklady histogramov je možné vidieť na obrázku 1.

Často sa môžeme stretnúť s histogramom v ktorom je vyjadrená pravdepodobnosť výskytu jednotlivých úrovní jasu. Takýto histogram nazývame normalizovaný a jednotlivé hodnoty pravdepodobností výskytu danej úrovne jasu $P(x_i)$ získame pomocou vzorca 3

$$P_{(x_i)} = \frac{n_i}{n}, i \in 0, 1, 2, \ldots, L-1 \tag{3}$$

kde n_i je počet pixelov s intenzitou i a n je počet všetkých pixelov.



(a) Histogram preexponovaného obrázku.

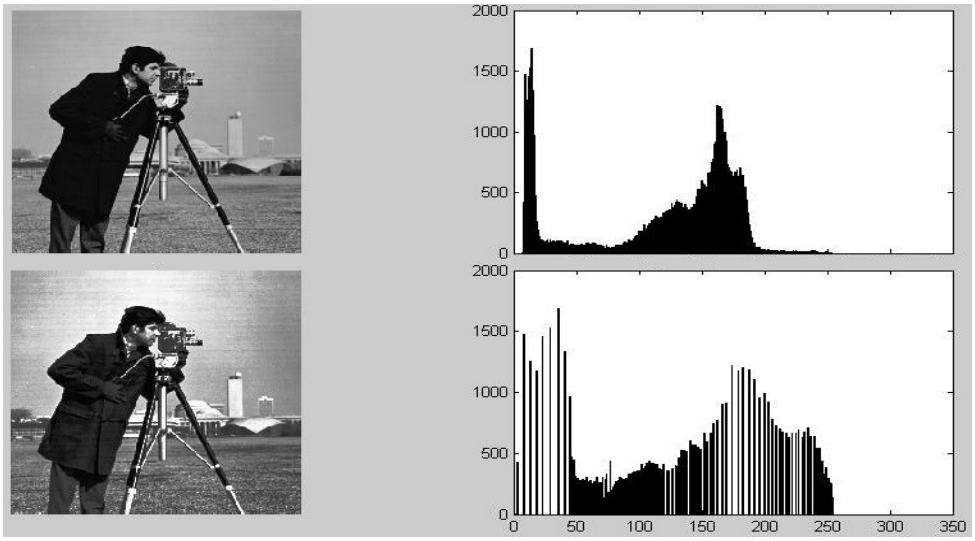
(b) Ideálny histogram.

(c) Histogram podexponovaného obrázku.

Obr. 1: Príklady histogramov

4.2.1 Histogramová ekvalizácia

V prípade nevhodne exponovaného obrázka je vhodné histogram „vyrovnať“. To môžeme dosiahnuť histogramovou ekvalizáciou. Na obrázku 2 môžeme vidieť obrázok a jeho histogram pred a po histogramovej ekvalizácií.



Obr. 2: Histogramová ekvalizácia.

Pri tejto metóde sa snažíme vyrovnať početnosť výskytu jednotlivých úrovní jasu. K ekvalizácií je nutné definovať kumulatívnu distribučnú funkciu danú vzorcom 4

$$c(i) = \sum_{j=0}^{i} P(j) \tag{4}$$

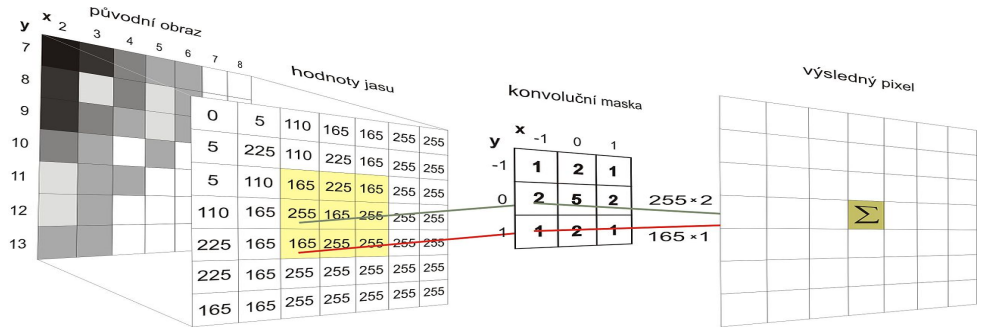
kde i je úroveň intenzity jasu a $P(j)$ je pravdepodobnosť výskytu pixelu s úrovňou jasu j. Touto metódou je možné zlepšiť kontrast obrázku avšak nie vždy je vhodné jej požitie nakoľko môže zvýrazniť pixely pozadia čo vedie k problémom pri segmentácii.

4.3 Filtrácie obrazu zahrňujúce okolie

Tieto filtrácie zahrňujú okolie spracovávaných bodov v obraze a teda výsledná hodnota skúmaného pixelu závisý na hodnote pixelov jeho okolia. Aplikáciou filtrov sa snažíme odstrániť nežiadúce efekty ktoré vznikajú pri snímaní obrazu – šum. Existujú rôzne druhy šumov a preto je potrebné zvoliť správnu metódu na ich odstránenie.

4.3.1 Konvolúcia

S týmto pojmom sa pri spracovaní obrazu stretneme pomerne často. Konvolúcia využíva masku, ktorá obsahuje číselné hodnoty. Túto masku môžeme označiť ako konvolučné jadro (*kernel*). Táto maska sa pohybuje po obraze a pre každý pixel ktorý je ňou prekrytý sa vykoná súčin s hodnotou v maske. Výsledný súčet sa vloží do nového obrázku na pozíciu x.



Obr. 3: Princíp 2D diskrétnej konvolúcie

Diskrétnu konvolúciu obrazu f a konvolučného jadra h môžeme definovať vzorcom 5 :

$$(f * h)(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x - i, y - j) * h(x, y) \tag{5}$$

4.3.2 Vyhladzovanie

Vyhladzovanie odstraňuje šum pomocou odstraňovania ostrých hrán aplikáciou matice koeficientov (konvolučného jadra) na obraz. Existujú viaceré metódy vyhladzovania. Medzi základné podľa [6] patria :

Spriemerovanie hodnota pixelu je určená priemernou hodnotou okolitých pixelov. Táto metóda potláča škvrny šumu, ktoré sú menšie ako veľkosť skúmaného okolia. Z tohto dôvodu by malo byť skúmané okolie menšie ako veľkosť najmenšieho významného detailu aby nedochádzalo k jeho chybnej zámene za šum. Nevýhodou tejto metódy je rozmazávanie hrán. Konvolučná maska pre okolie 1 pixelu vyzerá nasledovne :

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \frac{1}{9}$$

Gaussové vyhladzovanie hodnota pixelu je určená konvolúciou s maskou, kde koeficienty bližšie k stredu masky majú vyššiu hodnotu a odpovedajú hodnotám na gaussovej krivke. Dvojrozmerné Gaussové rozdelenie, kde stredná hodnota má hodnotu 0 je dané vzťahom 6

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{6}$$

kde σ^2 je rozptyl ktorý určuje strmosť Gaussovejfunkcie [7].

Mediánová filtrácia Jedná sa o nelineárny filter ktorého princíp spočíva vo výbere mediánu hodnôt pod maskou. V špecifických prípadoch (šum typu korenie a soľ) vykazuje veľmi dobré výsledky.

5 Segmentácia obrazu

Pomocou segmentácie obrazu oddeľujeme popredie od pozadia obrazu čím extrahujeme pre nás dôležité objekty. Táto časť patrí medzi najzložitejšie úlohy systému nakoľko nesprávne rozdelenie objektov vedie k strate informácií a následnému chybnému rozpoznávaniu. Pri segmentácií hľadáme podobnosti, homogénne oblasti, nespojitostí prípadne využívame hybridné metódy.

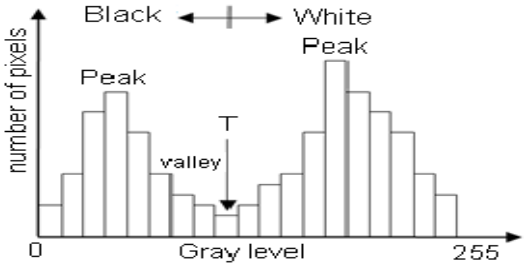
5.1 Prahovanie

Prahovanie slúži na vyčlenenie pixelov s informačnou hodnotou (pixely objektu) od ostatných (pozadie) za základne určitej hodnoty – práhu T. Výsledkom je čiernobiely (hodnoty 0 alebo 255) prípadne binárny (hodnoty 0 alebo 1) obraz. Jedná sa o pomerne spoľahlivú a jednoduchú metódu avšak vychádza z predpokladu, že hľadané oblasti majú rovnakú alebo podobnú úroveň jasu. Matematicky môžeme prahovanie zapísať ako 7

$$f(x) = \begin{cases} 1, & x \geq T \\ 0, & x < T \end{cases} \tag{7}$$

5.1.1 Automatické nájdenie prahu

Ďalším problémom prahovania je určenie samotného prahu. Keďže v tejto práci chcem minimalizovať potrebný zásah užívateľa je nutné použiť jednu z metód automatického nájdenia prahu. Najjednoduchšou z nich je metóda percentuálneho prahovania. Táto metóda vychádza z predpokladu, že poznáme akú veľkú oblasť má objekt na obraze zaberať a upravíme prah na takú hodnotu aby sa tento počet zhodoval s počtom pixelov objektu po prahovani. Keďže však nedokážeme túto veľkosť určiť je táto metóda pre túto prácu nepoužiteľná. Ďalšou možnosťou je využitie histogramu obrazu. V prípade bimodálneho histogramu nájdeme dve lokálne maxima a teda prahom by bolo minimum medzi nimi. Bimodálny histogram je na obrázku 4.



Obr. 4: Bimodálny histogram

Problém nastane v prípade že sa v histograme nachádza viacero maxím a nie vždy vieme určiť ich význam. Pri tom nám môže pomôcť nami zvolená vzdialenosť d . Za významné lokálne maximá potom považujeme iba tie, ktoré sú od seba vzdialené minimálne d jasových úrovní.

5.1.2 Otshuova metóda

Táto metóda je jednou z najpoužívanejších metód automatického určenia prahu. Princípom je rozdelenie pixelov do dvoch tried a minimalizácia ich spoločného rozptylu. Hodnotou prahu T dosiahneme maximalizáciou výrazu 8 :

$$\sigma_b^2(T) = n_b(T)\, n_o(T)\, [\mu_b(T) - \mu_o(T)]^2 \tag{8}$$

$$n_b(T) = \sum_{i=0}^{T-1} p(i) \tag{9}$$

$$n_o(T) = \sum_{i=T}^{N-1} p(i) \tag{10}$$

$$\mu_b(T) = \sum_{i=0}^{T-1} \frac{ip(i)}{n_b(T)} \tag{11}$$

$$\mu_o(T) = \sum_{i=T}^{N-1} \frac{ip(i)}{n_o(T)} \tag{12}$$

Kde $\sigma_b^2(T)$ je vzájomný rozptyl oboch skupín bodov, T je aktuálne skúmaná hodnota prahu a $[0, N\check{}1]$ je rozsah úrovní jasu. $p(i)$ je počet pixelov s danou intenzitou. n_b predstavuje počet pixelov pozadia, n_o počet pixelov objektu. μ_b a μ_o predstavujú priemer pixelov pozadia a objektu.

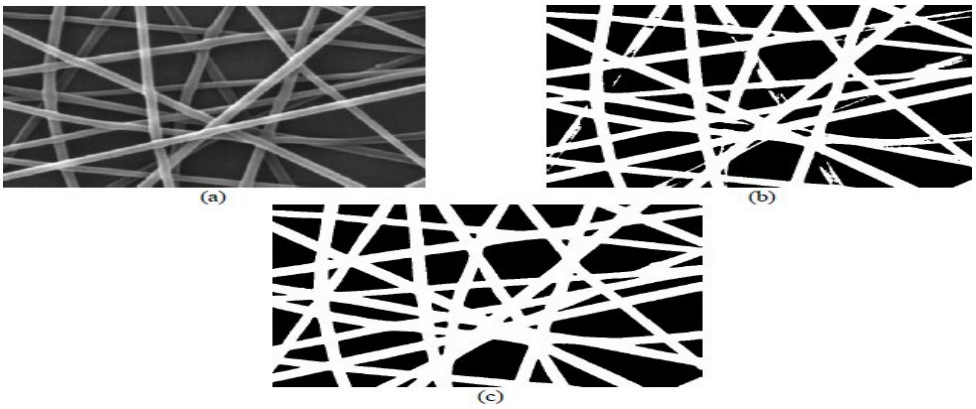
Tento postup môžeme pomocou znalosti histogramu zjednodušiť a urýchliť a to tak, že budeme presúvať pixely z jednej skupiny do druhej.

5.1.3 Lokálne prahovanie

V prípade, že je obrázok nerovnomerne osvetlený alebo má v rôznych častiach rôzne hodnoty jasu je vhodnejšie určovať prah podľa okolia pixelu. V tomto prípade sa teda urči veľkosť masky, ktorá urči veľkosť skúmaného okolia. Prah následne určime na základe informácií z tohto okolia. Pre porovnanie lokálneho a globálneho prahovania poslúži obrázok 5, kde môžeme vidieť chybné vyhodnotenie pri globálnom prahovaní (obrázok 5,b) a výsledok lokálneho prahovania (obrázok 5,c).

5.2 Detekcia hrán

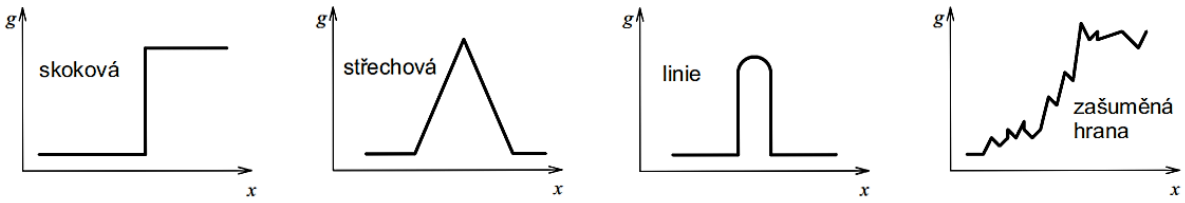
Pri detekcii hrán hľadáme miesta v obraze, kde sa významne mení hodnota jasu. Tieto zmeny môžeme detegovať pomocou prvej derivácie, druhej derivácie, pripadne detekciou zmeny znamienka derivácie. Pri detekcii hrán je nutné detegovať body hrany a následne hranu samotnú, čím získame požadovanú mapu hrán.



Obr. 5: Porovnanie lokálneho a globálneho prahovania

5.2.1 Hrana

Za hranu považujeme miesto v obraze v ktorom dochádza k výraznej zmene hodnoty jasu. Na obrázku 6 môžeme vidieť typické profily v okolí hranových bodov. Prvé tri zľava predstavujú ideálne profily, avšak reálne sa stretneme so zašumenou hranou.



Obr. 6: Typické jasové profily v okolí hranových bodov

V prípade, že je hrana zašumená je veľmi ťažké rozhodnúť, ktorá zo zmien úrovne jasu má byť považovaná za hranu. Z tohto dôvodu je vhodné aplikovať na obrázok filter na odstránenie šumu.

5.2.2 Detekcia hrán pomocou prvej derivácie

Výsledkom prvej derivácie obrazu v smeroch x a y je gradient. V mieste, kde sa nachádza hrana dochádza k najväčšej zmene intenzity. V homogénnej oblasti je prvá derivácia rovná nule [7].

5.2.2.1 Gradient Gradient je definovaný ako vektor funkcie dvoch premenných $f(x, y)$. Vypočítame ho pomocou vzorca 13. Veľkosť gradientu získame po dosadení do vzorca 14 a smer po dosadení do vzorca 15.

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{matrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{matrix} \tag{13}$$

$$mag\left(\nabla f\right) = \sqrt{G_x^2 + G_y^2} = sqrt\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \tag{14}$$

$$\Phi\left(x, y\right) = \arctan\frac{G_y}{G_x} \tag{15}$$

Gradient je kolmý na hranu a preto sa dá použiť ako informácia pri hľadaní hrán. Odhad derivácií G_x a G_y v diskrétnom obraze je možné určiť rozdielom dvoch susedných hodnôt. V [7] je uvedená symetrická varianta daná vzorcom 16 pre G_x a vzorcom 17 pre G_y

$$G_x \approx f\left(x+1, y\right) - f\left(x-1, y\right) \tag{16}$$

$$G_y \approx f\left(x, y-1\right) - f\left(x, y-1\right) \tag{17}$$

Existuje niekoľko operátorov (konvolučných jadier) pre hľadanie hrán založených na princípe gradientu. Napríklad :

- Prewittovej operátor

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- Sobelov operátor

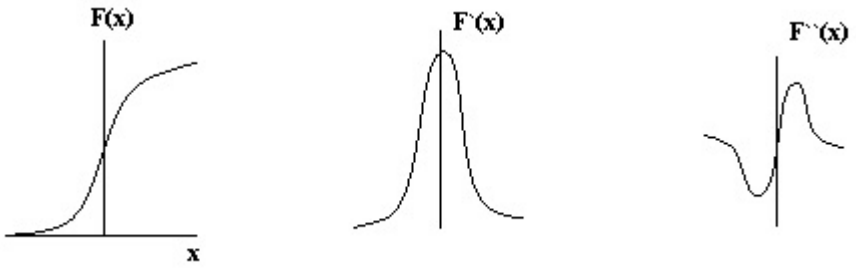
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Po konvolúcií obrazu s týmito operátormi získame hľadanú zložku gradientu. Ak chceme počítať smerové derivácie všetkých ôsmych smerov je možné masky rotovať po 45°.

5.2.3 Detekcia hrán pomocou druhej derivácie

Druhá derivácia predstavuje rýchlosť zmeny hodnôt jasu – zmenu zmeny. Túto metódu používame v prípade, že nepotrebujeme informácie o veľkosti alebo smere hrany ale vystačíme si s in-

formáciou o tom kde sa hrana nachádza. Na obrázku 7 môžeme vidieť, že v mieste maxima prvej derivácie prechádza druhá derivácia nulou.



Obr. 7: Priebeh obrazovej funkcie

Pri hľadaní hrán musíme teda vypočítať druhú deriváciu a následne určiť miesta prechodu nulou. V diskrétnom obraze je možné druhú deriváciu vypočítať pomocou vzorca 18 a 19.[7]

$$\frac{\partial^2 f(x,y)}{\partial x^2} \approx [f(x+1,y) + (x-1,y)] - 2f(x,y) \tag{18}$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} \approx [f(x,y+1) + (x,y-1)] - 2f(x,y) \tag{19}$$

5.2.3.1 Laplaceov operátor Laplaceov operátor patrí medzi operátory počítajúce druhú deriváciu. Vyznačuje sa tým, že súčet všetkých jeho prvok je rovný nule a kladie dôraz na stredové body. Tento operátor je však citlivý na šum, neurčuje smer hrany a môže spôsobiť nájdenie dvojitých hrán. Príklad Laplaceovho operátora :

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

5.2.3.2 Laplacian of Gaussian (LoG) Pomocou konvolúcie obrazu a Gaussovho filtra je možné docieliť rozmazanie obrazu a tým by sa znížila odozva šumu na konvolúciu s Laplaceovým operátorom. Tento výsledok je možné docieliť jednou operáciou (nemusíme teda najskôr aplikovať Gaussov filter a následne Laplaceov operátor). Druhá derivácia Gaussovho filtra je daná vzorcom 20. Veľkou výhodou LoG operátora je možnosť voľby veľkosti jadra. Jadro by malo byť tak veľké ako detaily, ktoré chceme filtrom zachytiť, avšak s veľkosťou jadra rastie aj časová náročnosť konvolúcie.

$$G''(x,y) = \frac{x^2+y^2-\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{20}$$

5.2.3.3 Difference of Gaussian (DoG) Pri tomto postupe sa obraz rozmaže dvakrát Gaussovým filtrom. Raz s väčším σ a raz s menším. Tieto dva obrazy od seba následne odčítame. Výsledný obraz je teda daný vzorcom 21.

$$DoG = G_{\sigma_1} - G_{\sigma_2}, kde \sigma_1 < \sigma_2 \tag{21}$$

Podľa práce Marra a Hildertha [8] dosiahneme aproximáciu LoG pri pomere $\frac{\sigma_2}{\sigma_1} = 1.6$

5.2.3.4 Nájdenie prechodu nulou Ako bolo spomínané v kapitole o detekcií hrán pomocou druhej derivácie je nutné nájsť miesta kde druhá derivácia prechádza nulou. Takto vytvoríme hľadanú mapu hrán. Jedna z možností je prejdenie obrazu bod po bode a všetky pixely kde druhá derivácia dosahuje hodnotu označiť za pixely hrany. Tento postup však nie je vhodný, pretože nulovú hodnotu druhej derivácie majú aj pixely v homogénnych oblastiach . S ďalším problémom sa môžeme stretnúť ak druhá derivácia prechádza nulou práve medzi dvoma pixelmi. Tieto pixely nemajú síce nulovú hodnotu druhej derivácie, napriek tomu sú pixelmi hrany. Tento problém môžeme odstrániť použitím masky 2x2, ktorú použijeme na vyhľadanie bodov, v ktorých došlo k zmene znamienka.

$$\begin{bmatrix} \mathbf{a} & b \\ c & d \end{bmatrix}$$

Za stred masky určíme prvok **a**. Ak sa líši jeho znamienko od iného prvku v maske je tento bod označený za hranový.

5.2.4 Cannyho detektor hrán

Na začiatku 80. rokov J. F. Canny stanovil nasledujúce vlastnosti ideálneho hranového detektora :

- **Minimálna chyba detekcie hrán** – všetky dôležité hrany musia byť detegované a nesmie byť detegovaná žiadna falošná hrana.
- **Správna lokalizácia** – vzdialenosť medzi skutočnou a detegovanou hranou musí byť čo najmenšia.
- **Iba jedna odozva** – každá hrana musí byť detegovaná ibaraz.

Tento detektor je môžeme zaradiť medzi metódy využívajúce prvú deriváciu, pretože pre detekciu hrán je nutné poznať smer a veľkosť gradientu.

Na obraz sa najskôr aplikuje filtet pre potlačenie šumu, vypočíta sa veľkosť a smer gradientu a vykoná sa *Nonmaxima suppression.*

5.2.4.1 Nonmaxima suppression V predchádzajúcej časti sa nachádza použitie tejto metódy. Jedná sa o poloprahovaciu metódu, ktorá je založená na predpoklade, že hrana dáva najväčšiu odozvu v mieste kde sa skutočne nachádza. Pre každý bod obrazu vyberie susedné body v smere gradientu a porovná ich hodnoty. Ak je hodnota daného bodu menšia ako hodnota ľubovoľného suseda priradí sa mu hodnota 0. Týmto postupom sa snažíme splniť podmienku správnej lokalizácie hrany ideálneho detektora. Veľkou nevýhodou je, že v mieste kde sa stretávajú tri hrany dôjde k ich rozpojeniu, pretože jedna hrana je vždy väčšia ako druhá.

5.3 Dilatácia

Dilatácia je morfologická operácia, ktorá slúži na zaplnenie malých otvorov v objekte, čím ich prepája a zväčšuje

5.4 Erózia

Je to opak Dilatácie. Erózia zjednodušuje štruktúru objektov, čím ich rozkladá na jednoduchšie časti. Taktiež odstraňuje nežiadúce prepojenia a malé izolované objekty. Porovnanie erózie a dilatácie môžeme vidieť na obrázku 8.

(a) Originálny obrázok.

(b) Výsledok po erózi

(c) Výsledok po dilatáci.

Obr. 8: Výsledok erózie a dilatácie

6 Extrakcia príznakov

Úlohou tejto časti systému je získanie charakteristických príznakov objektu, pomocou ktorých bude môcť klasifikátor určiť, o aký znak sa jedná. Na príznaky sú kladené rôzne požiadavky. Medzi základné podľa [6] patria :

- **Invariantnosť** na zmene jasu, kontrastu a zmene mierky.
- **Spoľahlivosť** – rovnaký objekt musí vykazovať podobné hodnoty príznakov.
- **Diskriminabilita** – rôzne objekty musia vykazovať rôzne hodnoty príznakov.
- **Efektivita výpočtu**

Hlavným problémom pri popise objektu je výber vhodných príznakov.

6.1 Vektor príznakov

Jedná sa o vektor, ktorý obsahuje číselné hodnoty alebo symboly zvolenej abecedy, ktoré reprezentujú jednotlivé príznaky. Všeobecne môže tento vektor obsahovať veľké množstvo odlišných príznakov bez ohľadu na ich diskriminabilitu [6]. Vzhľadom na fakt, že jedna zo základných požiadaviek na príznaky je práve diskriminabilita je nutné vykonať redukciu dimenzie príznakového priestoru. Na to slúžia dva základné prístupy :

- **Extrakcia príznakov** – transformácia pôvodného príznakového vektoru na vektor s nižším počtom členov, pričom nové príznaky majú odlišný význam od pôvodných [6].
- **Selekcia príznakov** – výber takej podmnožiny príznakov z pôvodného vektora príznakov, ktorá vykazuje najvyššiu diskriminabilitu. Vybraným príznakom zostáva ich význam, ale stráca sa informácia o objekte vplyvom nevyužitých príznakov [6].

6.2 Delenie príznakov

Príznaky je možné rozdeliť podľa mnohých hľadísk. Medzi základné tri podľa [6] patria :

- **Podľa domény popisovanej oblasti**
  - *Fotometrické* – odrážajú optické vlastnosti objektu
  - *Radiometrické* – odrážajú geometrické vlastnosti objektu ako veľkosť, dĺžka a podobne.
- **Podľa oblasti výpočtu**
  - *Založené na regiónoch* – nutná znalosť hodnôt pixelov objektu.

- *Založené na hraniciach* – nutná znalosť hranice objektu

- **Podľa oblasti popisu**

  - *Globálne príznaky obrazu* – tieto príznaky sú získavané pomocou integrálných transformácií (Foruierova transformácia, Diskrétna transformácia)
  - *Globálne príznaky objektu* – sú priradené celému objektu. Jedná sa o jednoduché príznaky ako napríklad veľkosť. Problémom je nutná precízna segmentácia a vysoká citlivosť na aditívny šum.
  - *Lokálne príznaky objektu* - príznak je získaný len z určitej časti lokálnej oblasti objektu. Nie je nutná predchádzajúca segmentácia nakoľko sa využívajú význačné body objektu ako rohy, priamosť, zakrivenosť a podobne.

6.2.1 Radiometrické príznaky založené na regiónoch

Tieto príznaky popisujú metrické vlastnosti objektu, ktoré získame z plošného rozloženia objektov. Medzi základné podľa [6] patria :

Veľkosť Jedná sa o jeden z najjednoduchších príznakov, ktorý vyjadruje počet pixelov daného objektu.

Obvod Je určený počtom hraničných pixelov daného objektu. Je možné ho vyjadriť pomocou 4–kolia prípadne 8–okolia. Pri 8–okolí by sme nemali zabudnúť na normalizačný koeficient $\sqrt{2}$. Tento príznak je závislý na rozlíšení.

Nekompaktnosť Tento príznak vyjadruje mieru podobnosti s kruhom. Čím nižšia je jeho hodnota, tým vyššia je jeho kompaktnosť a teda podoba s kruhom. Túto hodnotu získame pomocou vzorca $\frac{Obvod^2}{Veľkosť}$.

Konvexnost Príznak udávajúci podobnosť objektu ku svojej konvexnej schránke. Nadobúda hodnoty z intervalu $\langle 0, 1 \rangle$.

Hlavná a vedľajšia osa Hlavná a vedľajšia osa odpovedá dĺžke hlavnej respektíve vedľajšej osy v pixeloch.

Výstrednosť Výstrednosť je určená pomerom dĺžok najdlhšej tetivy a najdlhšej k nej kolmej tetivy [9].

Podlhovatosť Tento príznak je určený pomerom strán obdĺžnika s minimálnym obsahom, opísaného objektu. Postupným otáčaním objektu hľadáme opísaný obdĺžnik s minimálnym obsahom.

Pravouhlosť Pravouhlosť je určená pomerom veľkosti a plochy opísaného obdĺžnika. Rovnako ako pri podlhovatosti rotujeme objektom v snahe nájsť minimálnu plochu opísaného obdĺžnika.

Eulerovo číslo Tento príznak je určený rozdielom počtu súvislých častí objektu a počtu dier. Tento príznak je typologicky invariantný.

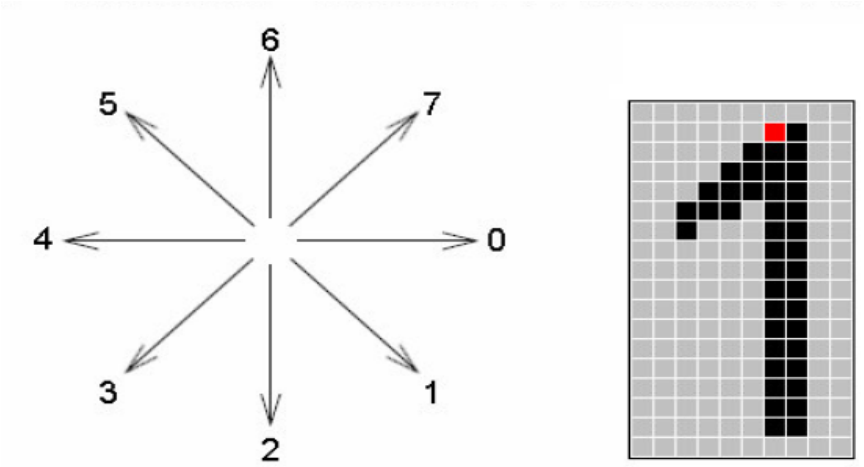
Vektor tvaru Ide o vektor pozostávajúci z dĺžok lúčov, ktoré sú vyslané všetkými smermi z ťažiska objektu k hranici objektu. Tento príznak je invariantný voči rotácii. Ak sú však hodnoty normované najdlhším lúčom na hodnotu 1 stane sa invariantným aj k zmene mierky.

6.2.2 Radiometrické príznaky založené na hraniciach

Rovnako ako predchádzajúce príznaky aj tieto popisujú metrické vlastnosti objektov, avšak tentokrát je nutná znalosť hranice objektu. Tieto hranice môžu byť určené zoznamom pixelov alebo pomocou geometrický entít. Medzi základné príznaky patrí Freemanov reťazový kód.

6.2.2.1 Freemanov reťazový kód Tento kód popisuje hranice objektov pomocou postupnosti čísel kde každá číslica vyjadruje smer. Táto postupnosť môže byť určená pomocou 4–okolia prípadne 8–okolia. Pri klasifikácii porovnávame reťazový kód objektu s reťazovým kódom vzoru. Probém však nastáva pri voľbe počiatočného bodu preto by mal byť tento bod vždy rovnaký. Freemanov kód pri opise obrázku 9 by bol nasledovný :

2 2 2 2 2 2 4 6 6 6 6 6 6 6 6 6 6 6 6 4 3 4 3 6 7 7 7 7



Obr. 9: Freemanov reťazový kód

6.2.3 Momentový popis

Geometrické momenty vychádzajú z tvaru objektu a jasových hodnôt jeho pixelov. Všeobecný moment rádu $p + q$ stanovíme pomocou vzorca 22. Všetky rovnice v tejto časti sú upravené pre diskrétne obrazy a prevzaté z [6] s výnimkou vzorca 26, kde sa pravdepodobne jedná o chybu a preto je táto rovnica prevzaná z [9].

$$m_{pq} = \sum_{Y} \sum_{X} x^p y^q s\left(x, y\right) \tag{22}$$

kde x, y prestavujú súradnice pixelov obrazu.

Tieto momenty ale nie sú invariantné voči zmene veľkosti, natočeniu ani posunutiu. Aby sme získali invariantnosť voči posunu použijeme *centralizovaný geometrický moment* definovaný v 23, kde x_t a y_t predstavujú súradnice ťažiska. Tieto súradnice je možné vypočítať pomocou momentou prvého a nultého rádu.

$$\mu_{pq} = \sum_{Y} \sum_{X} \left(x - x_t\right)^p \left(y - y_t\right)^q s\left(x, y\right) \tag{23}$$

$$x_t = \frac{m_{10}}{m_{00}} \tag{24}$$

$$y_t = \frac{m_{01}}{m_{00}} \tag{25}$$

Následne pomocou normovaného centrálneho momentu získame invariantnosť voči zmene mierky. Tento moment je daný vzťahom 26.

$$\vartheta_{pq} = \frac{\mu_{pq}}{\left(\mu_{00}\right)^{\frac{p+q+2}{2}}} \tag{26}$$

Tieto momenty sú používané pri kompozícii zložitejších štruktúr – momentových invariantov. Najstaršie a najznámejšie sú Huove momentové invarianty [6].

$$\phi_1 = \mu_{20} + \mu_{02} \tag{27}$$

$$\phi_2 = (\mu 20 - \mu_{02})^2 + 4\mu_{11}^2 \tag{28}$$

$$\phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \tag{29}$$

$$\phi_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2 \tag{30}$$

$$\begin{aligned}\phi_5 = {} & (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})\left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2\right] + \\ & + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})\left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right]\end{aligned} \tag{31}$$

$$\begin{aligned}\phi_6 = {} & (\mu_{20} - \mu_{02})\left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right] + \\ & + 4\mu_{11}^2(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})\end{aligned} \tag{32}$$

$$\begin{aligned}\phi_7 = {} & (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})\left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2\right] - \\ & - (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})\left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\right]\end{aligned} \tag{33}$$

7 Klasifikácia znakov

Klasifikácia je proces pri ktorom sa snažíme zaradiť segmentované objekty do tried. Nerozpoznávame však objekty ale ich obrazy. Ak sú tieto obrazy charakterizované vektorom hovoríme o príznakovom rozpoznávaní. Pri tomto type rozpoznávania tvorí množina všetkých obrazov n-rozmerný obrazový priestor. Ak sú množiny týchto obrazov separabilné je klasifikácia pomerne jednoduchá avšak táto vlastnosť väčšinou splnená nie je.

7.1 Metóda nájdenia zhody

Veľmi jednoduchá a stará metóda klasifikácie. Každý vzor je reprezentovaný binárnou šablónou, ktorá je porovnávaná so segmentovaným objektom pričom sa ukladá úroveň ich zhody. Toto porovnávanie je však veľmi pomalé nakoľko je nutné porovnať objekt s každým vzorom pixel po pixely. Pre urýchlenie tohto procesu môžeme rozdeliť šablóny na podšablóny. Ak sa nezhoduje jedna podšablóna ostatné sa už neporovnávajú. Ďalšou nevýhodou je nízka flexibilita ak obraz nezodpovedá vzoru.

7.2 Metóda najbližšieho suseda

Pre každú triedu je daná množina vzorových obrazov. Vektor príznakov týchto obrazov reprezentujeme v n–rozmernom priestore do ktorého umiestnime vektor príznakov neznámeho objektu. Pri klasifikácií hľadáme hľadáme najbližšieho respektíve najpodobnejšieho suseda. Modifikáciou tejto metódy je metóda *k–najbližších susedov*, kedy hľadáme k najbližších susedov. Ako triedu neznámeho objektu zvolíme tú, ktorá má väčšinu z týchto obrazov. Problémom môže byť samotná voľba k.

7.3 Adaboost

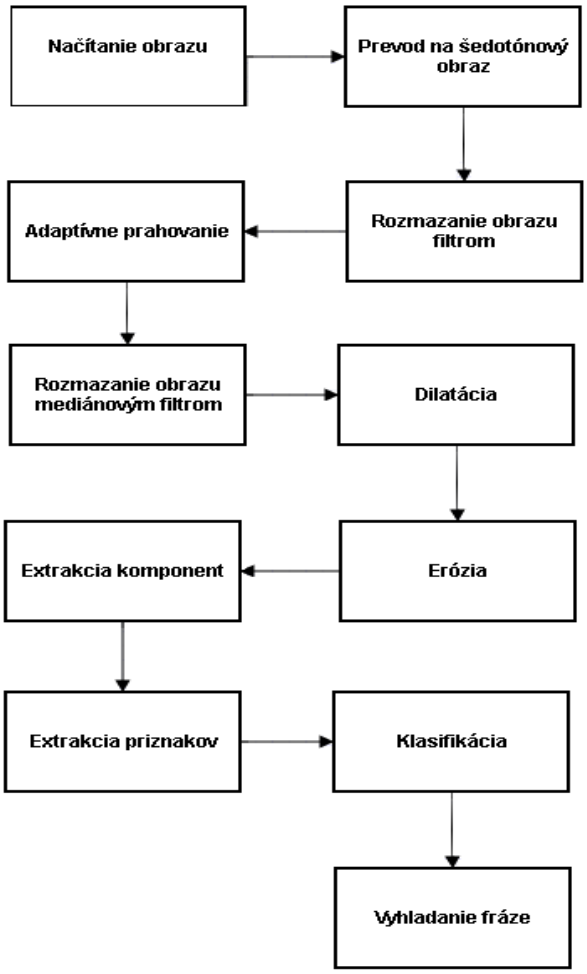
Adaboost je učiaci sa algoritmus, ktorý sa snaží dosiahnuť lepšie výsledky klasifikácie kombináciou viacerých jednoduchých klasifikátorov. Najskôr nájdeme najlepší klasifikátor. Následne zvýšime váhu zle klasifikovaných meraní a nájdeme klasifikátor ktorý najlepšie kalsifikuje chybné merania. Takto pokračujeme až pokým nezískame klasifikátor s požadovanou chybou.

8 Implementácia programu

Cieľom tejto práce bolo vytvorenie programu, ktorý vyhľadáva frázy v obrazový dátach typu JPEG. Program obsahuje dve základne funkcionality a to :

- Vyhľadávanie fráz
- Naučenie fontu.

Celý postup programu pri rozpoznavani fráz je zobrazený na obráku 10.



Obr. 10: Graf behu programu

8.1 Vyhľadávanie fráz

8.1.1 Načítanie obrazu a prevod na šedotónový obraz

Prvým krokom je samozrejme načítanie JPEG obraz. Tento obraz je uchovaný v objekte triedy **Picture**. Pri vytváraní objektu tejto triedy sa obraz prevedie na šedotónový pomocou vzroca 2 a uloží sa vo forme dvojrozmerného poľa. Z triedy **Configuration**, ktorá slúži ako konfiguračná trieda, je vybraná najväčšia maska a jej hodnota sa použije ako dvojnásobok veľkosti okraja. Je to z toho dôvodu aby sme pri každom algoritme, ktorý pracuje s obrazom nemuseli riešiť problematiku okrajov. Táto trieda taktiež uchováva komponenty, ktoré boli v tomto obraze detegované. Komponenta predstavuje jeden znak detegovaný v obraze.

8.1.2 Rozmazanie filtrom

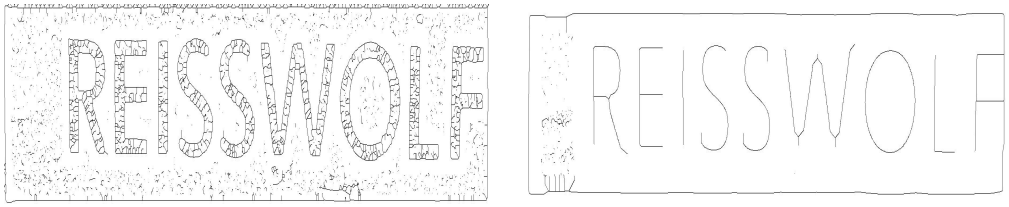
V ďalšom kroku je na obraz aplikovaný filter. Filtráciu poskytuje trieda **Filter**. Predvolený je Gaussov filter avšak uživateľ má možnosť vybrať si medzi priemerovacím, mediánovým a Gaussovým. Taktiež je možné tieto filtre nastaviť a teda určiť veľkosť masky. Samozrejme s narastajúcou veľkosťou masky rastie aj časová náročnosť. Výsledok po aplikáci Gaussovho filtra s maskou 20 pixelov a sigmou 1.8 môžeme vidieť na obrázku 11c.

(a) Originálny obrázok. (b) Obrázok po nahratí (c) Obrázok po aplikáci Gaussovho filtra

Obr. 11: Porovnanie originálneho, šedotónového a rozmazaného obrazu

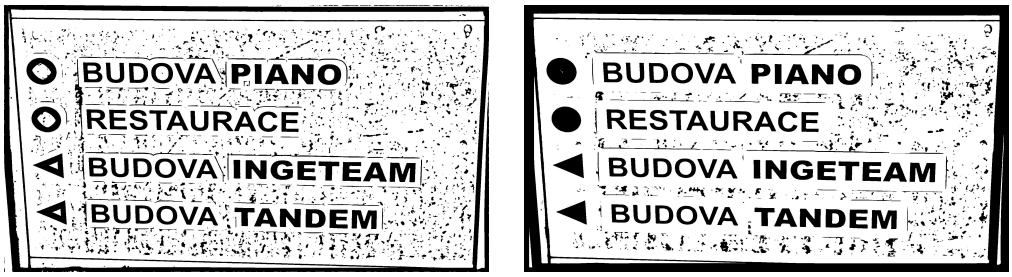
8.1.3 Adaptívne prahovanie

Po filtráci aplikujem segmentačnú metódu. Segmentačné metódy poskytuje trieda **Segmentation**. Táto metóda poskytuje všetky segmentačné metódy spomínané v kapitole 5. V tejto práci bolo zvolené adaptívne prahovanie, ktoré bolo opísané v kapitole 5.1. Pomocou Otsuho metódy opísanej v časti 5.1.2 počítame hodnotu prahu pre daný pixel z jeho okolia určeného veľkosťou masky. Problémom môže byť voľba masky, kde veľká maska dosahuje zlé výsledky pri malom (s malým počtom pixelov) písme. Naopak pri veľkom písme je vhodnejšie nakoľko, nevznikajú v strede písma biele miesta. Tieto miesta spôsobujú nesprávnu klasifikáciu nakoľko výsledky po erózi sú rozdielne. Tento rozdiel je možné vidieť na obrázku 12. Aj v tomo prípade má uživateľ možnosť nastavenia veľkosti masky. V prípade, že zvolí veľkosť masky 0, je vypočítaný iba jeden prah pre celý obraz. Na obrázku 13 môžete vidieť výsledky pre rôzne masky.



(a) Veľkosť masky prahu = 25 (b) Veľkosť masky prahu = 100

Obr. 12: Výsledky erózie

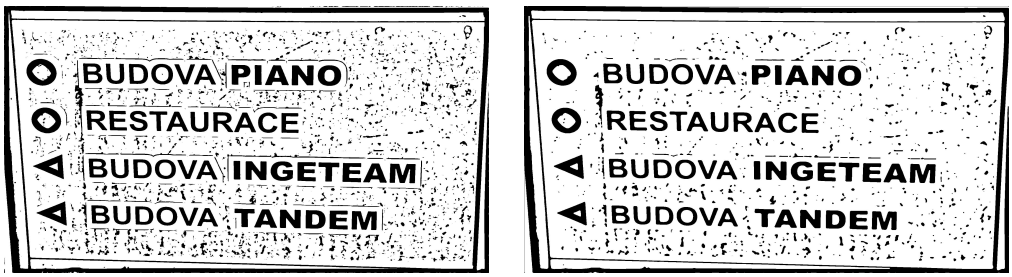


(a) Veľkosť masky = 50 (b) Veľkosť masky = 100

Obr. 13: Porovnanie výsledkov prahovanie pre rôzne masky

8.1.4 Rozmazanie mediánovým filtrom

Tento krok bol zvolený, kvôli potlačeniu malých zhlukov bodov. Na obrázku 14 môžete vidieť výledok pred a po mediánovej filtrácii. Čím väčšia je veľkosť masky tohoto filtra tým väčšie zhluky dokáže potlačiť. Problémom je, že táto filtrácia môže viesť k rozpájaniu prípadne spájaniu znakov a teda nesprávnej tvorbe komponent a ich následnej chybnej klasifikácii.



(a) Pred mediánovou filtráciou (b) Po mediánovej filtrácii s maskou 5x5

Obr. 14: Porovnanie výsledkov pred a po mediánovej filtrácii

8.1.5 Dilatácia a Erózia

Pomocou dilatácie je možné spojiť rozpojené písmená, ktoré vznikli pri mediánovej filtrácii. Avšak pri väčších dilatačnch maskách dochádza k nechcenému spájaniu komponent.

Následnou eróziou zredukujeme hrúbku písma na jeden pixel. Erózia bola pridaná z toho dôvodu, že vďaka nej sme shopný rozpoznať aj tučné písmo pomocou jednoduchého nakoľko po aplikácií erózie je výsledok pre vektor vzdialenosti veľmi podobný.

Pre túto prácu bol zvolený **Zhang-Suenov** algoritmus.

Definujme najskôr okolie skúmaného bodu X ako

$$\begin{bmatrix} H & A & B \\ G & X & C \\ F & E & D \end{bmatrix}$$

ďalej definujme

- $Z(X)$ ako počet zmien farby pixelu v poradí A->B->C->D->E->F->G->H->A.
- $P(X)$ ako počet susedných pixelov objektu.

Následne algoritmus prejde všetky pixely obrazu v dvoch krokoch. Ak v prvom kroku niektorý z pixelov spňuje nasledujúce podmienky je označený ako adept na zmazanie.

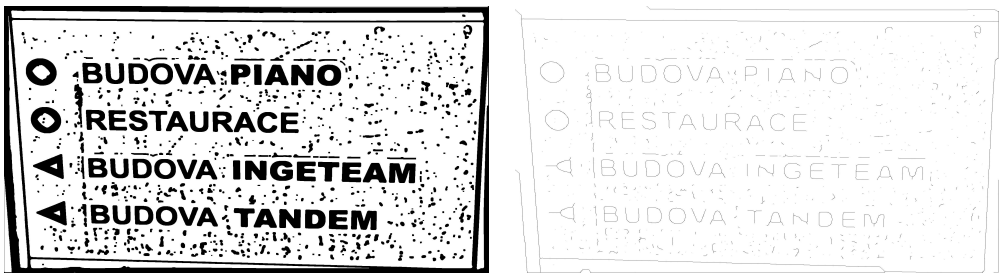
1. Pixel je pixelom objektu.
2. $2 \leq P(X) \leq 6$.
3. $Z(X) = 1$.
4. Aspoň jeden z pixelov A, C alebo E je pixel pozadia.
5. Aspoň jeden z pixelov C, E alebo G je pixel pozadia.

V druhom krouku sa znova prechádzajú vsetky pixely obrazu a testujú sa na nasledujúce podmienky :

1. Pixel je pixelom objektu.
2. $2 \leq P(X) \leq 6$.
3. $Z(X) = 1$.
4. Aspoň jeden z pixelov A, C alebo G je pixel pozadia.
5. Aspoň jeden z pixelov A, E alebo G je pixel pozadia.

Ak sú všetky splnené je pixel označený za adepta na zmazanie. Po vykonaní sa z obrazu odstránia adepti na zmazanie. Algoritmus skončí ak v ani prvom ani druhom kroku nie je žiaden pixel označený ako adept na zmazanie.

Obe tieto metódy poskytuje trieda **MorphologicalOperation**. Výsledky po dilatáci s veľkosťou masky 5 a erózi je možné vidieť na obrázku 15.



(a) Dilatácia s maskou 5 (b) Erózia

Obr. 15: Výsledok dilatácie a erózie.

8.1.6 Extrakcia komponent a príznakov

Ako už bolo spomenuté trieda **Picture** si udržiava zoznam svojích komponent. Tie sú extrahované algoritmom farbenia oblasti, ktorý poskytuje trieda **ComponentExtractor**.

Tento algoritmus prebieha v dvoch krokoch. V prvom sa každý pixel objektu testuje na nasledujúce podmienky :

- Ak sa v okolí nenachádza označený pixelm označí sa skúmaný pixel novou značkou
- Ak sa v okolí nachádza práve jeden označený pixel, označí sa skúmaný pixel jeho značkou
- Ak sa v okolí nachádza viac ako jeden označený pixel, vytvorí sa záznam o ekvivalencií.

V tomto algoritme predstavuje záznam ekvivalencií informáciu o rodičovi. Každý nový box je svojim rodičom. V prípade, že je nutné nastaviť ekvivalenciu dvoch boxov, porovnajú sa ich rodičia. Boxu s väčším rodičom je nastavený menší rodič.

V druhom kroku sa každý pixel nastaví na hodnotu svojho rodiča a pomocou súradníc pixelov sa vypočítajú začiatočné súradnice komponenty, jej vyška a širka. Pri prevode objektov typu **Box** na objekty typu **Component** sú odstránené tie komponenty, ktoré nemajú minimálnu veľkosť, prípadne presahujú maximálnu veľkosť. Tieto veľkosti môže opäť nastaviť uživateľ v nastaveniach systému.

Po vytvorení komponent sú pre ne vypočítané príznaky. Tento výpočet je realizovaný pomocou triedy **Extractor**. Momentálne sú dostupné Huové momenty a vektor vzdialenosti. Tieto príznaky boli zvolené pre ich invariantnosť, avšak neskôr sa u Huových momentoch ukázala slabá invariantnosť voči zmene mierky. Z tohto dôvodu bola klasifikácia na ich základe vypnutá.

Uživateľ si však tieto príznaky môže povoliť v nastaveniach systému. Ďalšie dostupné príznaky sú vektory vzdialeností.

8.1.7 Klasifikácia a vyhľadanie fráz

Vďaka získaným extraktorom môžeme pristúpiť ku klasifikácii. Systém má k dispozíci **kNN klasifikátor** poskytovaný triedou **Classifier**.

Po klasifikáci na základe uživateľom zvolených deskriptorov a hodnôt známych znakov je komponentam priradená hodnota. Zoznam a hodnoty známych znakov respektíve fontov sa nachádza v zložke Fonts v adresáry s aplikáciou.

Tieto fonty a ich hodnoty sú reprezentované pomocou xml súborov. Po klasifikáci sú komponenty zoradené do riadkov a slov. Toto zoraďovanie sa vykonáva na základe súradníc začiatku komponenty a teda má problémy s výraznejšie pootočeným textom, avšak pri odfotených dokumentoch sa nepredpokladá výraznejšie potočenie prípadne orientácia na šírku.

Po získaní textu a následnom zoradení sa pomocou Regexu vyhľadáva fŕaza. Regex je nastavený tak aby ignorolval ľubovoľný počet medzier prípadne odriadkovanie.

Nakoniec systém zobrazí iba tie obrazy, ktoré obsahujú hľadanú frázu.

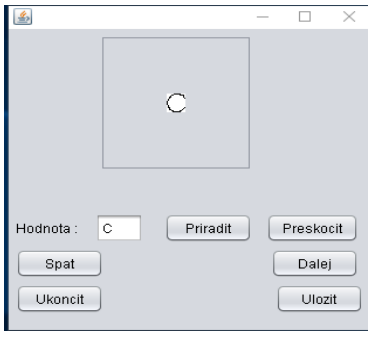
8.2 Naučenie fontu

Učenie fontu prebieha za asistencie uživateľa. Ten najskôr predloží zložku v ktorej sa nachádza obraz s fontom vo formáte JPEG. Následne systém vykoná primitívnu binarizáciu s hodnotou prahu 127. Nasleduje erózia a extrakcia komponent, ktoré boli popísané v predchádzajúcej časti. Následne sú pre kažú komponentu vypočítané všetky známe deskriptory.

V prípade že je detegovaných 62 komponent pokúsi sa systém priradiť hodnoty sám.

Uživaťel môže po extrakci prechádzať medzi jednotlivými komponentmai a priradzovať im hodnotu ako je vidieť na obrázku 16. V prípade, že bola extrahovaná komponenta, ktorá neobsahuje požadovaný znak, môže užívateľ túto komponentu preskočiť, prípadne nevyplniť hodnotu komponenty a pri ukladaní fontu bude vynechaná.

Po uložení voľby je font uložený do zložky Fonts v adresáry s aplikáciou pod názvom JPEG obrazu s fontom.



Obr. 16: Príklad priradenia hodnoty komponente.

9 Výsledky

Systém bol testovaný na testovacej množine 100 obrázkov.

Ako prvá bola testovaná segmentácia obrazu. Obrazy boli rozdelené na vhodné a nevhodné pre ďalšie spracovanie. Za nevhodný obraz bol považovaný ten, ktorý obsahoval väčšie množstvo rozpojených znakov, väčšsie množstvo spojených znakov alebo obsahoval nevhodný znak.

Pri testovaní sa meral čas segmentácie a uloženia obrazu. Výsledky testu sú uvedené v tabuľke 1. Sĺpecprírastok predstavuje počet nevhodných obrazov, ktoré v predchadzajúcom teste neboli označené ako nevhodné.

Pri tomto testovaní sa použili tri konfigurácie systému. Tieto konfigurácie sú uvedené v tabuľke 2.

Tabuľka 1: Výsledky testov.

Test	Čas na obrázok [min]	Počet nevhodných obrázkov	Prírastok
1	0.642	61	–
2	0.568	43	+6
3	2.75	11	+1

Tabuľka 2: Konfigurácia testov segmentácie.

Test	Veľkosť mediánovej masky	Veľkosť segmentačnej masky	Veľkosť dilatačnej masky
1	10	25	5
2	0	25	0
3	0	100	0

Z výsledkov je jasné, aký výrazný vplyv má veľkosť segmentačnej masky na časovú náročnost segmentácie ako aj na samotný výsledok segmentácie.

Rozdiely medzi prvým a druhým testom sú spôsobené dilatáciou a veľkosťou mediánvej filtrácie. Tieto parametre ovplyvňovali výsledky najmä pri menšom texte, kde jednotlivé znaky boli pomerné blízko seba.

Ďalší test bol zameraný na samotné rozpoznávanie. Do systému boli vložené fonty priamo z obrázkov. Pri teste na prvom obraze sa ukázala iba 50% úspešnosť klasifikácie.

Pri bližšom preskúmaní za zistila nízka diskriminabilita príznakov. Systém mal problém s rozpoznávaním písmen *I J L*, *O D 0* a *E P B*. Taktiež sa ukázalo, že sytém klasifikuje aj náhodné zhluky pixelov, ktoré neboli odstránené filtráciou. Posledný menovaný problém je možné odstrániť zmenou nastavenia systému, avšak sa tým poruší jedna z požiadavok na systém, a to čo najnižšia interakcia s uživateľom.

10 Záver

Cieľom tejto práce bolo vytvorenie systému schopného vyhľadávať fŕazy v netextových súboroch formátu JPEG. Ďalšimi požiadavkami bola implementácia tohoto systému v jazyku Java a čo najnižšia požadovaná interakcia od uživateľa.

Celý systém je rozdelený na samostatné moduly, kde každý modul predstavuje jednotlivú časť systému. Použitie týchto modulov je predstavené v časti venovanej implementáci systému.

V úvodnej časti systému sa prevádza farebný obraz na šedotónový. Následne sa vykoná filtrácia pre odstránenie šumu. Uživatel má možnosť upraviť jednotlivé parametre filtrácie ako aj zvoliť filter, ktorý ma byť použitý. Po filtráci sa obraz pomocou adaptívneho prahovania prevedie na binárny. Aj v tomto prípade má uživatel možnosť zvoliť veľkosť okolia, podľa ktorého sa vypočíta práh. Práh je vypočítaný pomocou Otshuho metódy. Následná mediánová filtrácia má za úlohu odstrániť zhluky pixelov, ktoré vznikli vplivom šumu. Aj v tomto prípade má uživateľ možnosť zmeniť veľkosť masky, prípadne túto možnosť vypnúť, nakoľko v niekorých prípadoch môže táto fitrácia spôsobiť rozpojenie alebo zliatie komponent. Na segmentovaný obraz sú aplikované morfologické operácie a to dilatácia a erózia. Voľba veľkosti dilatačnej masky, prípadne jej vypnutie je možné z nastavení systému. Ďalšou fázou je exktrakcia komponent pomocou farbenia oblasti. Pre každú komponentu je vypočítaný vektor príznakov. Ako príznaky boli zvolené Huove momenty. Pri neskoršom testovaní však vykazovali nízku invariantnosť voči zmene mierky a preto bol do systému pridaný vektor príznakov.

Tento vektor sa však ukázal ako málo diskriminabilný, čo spôsobovalo chybnú detekciu skupín písmen. Následkom tohoto problému systém nebol schopný korektne klasifikovať text, čo spôsobilo nízku uspešnosť systému pri vyhľadávaní fráz.

Aj v tomto prípade má uživateľ možnosť voľby príznakov, podľa ktorých sa vykoná následná klasifikácia. Ako klasifikátor bol zvolený kNN klasifikátor.

Systém by mohol byť rožšírený o algoritmus rozpájania spojených znakov čo by zvýšilo úspešnosť segmentácie. Prípadne rožšírenie skúmania rozloženia dokumentu by odstránilo problémy pri rotáci textu.

V budúcej práci by som sa rád venoval práve odstráneniu spomínaných problémov a rožšíreniu o spomínané funkcie ako aj rožšíreniu použitelnosti systému pre extrakciu textu z videa.

Pre systém bola vypracovaná uživatelská príručka a programatorská dokumentácia, ktoré sú súčasťou prílohy.

Literatura

[1] *SANKARAN N., JAWAHAR, C.V.* **Recognitionof Printed Devanagari Text Using BLSTM Neural Network**
IEEE, 2012

[2] *SUMETPHONG, CHAIVATNA, TANGWONGSAN, T.* **An Optimal Approach towards Recognizing Broken Thai Characters in OCR Systems.**
Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on. IEEE, 2012.

[3] *SALMAN, A., MALIK, A., a spol* **A novel approach for Braille images segmentation**
Multimedia Computing and Systems (ICMCS), 2012 International Conference on. IEEE, 2012.

[4] *CHAVRE P., GHOTKAR, A.* **Scene text extraction using stroke width transform for tourist translator on android platform**
International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), Pune, 2016, pp. 301-306.

[5] *URBANEC, Tomáš.* **Rozpoznávání ručně psaného textu pomocí fuzzy logiky**
Univerzita Palackého v Olomouci, Přírodovědecká fakulta, 2013
`http://theses.cz/id/hggjvv/`

[6] *HORÁK, Karel, KALOVÁ, Ilona, PETYOVSKÝ, Petr, RICHTER, Miroslav.* **Počítačové vidění**
Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008
`http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/Pocitacove_videni_S.pdf`

[7] *DOBEŠ, Michal.* **Zpracování obrazu a algoritmy v C#**
Praha: BEN - technická literatura, 2008
ISBN 978-80-7300-233-6.

[8] *MARR, D. HILDERTH, E. :* **Theory of edge detection**
Proc. Royal Soc. Lond., volume B 207, str.: 187-217, 1980.

[9] *ŽELEZNÝ, Miloš. :* **Zpracování digitalizovaného obrazu : Učební text (přednášky)**
Západočeská univerita v Plzni, Fakulta aplikovaných věd
`http://www.kky.zcu.cz/uploads/courses/zdo/ZDO_aktual_130215.pdf`

[10] *ŠIKUDOVÁ, E., ČERNEKOVÁ, Z., BENEŠOVÁ, W., HALADOVÁ, Z., KUČEROVÁ, J.* **Počítačové videnie Detekcia a rozpoznávanie objektov.**
Praha: Wikina Praha, 2013, p. 397.

Zoznam príloh

Prílohy k tejto práci sú uložené na priloženom CD. Jedná sa o :

1. Zdrojové kódy systému
2. Uživatelská príručka pre systém
3. Programátorská príručka pre systém.