VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

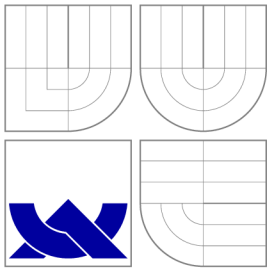
OPTICKÉ ROZPOZNÁVÁNÍ ZNAKŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

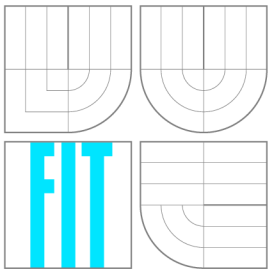
AUTOR PRÁCE
AUTHOR

PAVEL POKORNÝ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTICKÉ ROZPOZNÁVÁNÍ ZNAKŮ

OPTICAL CHARACTER RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL POKORNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOZEF MLÍCH

BRNO 2010

Abstrakt

V této práci jsou představeny některé z metod pro vyhledání a rozpoznání textu v obraze. Zabývá se problematikou extrakce příznaků a představuje nejčastěji používané algoritmy strojového učení. Popisuje postup při návrhu a implementaci aplikace určené k rozpoznávání tištěného textu a vytvoření datové sady znaků.

Abstract

This paper presents some of the methods for locating and recognizing text in an image document. It describes feature extraction issues and commonly used machine learning algorithms. In the latest part, there is description of design and implementation of application for printed text recognition.

Klíčová slova

Rozpoznávání a vyhledávání textu v obraze, strojové učení, klasifikace, výběr a extrakce příznaků.

Keywords

Optical character recognition, OCR, machine learning, classification, document image segmentation, feature extraction.

Citace

Pavel Pokorný: Optické rozpoznávání znaků, bakalářská práce, Brno, FIT VUT v Brně, 2010

Optické rozpoznávání znaků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha.

.....
Pavel Pokorný
18. května 2010

Poděkování

Děkuji vedoucímu této práce Ing. Jozefu Mlíchovi za poskytnuté konzultace, cenné rady a náměty.

Obsah

1 Úvod **3**

2 Zpracování obrazu **5**
2.1 Prahování 5
2.2 Detekce hran 6

3 Vyhledání a zpracování textu **7**
3.1 Korekce pootočení 7
3.2 Analýza rozvržení dokumentu 9
3.3 Extrakce příznaků 10

4 Klasifikace **12**
4.1 Rozhodovací stromy 13
4.2 Naive-Bayes 13
4.3 Support vector machines 14
4.4 K-nejbližších sousedů 15
4.5 Neuronové sítě 16

5 Implementace aplikace a vyhodnocení výsledků **17**
5.1 Binarizace 19
5.2 Vyrovnání 19
5.3 Segmentace 20
5.4 Výběr příznaků 21
5.5 Vytvoření datové sady 22
5.6 Dosažené výsledky 23
5.7 Návrhy možných vylepšení 26

6 Závěr **27**

Seznam obrázků

1.1 Běžné součásti systému OCR podle [10]. 4

3.1 Projekce bodů do parametrického prostoru. Zdroj [8]. 8

4.1 Ukázka rozhodovacího stromu. Zdroj [9]. 13
4.2 Princip SVM. Zdroj [6]. 14
4.3 Projekce dat do vícerozměrného prostoru. Zdroj [6]. 15

5.1 Kontroly vhodnosti provedené binarizace. 19
5.2 Použití Houghovy transformace. 20
5.3 Ukázka správné segmentace znaků. 21
5.4 Postup při zpracování nalezeného znaku. 22
5.5 Rozmanitost datové sady. 22
5.6 Část testovaného dokumentu 23
5.7 Vyhodnocení úspěšnosti rozpoznání jedné z testovacích sad. 25
5.8 Originál rozpoznávaného textu. 25

Kapitola 1

Úvod

Optické rozpoznávání znaků, označováno zkratkou OCR z anglického *optical character recognition*, je proces převodu naskenovaného obrázku s textem do podoby textového dokumentu. Může se jednat o text psaný ručně, na psacím stroji nebo tištěný. Často se tento převod provádí z důvodu lepší skladnosti takového textu a prakticky neomezené délce života nebo kvůli lepším možnostem šíření (přes internet, na přenosných médiích).

V textu získaném pomocí OCR lze jednoduše vyhledávat, překládat ho pomocí automatických překladových slovníků, převádět jej na počítačem mluvené slovo a mnoho dalších věcí, jejichž provedení by bez této metody nebo ručního přepisu textu nebylo možné.

Důvodů pro vývoj je tedy mnoho, o čemž ostatně svědčí i to, že probíhá již skoro celé jedno století a stále nebyl nalezen postup k dosažení 100% výsledků. Jedná se o složitý proces, v průběhu kterého se může vyskytnout mnoho problémů. O některých z nich a jejich možných řešení pojednává tato práce. Výzkum optického rozpoznávání znaků zasahuje do oblasti rozpoznávání vzorů (*pattern recognition*), umělé inteligence (*artificial intelligence*) a počítačového vidění (*computer vision*).

Tato bakalářská práce se zabývá rozpoznáváním tištěného textu v obraze. Kapitoly 2, 3 a 4 pojednávají o běžně používaných metodách v této oblasti. Součástí této práce je aplikace sloužící k segmentaci znaků z obrazu a jejich rozpoznávání. Její návrh a popis implementace je uveden v kapitole 5. S pomocí této aplikace byla vytvořena trénovací sada znaků, podrobněji v sekci 5.5. Dosažené výsledky a návrhy možných vylepšení aplikace jsou uvedeny v 5.6.

Cílem OCR je rozpoznat s co největší přesností všechny znaky v dodaném obraze, čehož je i v dnešní době těžké dosáhnout, zejména kvůli velkému počtu různých fontů, nežádoucím objektům v obraze (šum), nepřesnosti tiskáren atd. Účinnost metod je závislá na vstupních datech, proto je důležité vhodně zvolit metody zpracování obrazu a rozpoznávací algoritmus.

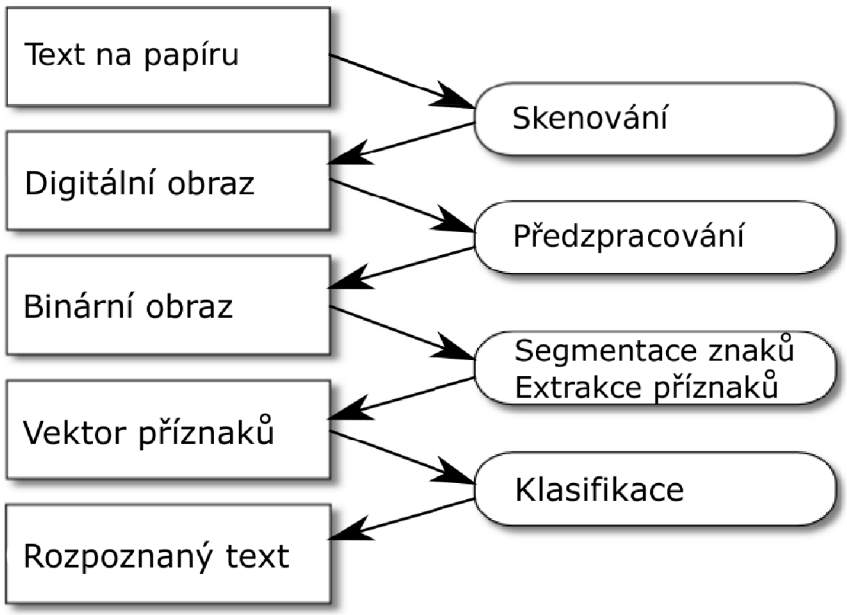
Průběh rozpoznávání textu lze rozložit na několik samostatných částí (obr. 1.1).

Nejdříve je třeba získat obraz textu pomocí skeneru nebo fotoaparátu. Takto získaný obraz obsahuje informace o zobrazeném textu, ale také spoustu dalších informací o pozadí, intenzitě osvětlení, nečistotám na papíru atp. Přebytečným elementům v obraze se říká obrazový spam, šum. Odstraněním obrazového spamu se zabývá kapitola 2.

V upraveném obrazu se vzápětí vyhledají bloky textu, řádky a jednotlivé znaky (segmentace). Získané znaky mohou být poté zpracovány do normované podoby a následně je provedena extrakce příznaků. Zvolené příznaky mají velký vliv na úspěšnost celého systému a opět nelze říci, že některý z přístupu je ideální pro všechna vstupní data. Těmto částem se věnuje kapitola 3.

Posledním krokem je klasifikace znaků na základě vektoru příznaků získaných z jednot-

livých znaků. Existuje celá řada klasifikátorů. Většina z nich se musí předem natrénovat na sadě příkladů se správně určenou třídou, tak vznikne model obsahující obecnou charakteristiku jednotlivých tříd. S obecnými popisy tříd v tomto modelu jsou poté porovnávány příznaky jednotlivých neznámých znaků a vybere se třída s největší podobností. Klasifikací se zabývá kapitola 4.



Obrázek 1.1: Běžné součásti systému OCR podle [10].

Rozpoznaný text je poté možno dále analyzovat. Provádí se například vyhledávání rozpoznaných slov v obsáhlém slovníku podle zvoleného nebo automaticky rozpoznaného jazyka a na základě nejpodobnějšího nalezeného slova se mohou opravit některé špatně rozpoznané znaky. Analýze rozpoznaného textu se tato práce dále nevěnuje.

Kapitola 2

Zpracování obrazu

Prvním krokem v procesu získání textu z obrazu je dostat vytištěný text do digitální podoby. Nejčastěji se k tomuto účelu používá optický scanner nebo fotoaparát. Takto získané obrazy jsou v počítači uloženy jako popis jednotlivých bodů (pixelů). Počet bodů (rozlišení) se liší podle kvality použitého zařízení a taktéž informace obsažené v popisu jednoho pixelu mohou být různé vzhledem k použitému formátu obrazu (stupně šedi nebo některá z reprezentace barev).

Pro další zpracování je žádoucí, aby obrázek obsahoval informace pouze o zobrazeném textu. Při OCR jsou proto důležité hrany písmen a naopak není potřebná např. informace o barvě a šum (spam) na pozadí, který vzniká při nedostatečném osvětlení, použití nekvalitního papíru nebo zařízení pro převod do digitální podoby nebo při jakékoliv deformaci skenovaného dokumentu. Šumu je možné se zbavit např. *prahováním* nebo detekcí hran a prováděním dalších operací pouze na těchto hranách.

2.1 Prahování

Prahování je převod obrázku ze stupní šedi do binární podoby (černobílý obraz), kde každý pixel obsahuje jen informaci černá/bílá.

Globální prahování

Algoritmus je jednoduchý, sekvenčně prochází všechny pixely obrazu a porovnává je s pevně určeným prahem (např. průměr intenzit všech pixelů). Pokud je hodnota pixelu větší (resp. menší) než prah, hodnota bodu se nastaví na maximum (resp. minimum). Hodnota prahu je jediný parametr této metody.

Na obrazy, které nebyly při pořizování rovnoměrně osvětleny, je tato metoda často nepoužitelná. Mohlo by se totiž stát, že odstín všech bodů v některých částech by byl pod nebo nad daným prahem a celá tato oblast by byla chybně zpracována. V takových případech je vhodné použít adaptivního prahovaní.

Globální prah lze určovat ručně nebo je možné použít některý ze způsobů nalezení optimálního prahu. Jednou z těchto metod je Otsuho metoda, při které je prah vybrán na základě rozptylu jednotlivých intenzit. Algoritmus hledá největší rozptyl mezi pozadím a popředím (*between class variance*), který se poté použije pro globální prahování.

Adaptivní prahování

Při použití adaptivního prahování se dynamicky mění hodnota prahu. Je možné prah vypočítávat zvlášť pro každý pixel nebo pro části obrazu určité velkosti. Tato metoda se někdy nazývá také dynamické nebo lokální prahování.

Při výpočtu prahu pro každý pixel, se hodnota prahu určuje na základě okolních pixelů. Výsledný prah lze vypočítat několika způsoby, všechny vypočítávají nějakou statistiku z okolních pixelů. Mezi nejpoužívanější patří *střední hodnota*, *medián* nebo *průměr* z maximální a minimální hodnoty jasu z okolních bodů. Takto získaná hodnota prahu se pak použije u daného pixelu stejným způsobem jako u globálního prahování. Je nutné zvolit správnou velikost okolí, která by měla pokrýt dostatečné množství bodů pozadí i hledaných objektů. Pokud je prah určován pro větší části obrazu, počítají se tyto statistiky většinou ze všech pixelů této oblasti a zjištěný prah je použit pro celou tuto oblast.

V částech obrazu, kde se vyskytuje např. pouze pozadí, mohou mít sousední pixely velice podobné hodnoty a může docházet ke špatnému výpočtu hodnoty prahu. Proto se využívá přídavného globálního prahu, který určuje minimální intenzitu hledaných objektů.

2.2 Detekce hran

Při hledání objektů lze využít pouhé prahování, ale v praxi se často používají i tzv. detektory hran. Detekce hran je vyhledání takových bodů, ve kterých se výrazně mění intenzita (jas) obrazu. Takovýchto detektorů existuje velké množství, liší se v metodách hledání, jejich reprezentací, citlivosti rozpoznání či odolnosti proti šumu.

Jednoduché detektory jsou založeny na principu konvolučního filtrování obrazu a liší se pouze jádrem filtru (např. Sobel, Roberts, ...).

Cannyho detektor hran

Tento detektor nepatří mezi triviální, skládá se z několika kroků. Mezi hlavní přednosti patří velice malý počet chybně detekovaných hran, přesnost určení pozice detekované hrany a jednoznačnost hran, tzn. že každá hrana je detekována jen jednou. [11]

Algoritmus zahrnuje čtyři kroky:

- Odstranění šumu z obrazu pomocí Gaussova filtru
- Aplikace Sobelova filtru pro nalezení velikostí gradientů
- Ztenčování
- Prahování

Ztenčování

Ztenčování (*thinning*, *skeletonization*) je metoda pro redukci oblastí v obraze na pouhé úsečky (křivky) o tloušťce jednoho pixelu. Vytvoří se tak pouze jakási kostra a ztrácí se informace o tloušťce písma. Dochází ke zobecňování, což může přispět k větší účinnosti rozpoznání textu, protože se stírají rozdíly mezi různými druhy fontů. Algoritmy ztenčování prochází pixely obrazu a zkoumají jejich okolí. Velikost okolí pixelu se používá různá, nejčastěji tzv. osmi-okolí. Postupně se odstraňují okrajové pixely objektů dokud nezůstanou jen jedno-pixelové křivky.

Kapitola 3

Vyhledání a zpracování textu

Naskenované dokumenty mohou kromě textu obsahovat i jiný druh obsahu, nejčastěji různé tabulky a obrázky. Text často bývá členěn na odstavce, které mohou být vůči sobě v různých pozicích (jako v novinách) a není snadné rozlišit jednotlivé řádky. Text je tedy nutné nejprve v obraze vyhledat. Celý dokument může být nekvalitně naskenován a být pootočen, což by mohlo vést k potížím při jeho vyhledání. Vyrovnáním obrazu se zabývá sekce 3.1.

Nalezení důležitých objektů a správné oddělení od zbytku dokumentu se nazývá segmentace. V OCR je nutné najít v obraze odstavce textu, ve kterých je poté možné detekovat jednotlivé řádky, slova a nakonec i znaky. V sekci 3.2 jsou představeny možné způsoby vyhledání bloků textu. V kapitole věnující se implementaci programu je pak v sekci 5.3 vysvětlen jeden možný přístup k segmentaci řádků, slov a znaků.

Ke klasifikaci nalezených znaků je vhodné místo všech pixelů oblasti znaku použít pouze některé vlastnosti získané z těchto pixelů. Tyto vlastnosti se nazývají příznaky (*features*) a jejich získávání je tzv. extrakce příznaků. Výběrem vhodných příznaků a jejich extrakcí se zabývá sekce 3.3.

V této kapitole jsem čerpal ze zdrojů [11], [12], [1], [10], [7].

3.1 Korekce pootočení

Hledaný text může být v obraze pootočen, což by znepříjemnilo jeho zpracování. Je tedy výhodné obraz nejprve vyrovnat. Nejdříve je potřeba nalézt úhel, o který je text pootočen. Metod pro vyhledání úhlu pootočení existuje více.

Proječní profily

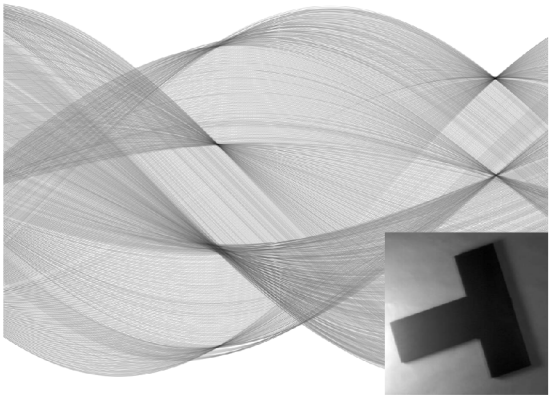
Metody založené na analýze profilu obrazu (*projection profile methods*) jsou intuitivní a při kvalitních datech také úspěšné. Profil obrazu je histogram z počtu černých pixelů, které jsou protnuty paralelně vedenými přímkami. Tyto přímky mohou být pod jakýmkoliv úhlem, ale nejčastěji se používají horizontální a vertikální přímky a vznikají tak horizontální a vertikální projekční profily. Možným použitím je výpočet profilů pod různými úhly a počítání odchylky mezi počty pixelů protnutých jednotlivými přímkami. Úhel s největší odchylkou je pravděpodobně hledaný úhel pootočení, protože při něm mnoho přímek protíná velké množství pixelů (řádky) a stejně tak hodně přímek neprotne pixel žádný (mezery mezi řádky).

Nejbližší sousedé

Přístup využívající shlukování nejbližších sousedů (*nearest-neighbours*) není na rozdíl od ostatních limitován velikostí úhlu pootočení. Tato metoda zahrnuje nejprve nalezení souvislých objektů (*connected component*), poté je zjištěn nejbližší soused každého objektu a je vypočítán úhel mezi centrálními body nejbližších sousedů. Úhel, který se vyskytuje nejčastěji, je poté určen jako úhel pootočení.

Houghova transformace

Jedná se o metodu pro vyhledání obecného tvaru v obraze. Podmínkou je znalost analytického popisu tohoto tvaru a tak se nejčastěji využívá k nalezení přímek, kružnic a podobných známých geometrických útvarů. Při vhodně nastavených parametrech nijak výrazně nevadí, když se v obraze tyto útvary vyskytují přerušované. Díky tomu je možné tuto metodu použít i pro vyhledání řádků textu a poté obrázek pootočit na základě průměrné hodnoty pootočení řádků.



Obrázek 3.1: Projekce bodů do parametrického prostoru. Zdroj [8].

Využívá se parametrizace, která reprezentuje přímku jako bod v polárním souřadnicovém systému. Převodem tohoto bodu zpět do Kartézské soustavy souřadnic je získán opět pouze bod, avšak tímto bodem prochází původní přímka a zároveň je kolmá na přímku, která vede od počátku souřadnicového systému k tomuto bodu. Používané parametry jsou ρ (vzdálenost bodu od počátku) a θ (úhel, který svírá osa x s přímkou spojující bod a počátek). Rovnice takové přímky je:

$$\rho = x * cos\theta + y * sin\theta \tag{3.1}$$

Samotná transformace pak probíhá tak, že souřadnice x a y každého nenulového bodu vstupního obrazu jsou dosazeny do rovnice 3.1. Za hodnotu θ jsou postupně dosazovány hodnoty z intervalu $\langle 0, 2\pi \rangle$ s určeným krokem a dopočítává se ρ. Tak v parametrickém prostoru vznikne pro každý vstupní bod jedna křivka, v případě úseček se vytváří sinusoidy. Pokud některé body v původním obraze leží na stejné přímce, jejich křivky v parametrickém prostoru se protínají nejčastěji v bodě představujícím parametry této přímky. Na obrázku 3.1 jsou zobrazeny křivky v parametrickém prostoru, které byly vytvořeny z bodů obrázku zobrazeného v pravém dolním rohu. Po převodu všech nenulových pixelů obrazu jsou v parametrickém prostoru vybrána lokální maxima představující hledané přímky a úhel

pootočení obrazu se může vypočítat jako průměrný úhel pootočení těchto přímek vůči ose x.

Rotace obrazu

Po nalezení správného úhlu je třeba obraz vyrovnat, to je možné pomocí operace pootočení (rotace), která patří mezi základní afinní geometrické transformace (dalšími jsou posunutí, změna měřítka, zkosení a operace vzniklé jejich skládáním). Při použití **homogenních souřadnic** mohou být tyto transformace reprezentovány jako násobení matic. Rotaci a změnu velikosti ve 2D lze reprezentovat jako násobení P maticí 2x2, posunutí však nikoli, proto se zavádí třetí (homogenní) souřadnice. Bod P v homogenních souřadnicích má souřadnice $[x, y, \omega]$ za předpokladu, že platí:

$$X = \frac{x}{\omega}; Y = \frac{y}{\omega}$$

Kde X a Y jsou souřadnice v původním 2D prostoru. Souřadnice ω se nazývá váha bodu. Často se volí $\omega = 1$. Po zvolení ω mají homogenní souřadnice hodnotu

$$P' = [\omega X, \omega Y, \omega]$$

Afinní transformace jsou v homogenních souřadnicích vyjádřeny vztahem $P' = P * A$, kde P je bodem, který transformujeme pomocí matice A. Jednotlivé operace se liší pouze transformační maticí. Pro otočení obrazu s textem se využívá geometrické transformace rotace. Otáčením bodu P kolem počátku soustavy souřadnic $O = [0, 0]$ o orientovaný úhel α je získán v 2D prostoru bod P' o souřadnicích

$$X' = Xcos\alpha - Ysin\alpha$$

$$Y' = Xsin\alpha + Ycos\alpha$$

Transformační matice pro otáčení v homogenních souřadnicích má tvar

$$A_R = \begin{bmatrix} cos\alpha & sin\alpha & 0 \\ -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nové souřadnice bodu pro pootočení obrazu se získají vynásobením homogenních souřadnic bodu touto transformační maticí.

3.2 Analýza rozvržení dokumentu

Analýzou obrazu dokumentu se rozumí získání informací o struktuře jeho obsahu. Počítač reprezentuje obraz jako množinu bodů a o jejich významu nemá jiné informace, než kde a jakou barvou je má zobrazit. Komerční systémy, které jsou dnes k dispozici, zvládají už rozpoznávání formulářů, tabulek, obrázků i odstavců textu na dobré úrovni.

O segmentaci se mluví ve dvou souvislostech. Pokud dokument obsahuje grafiku i text, je nutné nejprve oddělit bloky s různým typem obsahu a ty jsou dále zpracovávány odlišnými metodami. Jinou činností je poté segmentace textu, kdy jsou oddělovány sloupce, odstavce, slova a znaky, nebo grafiky, která zahrnuje oddělení symbolů (obdélníky, kružnice) a úseček. Obrázky (fotky) se v běžných dokumentech většinou nijak dále nesegmentují.

Detekce kontur

Zjištění výskytu znaků lze provést detekcí kontur, což znamená vyhledání vnějších hranic objektů. Z kontur je možné zjistit počet objektů v dokumentu, jejich pozice, velikost i tvar. Jakmile jsou rozpoznány všechny kontury, je možné přistoupit k rozpoznávání každé z nich nebo je na základě jejich vlastností shlukovat do větších regionů. Na vyhledávání kontur existují různé algoritmy, např. *Moore-Neighbor* nebo *Radial Sweep*, jedná se o velice jednoduché postupy, které určují jak po nalezení prvního bodu nějakého objektu vyhledat celou jeho okrajovou hranici. Smyslem je procházení sousedních okrajových pixelů v jednom směru dokud se nenarazí na počáteční pixel.

Connected component labeling

Jiným přístupem k vyhledávání objektů v obraze je označení spojitých objektů (*connected component labeling*, někdy region *coloring*). Jedná se o dvou-průchodový algoritmus, ve kterém jsou bodům přiřazovány třídy představující samostatné objekty.

V prvním průchodu se zkoumají všechny pixely, pro ty s hodnotou popředí se zjistí hodnoty sousedů vlevo a nahoře (tyto pixely již byly zpracovány) a pokud nemá žádný z nich hodnotu popředí, označí se aktuální bod novou třídou. Pokud má hodnotu popředí jen jeden pixel, označí se aktuální pixel stejnou třídou. Když je jich více než jeden, označí se některou z jejich tříd a všechny zjištěné třídy jsou vloženy do třídy ekvivalence (pro pozdější sloučení). V druhém průchodu proběhne nejdříve sloučení všech tříd z každé třídy ekvivalence a poté se všem označeným pixelům přiřadí konečná třída. Každá z tříd reprezentuje jeden nalezený objekt.

Detekce řádků a slov

Někdy je potřeba zjistit, kde se nalézají jednotlivé řádky a na nich detekovat hranice mezi slovy. Jednou z metod je využití vertikálního a horizontálního histogramu. Předpokladem je, že obraz není nikterak pootočený. Podobné metody lze využít ke zjištění pootočení obrazu (viz 3.1).

Stačí zjistit počet černých bodů ležících na jednotlivých řádcích. Řádky v obraze, které neobsahují žádné černé pixely (nebo výrazně méně než ostatní řádky) pak můžeme považovat za hranice mezi jednotlivými řádky textu.

Pro všechny řádky textu stačí pak opakovat skoro stejný postup pro získání hranic mezi slovy. Rozdíl je pouze v tom, že je třeba počítat počet černých pixelů ve všech sloupcích řádku.

3.3 Extrakce příznaků

Data získaná segmentací znaků často obsahují zbytečné údaje v poměru k potřebných informací. Proto je vhodnější data reprezentovat menší množinou jejich vlastností (vektor příznaků). Proces vytvoření takové množiny vlastností se nazývá extrakce příznaků (*feature extraction*). Pokud jsou vybrány správné vlastnosti, je možné tuto zjednodušenou reprezentaci použít v algoritmech strojového učení jako adekvátní náhradu za úplná data. Problematika extrakce příznaků je značně obsáhlá a stále se vyvíjející oblast.

Mohlo by se zdát, že nejjednodušším přístupem je žádné příznaky neextrahovat a použít všechny pixely z oblasti nalezeného znaku. Tento přístup má však jisté nevýhody. Analýza

velkého počtu proměnných (pixelů), které bychom získali, by mohla být paměťově a výpočetně náročná. Nebylo by možné se zbavit závisloti na orientaci a velikosti textu a bylo by nutné provádět dodatečné úpravy. Je možné, že s tak velkým počtem příznaků by klasifikátor špatně zobecňoval popis tříd a nedokázal by rozpoznávat data odlišná těm z trénovací sady.

Možným vylepšením je použití ztenčování nebo kontur. Tento způsob přináší hlavně úsporu paměti. Samotný objekt je ale stále reprezentován pouze množinou bodů, což může být nedostačující. Následující postupy mohou využívat všechny pixely znaku nebo právě jen jejich kontury či kostry. Jedná se pouze o výběr z mnoha možných příznaků, které se dají použít. Čerpáno hlavně z [10].

Zoning Jiným přístupem je sledování rozložení bodů objektu v různých zónách obrazu, také *zoning*. Počet zón i jejich rozmístění se v různých zdrojích liší a jejich výběr probíhá většinou na základě výsledků experimentů.

Template matching Možné je provádět pouze tzv. *template matching*, což je porovnávání znaku s vytvořenými šablonami znaků a výběr třídy na základě jejich podobností.

Crossings Dalším z možných příznaků je počet průsečíků zkoumaného objektu s předem danými úsečkami, v literatuře se vyskytuje tento přístup pod názvem *Crossings*. Počet i pozice úseček se opět u různých zdrojů liší.

Momenty Jedná se o statistické vlastnosti obrazu. Moment je určitý speciální vážený průměr intenzit pixelů obrazu, který by měl dostatečně odlišit rozdíly mezi obrazy. Existují i normalizované momenty, které jsou invariantní vůči měřítku. Dokonce lze vypočítat momenty invariantní k velikosti i otočení, např. Hu momenty, které jsou získány lineární kombinací normalizovaných centrálních momentů.

Strukturální analýza Mezi nejsložitější patří strukturální analýza. Jedná se o rozpoznávání úseček a křivek. Tato reprezentace objektu nese nejvíce informací a je také nejbližší tomu, jak by znak popsal člověk. Strukturální analýza má nejvyšší toleranci šumu, její implementace je ale velice složitá a výzkum v této oblasti stále probíhá.

Použité příznaky mají značný vliv na úspěšnost natrénování klasifikátoru a pozdější rozpoznávání. Ideální jsou takové příznaky, které v co největší míře popisují tvar znaku a dostatečně je od sebe odlišují. Přístupy se liší citlivostí na šum, deformaci znaků a také složitostí implementace a použití. Pro různá data mohou být vyhovující jiné příznaky. Nelze tedy s jistotou doporučit použití některého přístupu, musí se vybrat na základě výsledků experimentů.

Kapitola 4

Klasifikace

Klasifikace neboli rozpoznávání je proces zařazení neznámého objektu do určité třídy na základě dříve získaných vědomostí. Existuje celá řada klasifikátorů, jejichž účinnost je závislá na klasifikovaných datech. Nelze tedy tvrdit, že by některý z nich byl nejlepší a fungoval nejlépe bez ohledu na použitá data. Výběr klasifikátoru může někdy výrazně ovlivnit výsledky rozpoznávání, vždy je lepší vyzkoušet jich více a až podle výsledku se rozhodnout.

V této kapitole bylo čerpáno z [6], [9] a [1].

Při klasifikaci znaků je k dispozici popis znaku $z \in \mathbb{X}$ (kde $\mathbb{X}$ je prostor popisů všech znaků) a předem danou množinu tříd $\mathbb{C} = \{c_1, c_2, ..., c_j\}$. Třídy jsou někdy také nazývány kategorie nebo anglicky *labels*. Popisem znaku se rozumí množina všech pixelů, které ho tvoří, nebo vektor příznaků, které ho nějakým způsobem charakterizují. Obecně je prostor popisů znaků $\mathbb{X}$ nějaký typ více-rozměrného prostoru a třídy jsou stanoveny člověkem pro potřeby aplikace. V OCR systému třídy představují všechny možné znaky, které je třeba rozeznat.

Nejdříve je potřeba vytvořit nebo získat trénovací sadu $\mathbb{D}$ správně rozpoznaných znaků $\langle z, c\rangle$, kde $\langle z, c\rangle \in \mathbb{X} \times \mathbb{C}$. Například:

$$\langle z, c\rangle = \langle \text{příznaky získané ze znaku B}, B\rangle$$

Použitím strojového učení (*machine learning*) se snažíme naučit klasifikátor nebo klasifikační funkci γ, která mapuje znaky na třídy:

$$\gamma : \mathbb{X} \to \mathbb{C}$$

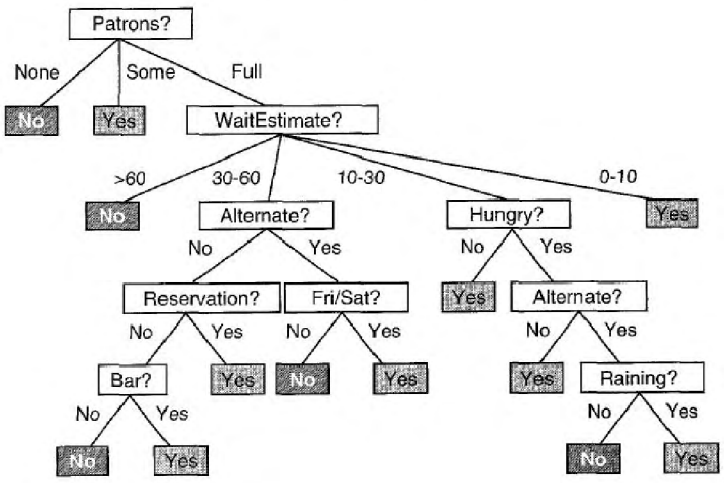
Tento způsob učení se nazývá **učení s učitelem** (*supervised learning*), protože učitel (člověk, který určuje třídy a tvoří testovací sadu) udává směr procesu učení. Pokud označíme metodu učení Γ, můžeme napsat $\Gamma(\mathbb{D}) = \gamma$. Metoda učení Γ přijímá jako vstup trénovací sadu $\mathbb{D}$ a vrací naučenou klasifikační funkci γ.

Většinou se jméno metody strojového učení Γ používá také pro klasifikátor γ. Lze tedy mluvit např. o SVM metodě i SVM klasifikátoru.

Přeučení Jedná se o jev, kdy je klasifikátor příliš přizpůsobený trénovací sadě. Na těchto datech má pak velkou úspěšnost rozpoznání, ale při rozpoznávání neznámých dat selhává. Přeučením se ztrácí schopnost klasifikátoru zobecňovat popis tříd.

4.1 Rozhodovací stromy

Rozhodovací stromy (*decision trees*) patří k nejstarším, ale velice úspěšným a používaným algoritmům. Vstupní atributy mohou být diskrétní (jedná se o klasifikaci) nebo spojité (pak je správné označení regrese - hledá se spojitá funkce). Spojité vstupy se v praxi často pro zjednodušení disktretizují do intervalů.



Obrázek 4.1: Ukázka rozhodovacího stromu. Zdroj [9].

Rozhodovací strom dosáhne svého rozhodnutí provedením posloupnosti testů. Každý uzel odpovídá testu hodnoty jednoho z atributů a větve vycházející z tohoto uzlu jsou popsány možnými hodnotami. Listy stromu pak obsahují hodnotu, která bude algoritmem vrácena, pokud bude listu dosaženo. Algoritmus je blízký lidskému rozhodování, samotný pohled na strom 4.1 nám prozradí, na základě kterých atributů se rozhoduje. Zobrazený rozhodovací strom řeší problém, zda čekat na uvolnění stolu v plné restauraci.

Důležitý při sestrojování stromu je výběr správného atributu, který se bude testovat v právě tvořeném uzlu stromu. Ideální vlastností pro testování je taková, která rozděluje trénovací data do dvou množin, ve kterých jsou prvky jen jedné třídy. Čerpáno z [9].

Existuje mnoho rozšíření rozhodovacích stromů, jedním z nich sou náhodné stromy (random trees, random forests). Jedná se kolekci více rozhodovacích stromů. Při rozpoznávání nového objektu se jeho příznaky vyhodnotí všemi stromy a každý z nich určí třídu a tím pro ni „hlasuje“. Výslednou třídou je pak ta, která získala nejvíce hlasů. [2]

4.2 Naive-Bayes

Naive Bayes je metoda pracující s pravděpodobností. Pravděpodobnost, že objekt (znak) d patří do třídy c je počítána jako

$$P(c|d) \propto P(c) \prod_{1 \leq k \ \leq n_d} P(t_k|c) \tag{4.1}$$

kde $P(t_k|c)$ je podmíněná pravděpodobnost výskytu atributu t_k u znaku třídy c. Metoda předpokládá vzájemnou nezávislost hodnot atributů (proto „naive“, naivní). Cílem je najít

nejvhodnější třídu pro zkoumaný znak. Při klasifikaci pomocí NB je to třída s největší pravděpodobností (maximum a posteriori, MAP). Čerpáno z [6].

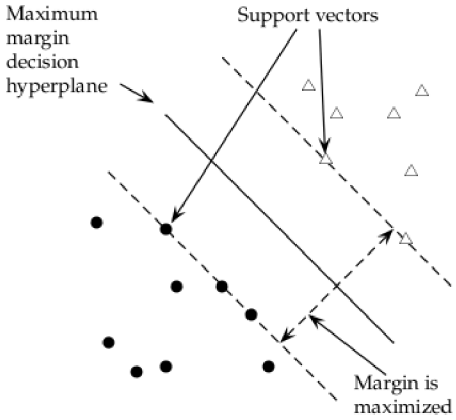
$$c_{map} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \ \leq n_d} \hat{P}(t_k|c) \tag{4.2}$$

4.3 Support vector machines

Cílem této metody je najít nadrovinu, která v prostoru příznaků optimálně rozděluje trénovací data. Ne vždy jsou však příznaky lineárně separovatelné. Důležitou součástí SVM je transformace prostoru příznaků do vícerozměrného prostoru, čehož je dosaženo pomocí tvorbou nových dimenzí pomocí kombinace příznaků. Pomocí dostatečného počtu přidaných dimenzí lze skoro vždy dosáhnout dokonalého rozdělení tříd. Tato metoda podává většinou velice dobré výsledky i při omezené velikosti trénovací sady.

Jedná se binární metodu, pomocí které se mohou rozdělit data dvou tříd. Pro klasifikaci více tříd je možné využít více způsobů. Mezi ty jednodušší patří vytvoření $|\mathbb{C}|$ jeden-proti-všem (*one-versus-all*) klasifikátorů. Pokud je znak schválen více klasifikátory, vybere se ta třída, jejíž dělící nadrovina je od tohoto znaku nejdále.

Lineární SVM



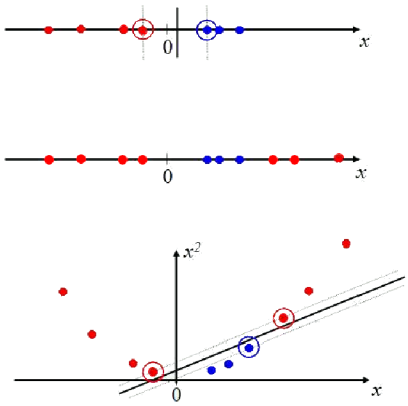
Obrázek 4.2: Princip SVM. Zdroj [6].

Trénováním SVM se hledá optimální nadrovina, což je ta, která je nejdále od jakýchkoliv datových bodů. Vzdálenost od dělící roviny k nejbližším bodům určuje margin klasifikátoru. Rozhodovací funkce je tedy dána jen (většinou malou) podmnožinou dat, která určuje pozici dělící roviny. Tyto nejdůležitější body se nazývají podpůrné vektory (*support vectors*). Obrázek 4.2 zobrazuje *margin* a podpůrné vektory na jednoduchém ukázkovém problému. Podpůrnými vektory je pětice bodů ležící na hranicích klasifikátoru, cílem je maximalizovat *margin* a oddělit data rovinou.

Nelineární SVM

Data na obrázku 4.2 jsou lineárně separovatelná a nalezení dělící roviny proto není velkým problémem. V případě, že data nedovolují klasifikaci lineárním klasifikátorem, je situace

o něco složitější. Jedním možným řešením, které SVM využívá, je namapování dat do vícerozměrného prostoru. Na obrázku 4.3 jsou zobrazeny tři případy. První dva jsou v jednodimenzionálním prostoru, kde první případ je lineárně separovatelný a druhý nikoliv (třídy jsou barevně odlišeny). Ve spodní části obrázku jsou zobrazena data z druhého případu, ale jsou namapována do dvourozměrného prostoru, ve kterém je již možné je rozdělit lineárním separátorem (přímkou). Projekce lineárně neoddělitelných dat do vícerozměrného prostoru je tedy může udělat lineárně oddělitelnými. Obrázek převzat z [6].



Obrázek 4.3: Projekce dat do vícerozměrného prostoru. Zdroj [6].

Pro transformaci dat do více dimenzí existují metody, kterým se souhrnně říká jádrová transformace (*kernel trick*). Jedná se o výpočet souřadnic v novém více-dimenzionálním prostoru na základě původních souřadnic.

4.4 K-nejbližších sousedů

K nearest neighbor (kNN) patří mezi nejjednodušší klasifikační metody. V modelu jsou uloženy všechny trénovací data a při klasifikaci se pouze zjistí, jaké třídy je k nejbližších (nejpodobnějších) prvků a ten nový je označen třídou, která je nejvíce rozšířena u jeho sousedů. Při velkém počtu testovacích dat, může být tato metoda pomalejší a paměťově náročná.

Pseudokód 4.1: Natrénování kNN a jeho použití ke klasifikaci. Zdroj [6].

```
TRAIN-KNN(ℂ, 𝔻)
    𝔻′ ← PREPROCESS(𝔻)
    k ← SELECT-K(ℂ, 𝔻′)
    return 𝔻′, k

APPLY-KNN(ℂ, 𝔻′, k, d)
    S_k ← COMPUTENEARESTNEIGHBORS(𝔻′, k, d)
    for each c_j ∈ ℂ
    do p_j ← |S_k ∩ c_j|/k
    return arg max_j p_j
```

Z pseudokódu 4.1 lze pochopit na jakém principu algoritmus funguje. $\mathbb{C}$ je množina všech tříd, $\mathbb{D}$ trénovací sada dat s určenými třídami, k parametr metody kNN, který je

zde určen při procesu trénování, d data určena ke klasifikaci. Nejvíce času zabere algoritmu vyhledání nejbližších sousedů, tento proces se děje při každé klasifikaci a při použití velké trénovací sady může výrazně zpomalovat rozpoznávání. Z toho důvodu se kNN někdy také nazývá líný student (*lazy learner*).

4.5 Neuronové sítě

Tato metoda vychází z poznatků biologie o fungování nervového systému u člověka. Neuronová síť se skládá z neuronů a synapsí. Neurony představují funkci o několika možných vstupech a jednou hodnotou na výstupu. Synapse jsou ohodnocené spoje mezi jednotlivými neurony. Sítě se zobrazují jako ohodnocený orientovaný graf. Při trénování modelu se určují funkce jednotlivých neuronů (nejčastěji se jedná o funkci sigmoidy nebo logickou funkci), jejíž hodnoty se pohybují v intervalu $\langle 0, 1 \rangle$. Jednotlivé neurony pouze modifikují vstupní data a předávají je dále.

Architektura Síť se většinou skládá z několika vrstev. Vstupní vrstva slouží pouze pro předání parametrů do sítě, neurony nijak nemodifikují data. Výstupní vrstva obsahuje tolik neuronů, jaký je počet tříd, do kterých se mají data zařadit. Neuron s největší hodnotou určuje pak rozpoznanou třídu objektu. Mezi těmito vrstvami se nalézá zpravidla několik dalších, které se souhrnně označují jako skrytá vrstva.

V základní verzí neuronových sítí jsou data předávána z předchozích vrstev do následujících. Novější algoritmus toto pravidlo nedodržuje, jedná se o neuronové sítě se zpětnou propagací chyby (*backpropagation*).

Kapitola 5

Implementace aplikace a vyhodnocení výsledků

Součástí této práce je nástroj pro vyhledávání a rozpoznání textu v obraze aplikující některé z popsaných metod. Aplikace je implementována v jazyce C++ a vyvíjena na platformě GNU/Linux. Pro práci s obrazem je využito knihovny OpenCV [1] v aktuální verzi 2.1. Tato knihovna obsahuje i několik implementací algoritmů strojového učení, které jsou v programu také použity.

Dříve než se může začít se segmentací textu, je potřeba obraz analyzovat a provést některé úpravy. Mezi tyto úpravy patří převod obrazu do binární podoby popsaný v sekci 5.1 nebo zjištění úhlu pootočení naskenovaného dokumentu a následné vyrovnání, kterým se zabývá část 5.2. Vybranému způsobu segmentace obrazu na řádky, slova a znaky se věnuje sekce 5.3. Dále je v této kapitole popsán proces výběru použitých příznaků a jsou zde prezentovány dosažené výsledky.

K dispozici je několik módů aplikace, jejich použití je popsáno v dokumentaci programu.

- Vyrovnání obrazu

  Nalezení úhlu pootočení a vyrovnání obrazu. Možnost zadat úhel, o který se má obraz nejdříve pootočit, což je možné využít pro testování přesnosti použitého algoritmu.

- Tvorba trénovací sady

  Vyhledání jednotlivých znaků a uložení každého z nich do vlastního souboru. Program je možné spustit v interaktivním režimu, kdy postupně zobrazuje vyhledané znaky a ptá se na jejich typ. Uživatel pak pomocí klávesnice vybere znak, který je zobrazen a program uloží obraz znaku do složky určené pouze pro tento typ znaku. Jsou podporovány pouze znaky anglické abecedy. Při spuštění v neinteraktivním režimu (implicitní nastavení) se uloží všechny znaky do výstupního adresáře a je možné si je pak ručně roztřídit. U znaků nejsou nijak modifikovány rozměry.

- Extrakce příznaků z datové sady

  Program prohledá adresář s roztříděnými soubory znaků a provede extrakci příznaků. Třída se určí z názvu adresáře a příznaky se vypočítávají ze souborů obrazů. Výsledkem je soubor s popisem všech znaků v zadaném adresáři. Formát výstupu je stejný jako používá knihovna LIBSVM [3], tj. na každém řádku příznaky jednoho souboru. Prvním číslem je kód popisovaného znaku následovaný očíslovanými příznaky.

- Trénování klasifikátoru

  Tvorba klasifikačního modelu na základě trénovacích dat. Výstupem je model uložený v souboru a připraven k použití při klasifikaci nových dat. Úspěšnost se vyhodnocuje na trénovacích datech a volitelně i na testovacích datech.

- Rozpoznávání textu

  Vstupní obraz se převede do binární podoby, provede se segmentace znaků a extrakce příznaků, ze kterých poté klasifikátor odhadne, o jaký znak se jedná. V programu je implementováno použití dvou klasifikátorů a sice SVM a rozhodovacích stromů (varianty *random trees*), jejich výběr a nastavení je možný pomocí konfiguračního souboru. Rozpoznaný text je vypsán na standardní výstup a volitelně do souboru.

- Testování úspěšnosti rozpoznání datových sad

  V tomto režimu se provede rozpoznání extrahovaných příznaků některé z testovacích sad. Vypíše se procentuální úspěšnost a tabulka pro vyhodnocení úspěšnosti rozpoznání (tzv. *confusion matrix*).

- Zpracování trénovacích dat systému Tesseract

  Data jsou k dispozici na webu tohoto OCR nástroje. Jedná se o obrazy s ukázky textu v deseti různých fontech a soubory s popisem těchto obrazů. Znaků je v těchto souborech velké množství (přes 50 tisíc), ale v rámci jednoho souboru se jednotlivé typy znaků příliš neliší, proto program získá pouze několik (3) ukázek od každého znaku a uloží je do odpovídajících adresářů.

- Zpracování MNIST databáze ručně psaných číslic

  Tato databáze obsahuje 60 tisíc trénovacích a 10 tisíc testovacích obrázků s ručně psanými číslicemi. Program v tomto režimu je extrahuje z původního formátu do oddělených souborů pro každý obraz. Data nejsou samozřejmě vhodná pro trénování klasifikátoru na rozpoznávání tištěného textu, ale je možné na nich vyzkoušet použití různých příznaků a klasifikátorů a při úspěšném natrénování rozpoznávat programem ručně psané číslice. Používané příznaky se ale pro klasifikaci těchto dat zřejmě nehodí. S SVM se mi nepodařilo dosáhnout úspěšnosti rozpoznání větší než 72% na testovací sadě, což je při takovém počtu trénovacích dat málo. Při použití náhodných rozhodovacích stromů byla úspěšnost rozpoznání 85%. V textu práce se tímto dále nezabývám.

- Porovnání dvou textů

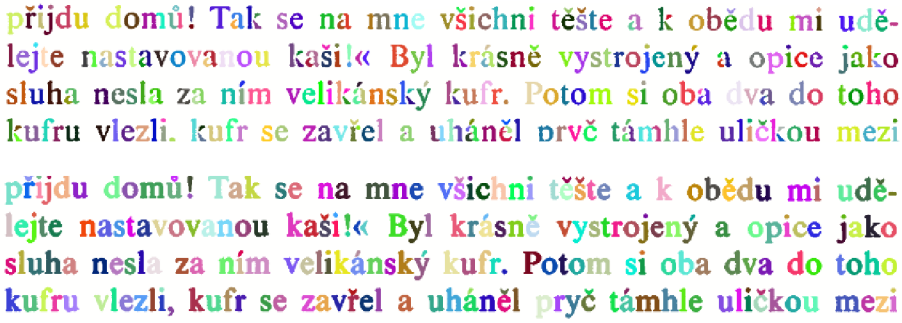
  Možnost porovnání originálního textu, který je v obraze, s výsledkem rozpoznávání. Při porovnání se nehledí na tzv. bílé znaky a kontroluje se, zda nebyla špatně určena jen velikost znaku. U každého znaku je vypsáno, jako co byl rozpoznán a také je spočítána procentuální úspěšnost (i bez ohledu na velikost znaku). Předpokladem však je naprosto správná segmentace obrazu, v opačném případě výpočty nemají smysl.

5.1 Binarizace

Vstupní obraz se převádí na obraz ve stupních šedi (běžně označován jako černobílý obraz), protože informace o barvě pro rozpoznávání textu není většinou důležitá. Černobílý obraz je pak vhodné převést do binární podoby pro oddělení nejdůležitějších informací obsažených v obraze a zbavení se šumu.

Vyzkoušel jsem několik způsobů binarizace, dokumenty jsou ale někdy velice specifické a žádný způsob nevyhovoval pro binarizaci všech zkoušených dokumentů. Rozhodl jsem se tedy pro použití interaktivního nastavení prahu. Po spuštění programu se pomocí Otsuho metody vybere prah a provede se globální prahování. V takto získaném obraze se vyhledají kontury a zobrazí se uživateli vyplněné barvou na černém pozadí. Pro každou konturu je vygenerována jiná barva a uživatel tak může zkontrolovat, zda jsou jednotlivé znaky správně odděleny. Jeden znak by měl být vykreslen jednou barvou a sousední znaky by měly mít barvu jinou. Uživatel má možnost pomocí jezdce nastavit jinou hodnotu prahu.

Program je možné pomocí přepínače donutit použít adaptivní prahování, což se hodí v případě, že dokument nebyl při skenování či fotografování rovnoměrně osvětlen. Na obr. 5.1 je zobrazena část okna spuštěného programu ve fázi kontroly správnosti provedené binarizace. V horní části obrázku je úkázáno nevhodně použité adaptivní prahování. Vespod je výsledek použití Otsuho metody pro nalezení prahu a jeho využití při globálním prahování.

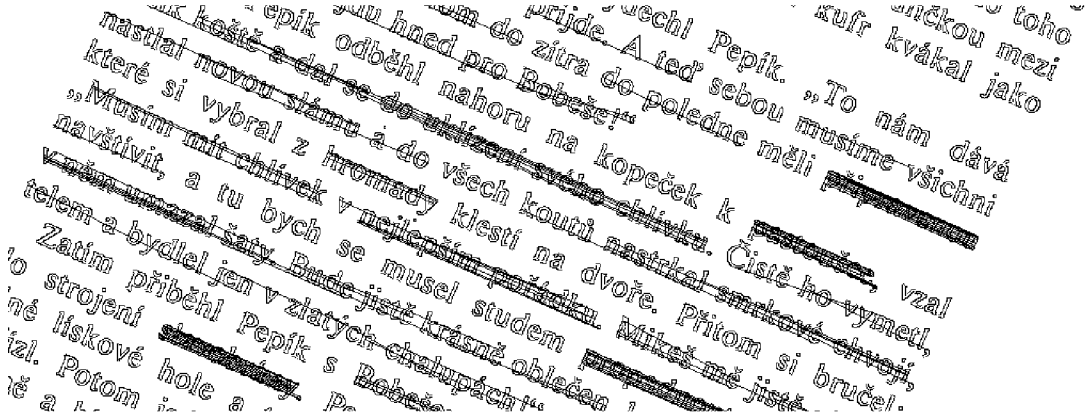


Obrázek 5.1: Kontroly vhodnosti provedené binarizace.

5.2 Vyrovnání

Pomocí aplikace je možné vyhledat úhel pootočení a obraz vyrovnat. V obraze jsou nejdříve vyhledány hrany Cannyho detektorem a poté jsou pomocí Houghovy transformace nalezeny řádky textu. Ještě před aplikací Hougovy transformace je obraz zmenšen, aby použité parametry této transformace byly platné pro obrazy různých velikostí. Úhel pootočení je vypočítán jako průměr z úhlů, které svírají řádky s horizontální osou.

Ukázka 5.2 zobrazuje výřez z pootočeného dokumentu, ve kterém byly vyhledány řádky. Parametry jsou nastaveny přísně a neoznačují se tak celé řádky, ale většinou jen delší slova. Je to z toho důvodu, aby se úsečky označovaly opravdu jen v rovině jednoho řádku a nezasahovaly do předchozích nebo následujících řádků.



Obrázek 5.2: Použití Houghovy transformace.

Úspěšnost určení úhlu je při na obrázcích s mnohařádkovým textem relativně vysoká. Na testovaných obrázcích bylo správně rozpoznáno pootočení v intervalu od jednoho do dvaceti stupňů.

5.3 Segmentace

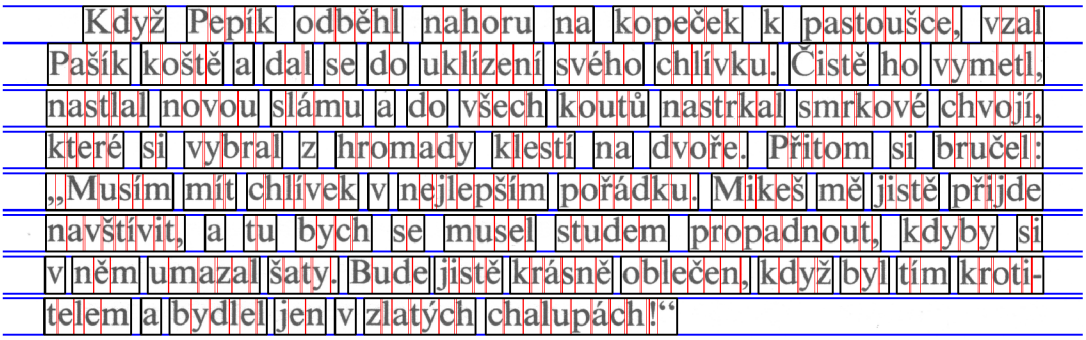
V programu není vypracována analýza obrazu jako celku. Jako vstup může být použit pouze obraz s odstavcem textu, který již musí být vyrovnán (vyrovnání je nutné provést zvlášť, aby neovlivňovalo výsledky segmentace a následného zpracování).

Ze vstupního obrazu se vytvoří horizontální projekční profil, který se pak analyzuje shora dolů. Když počet pixelů na některé horizontální lince (vysoké jeden pixel) překročí stanovenou hranici, zaznamená se pozice jako začátek nějakého řádku a pokračuje se v analýze. Po nalezení linky s menším počtem pixelů, než je stanovená hranice, je tato pozice zaznamenána jako konec řádku. Pokud má nalezený řádek výšku alespoň deset pixelů, je uznán.

Po získání pozice řádku se může přistoupit k segmentaci slov. Zpracovává se vertikální projekční profil řádku. Nejdříve je zjištěna průměrná vyskytující se mezera ze všech mezer na řádku (mezi slovy i mezi jednotlivými znaky). Poté se hledají postupně zleva doprava mezery podobně jako se vyhledávaly v obraze řádky. Velikost každé mezery se porovnává s nějakým násobkem průměrné velikosti mezer na řádku a pokud je větší, je mezera brána jako mezera mezi dvěma slovy.

V oblasti slova jsou pomocí funkce knihovny OpenCV vyhledány všechny kontury a ke každé kontuře je zjištěn obdélník, který ji těsně obklopuje. Pokud levý spodní roh tohoto opisujícího obdélníku neleží pod hranicí poloviny řádku, dále se tento objekt nezpracovává. Oblast ostatních kontur se rozšíří až k horní a dolní hranici řádku. Tímto způsobem je docíleno také toho, že malé a velké znaky jsou zobrazeny ve správném poměru a při použití jiných příznaků by měla být velikost znaků správně rozlišována. Nalezené oblasti pak představují oblasti jednotlivých znaků a jsou provedeny menší úpravy, jako je odstranění objektů dotýkajících se okrajů (většinou části okolních písmen) a znak je připraven pro extrakci příznaků.

Na obrázku 5.3 je ukázka správně provedené segmentace znaků, které bylo dosaženo zmíněnými kroky. Modře jsou zobrazeny hranice řádků, černě hranice slov a červeně oblasti znaků.



Obrázek 5.3: Ukázka správné segmentace znaků.

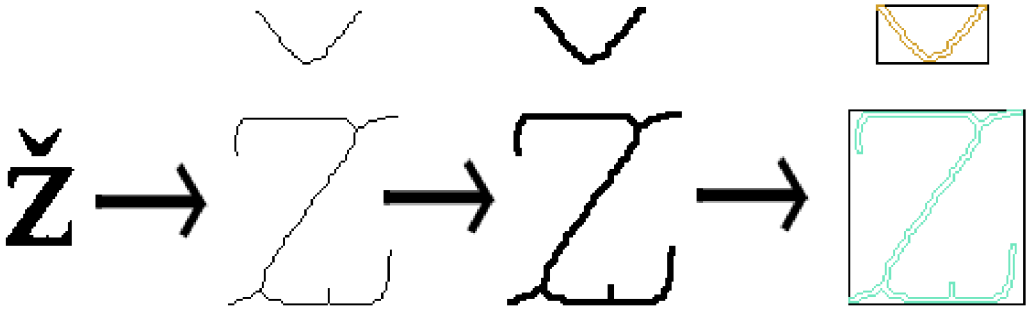
5.4 Výběr příznaků

S výběrem správných příznaků byly největší potíže. První variantou bylo vyzkoušení klasifikátoru kNN s použitím všech pixelů obrazu znaku (bez extrakce příznaků), ale výsledky nebyly příliš uspokojivé, proto jsem se tímto dále nezabýval.

Druhým vyzkoušeným způsobem bylo použití invariantních Hu-momentů vypočítaných z celé oblasti nalezeného znaku a klasifikace pomocí SVM. Tento přístup také nebyl vyhovující, klasifikátor nebylo možné natrénovat tak, aby rozděloval správně alespoň trénovací data (úspěšnost byla nejvýše 30%).

Aby byl počet vstupních informací co nejmenší a míra poskytnutých informací naopak vysoká, zkusil jsem v oblasti znaků nejdříve vyhledat kontury a teprve z nich počítat invariantní momenty. Přesnost rozpoznávání byla o mnoho lepší a pomocí natrénovaného klasifikátoru bylo už v omezené míře možné rozpoznávat neznámá data. Nebylo možné odlišit od sebe dvojce znaků lišící se hlavně rotací (dp, nu, cn) nebo zrcadlově převrácené znaky (bd, pq), což ale vyplývá z podstaty použitých příznaků invariantních vůči těmto jevům. Odlišit zrcadlově převrácené obrazy sice lze podle sedmého Hu momentu, ale jeho hodnoty byly příliš malé na to, aby se projevily ve výsledcích rozpoznání. Program také chybně určoval velikost písmen (vV atp.), opět protože příznaky byly osvobozeny od závislosti na velikosti.

Posledním vylepšením je použití pouze normalizovaných centrálních momentů, které nejsou invariantní k pootočení. Ty se navíc nepočítají pouze z kontur nalezených znaků. Na obrazech jednotlivých písmen je nejprve provedeno ztenčení (algoritmus podle [4]) a poté jeden průchod morfologické operace dilatace z důvodu zachování spojitosti znaku. V takto upraveném obraze jsou vyhledány kontury pomocí vestavěné funkce knihovny OpenCV (princip algoritmů vyhledávání kontur [5], použit *Moore-Neighbor*). Teprve z těchto kontur jsou vypočítány momenty. Pokud je v oblasti znaku nalezeno více kontur, jsou z nich vypočítané momenty sčítány. Pokud je nalezena kontura, jejíž spodní hranice je nad polovinou oblasti znaku, tak se tato kontura zahodí. Takto je získáno 7 příznaků charakterizující odpovídající tvar znaku. Na obrázku 5.4 je vidět tato procedura aplikovaná na znak „ž“.



Obrázek 5.4: Postup při zpracování nalezeného znaku.

5.5 Vytvoření datové sady

Trénovací sada je vytvořena z ideálních dat. Použité znaky bez výrazné deformace a jsou dobře čitelné. Data byla vygenerována nebo vyextrahována z existujících trénovacích sad.

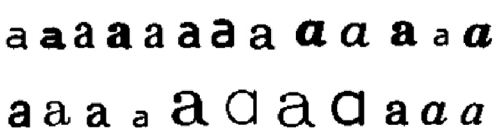
- Trénovací data Tesseractu

  Pro vytvoření trénovací sady jsem použil obrazy s textem, které jsou ke stažení na webových stránkách OCR nástroje Tesseract, což je jeden z nejlepších dostupných open source OCR nástrojů, který je nyní sponzorován hlavně společností Google. Na každém obraze je ukázka vzhledu různých znaků za použití jednoho druhu písma. V trénovací sadě jsou 3 ukázky malé i velké varianty každého znaku anglické abecedy a číslic ve všech fontech (těch je použito 18). Pro každý znak je tedy 54 ukázek, celkem 3348 vzorů znaku.

- Vygenerované znaky

  Dalšími použitými znaky jsou vygenerována písmena a číslice pomocí funkcí knihovny OpenCV. Z dostupných šesti druhů písma obsahuje trénovací sada od každého typu 4 znaky. Celkem je takto vygenerováno 1488 vzorů.

Na obrázku 5.5 je ukázka rozmanitosti prvního znaku abecedy. Celkem trénovací sada obsahuje 4836 znaků a několik obrazů interpunkčních znamének.



Obrázek 5.5: Rozmanitost datové sady.

Testovací sada se skládá ze znaků z tří naskenovaných stran textu segmentovaných a roztříděných pomocí interaktivního režimu vytváření datové sady.

- Alenka

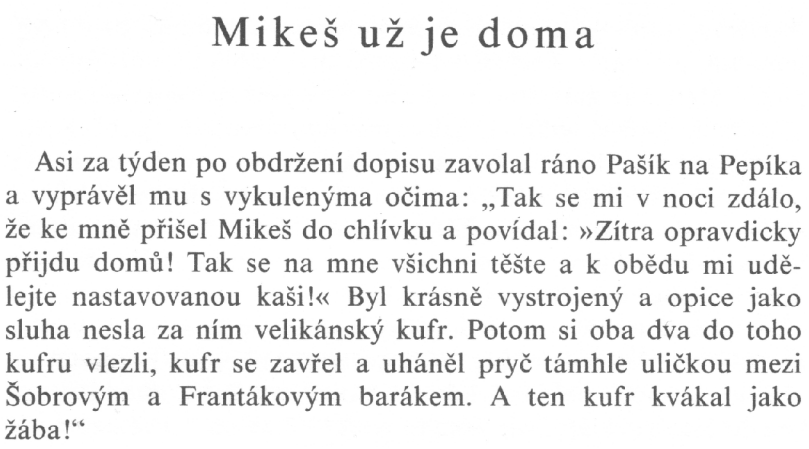
  Znaky z naskenované první strany anglické verze knihy Alenka v říši divů (použito jako testovací obrázek u OCR nástroje Ocropus). Obsahuje 48 typů písmen, celkem 2030 znaků.

- Kačer

  Zpracovaná jedna strana naskenované stránky z knihy Příběhy z Kačerova. jak byl rozpoznán programem (všechna písmena byla po rozpoznání zmenšena pro lepší čitelnost). Celkem 32 různých znaků ve 492 variantách.

- Mikeš

  Náhodně vybraná strana z knihy Mikeš od Josefa Lady. Na obr. 5.6 je zobrazen text prvního odstavce. Obsahuje 34 různých znaků, celkem 1404 položek.


Mikeš už je doma

Asi za týden po obdržení dopisu zavolal ráno Pašík na Pepíka a vyprávěl mu s vykulenýma očima: „Tak se mi v noci zdálo, že ke mně přišel Mikeš do chlívku a povídal: »Zítra opravdicky přijdu domů! Tak se na mne všichni těšte a k obědu mi udělejte nastavovanou kaši!« Byl krásně vystrojený a opice jako sluha nesla za ním velikánský kufr. Potom si oba dva do toho kufru vlezli, kufr se zavřel a uháněl pryč támhle uličkou mezi Šobrovým a Frantákovým barákem. A ten kufr kvákal jako žába!“

Obrázek 5.6: Část testovaného dokumentu

5.6 Dosažené výsledky

Při použití programu je možné si zvolit mezi použitím klasifikátoru *support vector machines* nebo *random trees*. Při vývoji a testování jsem se zaměřil na použití SVM, ale při správném nastavení parametrů jsou použitelné i rozhodovací stromy. Protože je použit stejný formát uložení extrahovaných příznaků, jako používá knihovna LIBSVM, je možné použít nástroje této knihovny pro vyhledání nejvhodnějších parametrů pro trénování. Vyhledání probíhá trénováním klasifikátoru za použití různých parametrů a následná cross-validace na vstupních datech. Za nejlepší parametry se považují ty, se kterými poté klasifikátor rozpozná znaky z testovací sady s nejlepší přesností. Všechny parametry pro klasifikátory lze změnit v konfiguračním souboru, není proto nutné pro vyzkoušení jiných parametrů program překompilovávat.

Po ukončení trénování klasifikátoru se otestuje úspěšnost rozpoznání na trénovacích datech i na testovacích datech, pokud jsou k dispozici. Výsledky se vypíšou v procentuální podobě včetně úspěšnosti v případě, že by se nebrala v úvahu velikost znaků. A dále se

vytvoří a zobrazí tabulka pro posouzení úspěšnosti (tzv. *confusion matrix*, viz obr. 5.7), kde jsou vypsány všechny testovací znaky a k ním kolikrát a jak byly rozpoznány.

Pro řešení neoddělitelných dat se používá u SVM tzv. měkkého okraje (*soft margin*), kterým se určuje, kolik vzorků dat se při trénování může nacházet uvnitř maximalizované hranice hledané nadroviny. K tomuto účelu se nastavuje parametr C, jehož zvyšováním roste penalizace za špatně klasifikované objekty a vytváří se přesnější model, který nemusí dobře zobecňovat popisy tříd. Druhým potřebným parametrem je γ, což je koeficient pro použitou jádrovou metodu transformace do více dimenzí. U různých metod má jiný význam.

Pomocí nástrojů knihovny LIBSVM byly zjištěny pro trénování datové sady vhodné hodnoty parametrů ($C = 2048$ a $\gamma = 2$). V tabulce 5.1 jsou vypsány procentuální úspěšnosti klasifikátoru při použití různých parametrů. Testováno bylo na datech popsaných v 5.5. Je vidět, že parametry nalezené parametry jsou zřejmě vyhovující, jejich větší obměnou bylo dosaženo horších výsledků, které v tabulce nejsou zobrazeny.

Z tabulky lze vyčíst, že úspěšnost rozpoznání se pohybuje přes 70% a pokud se nehledí na velikost písmen, tak přes 80%.

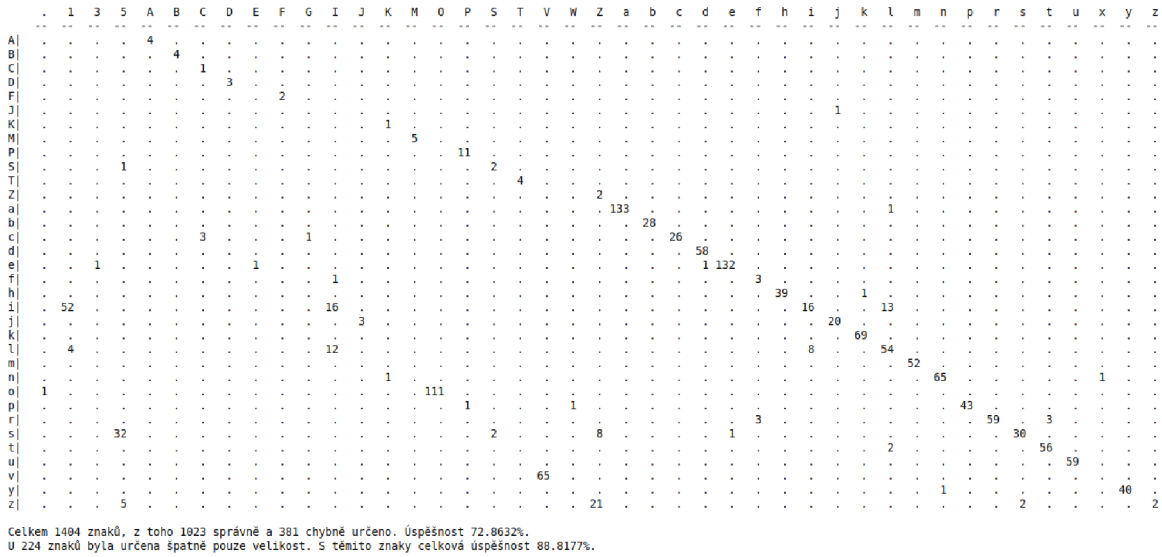
γ	C	Mikeš	Kačer	Alenka	Trénovací sada
0.5	2048	70.8/86.9	67.9/76.6	69.0/81.0	92.3/94.9
2	256	72.7/88.6	74.0/82.1	74.0/85.6	94.5/96.6
2	512	72.7/**88.6**	74.0/82.1	74.0/**85.6**	94.5/96.6
2	1024	72.9/88.8	74.2/**82.7**	74.0/85.2	95.7/97.4
2	2048	72.8/88.2	71.3/82.1	73.5/85.3	96.0/97.3
4	2048	68.9/83.8	66.3/74.2	68.5/79.9	96.0/97.3
6	2048	71.5/85.9	65.9/72.2	70.7/82.1	97.2/**98.5**
8	2048	67.9/82.5	65.2/72.0	68.4/80.2	96.5/97.6

Tabulka 5.1: Úspěšnost rozpoznání znaků z testovacích datových sad

Výsledek rozpoznání textu prvního odstavce strany z Mikeše (viz. 5.6). Pro lepší čitelnost byla velká písmena převedena na malá.

mlke5 uz je doma
a5l za tyden po obdrz3n1 dop1su zavolal rano pasik na pep1ka
a vypravel mu s vykulenyma ocimax yytak se m1 v nocl zdaloy
ze ke mne pr1sei m1ke5 do chllvku a povldalx jjzitra opravd1cky
prijdu domull tak 5e na mne vs1chn1 te5te a k obedu m1 udel
lejle na5tavovanou kasllll byl krasne vystfojeny a op1ce jako
sluha ne5ia za n1m vellkanzky kufrx potom 51 oba dva do toho
kuiru vleziiy kufr ze zavre1 a uhanel pryc tamhle uiickou mezi
5obrovym a frantakovym batakemw a ten kufr kvakal jako
zabail

S trochou představivosti je tento text čitelný. I na této malé ukázce je vidět, s kterými písmeny má program problémy. Zřetelnější to je v tabulce 5.7, která je vyhodnocením rozpoznání celé strany. Znaky i, j, z, s patří mezi nejčastěji špatně rozpoznané. Nezávislost použitých příznaků na velikosti znaku se projevuje špatným rozpoznáním znaku 'o' jako 'O' nebo 'v' jako 'V' a jiných.



Obrázek 5.7: Vyhodnocení úspěšnosti rozpoznání jedné z testovacích sad.

Druhou ukázkou je první odstavec z knihy Alenka v Říši divů (obr. 5.8).

1 Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

Obrázek 5.8: Originál rozpoznávaného textu.

A zde je přepis rozpoznaného textu:

1 down the rabbltlhole
allce was beg1nn1ng to get very tlred ol zltt1ng by her x1zter on the bank?
and ol hav1ng noth1ng to dow once or tw1ce she had peeped 1nto the book her
51ztef ww read1ngj but 1t had no picturez or conveizat1onz ln ltj and what ls
the use ot a book? thought allce wlthout plc1ures or conversation?l

Uvedené ukázky jen potvrzují úspěšnost rozpoznání kolem 80% bez přihlížení na velikost písmen. Na českých textech lze vidět, že program odstraňuje diakritiku. To je způsobeno tím, že nebyl natrénován na českých znacích a z toho důvodu se při zpracování nalezeného znaku diakritika odstraňuje a příznaky se tak extrahují pouze z písmen anglické abecedy. Míra rozpoznání je slabá pro běžné využití programu v praxi, program je však implementován tak, že jednotlivé části procesu rozpoznávání je možné snadno vylepšit a nahradit. Návrhy možných vylepšení jsou uvedeny v následující sekci (5.7).

5.7 Návrhy možných vylepšení

Navrhnout vylepšení aplikace je možné ve všech směrech. Program neprovádí analýzu dokumentu, tzn. rozlišení textu od jiných částí (obrázky, tabulky atd.). Tu by bylo možné doimplementovat a nalezené bloky textu by se pak předávaly současným metodám. Předzpracování obrazu by také mohlo být vylepšeno některou z metod pro odstranění šumu či použitím jiných postupů binarizace.

Způsob vyhledání řádků, slov a znaků je v nynějším stavu použitelný, ale při horší kvalitě obrazu nastávají problémy. Pokud mezi dvěma řádky nelze nakreslit přímka, která je odděluje a neprochází žádným ze znaků, pak jsou tyto řádky sloučeny. Jedno z možných řešení by bylo například využít metody pro vyhledání kontur, u všech kontur spočítat střed objektu, který popisují a na základě analýzy těchto centrálních bodů se rozhodnout, na kterém řádku daný objekt leží.

Nejvíce potřebným vylepšením je změna nebo přidání příznaků rozlišujících velikost znaků. Použité příznaky mají problémy s odlišením některých podobných tvarů a nerozlišují velikost, což se projevuje na špatné detekci velikosti mnoha písmen. Dobrým řešením by mohlo být například použití příznaků získaných více postupy (zoning, crossing, momenty).

Nabízí se také možnost použít jiný klasifikátor, ale to by zřejmě nevedlo k výrazným rozdílům v přesnosti rozpoznávání, problém je zřejmě v použitých příznacích.

Výrazně zvýšit úspěšnost rozpoznání by mohla oprava výsledků pomocí slovníků. Pro každý rozpoznávaný jazyk by bylo nutné přidat jeden slovník, ve kterém by se pak vyhledávala slova nejbližší těm rozpoznaným.

K programu by bylo možné udělat grafické rozhraní, kde by uživatelé mohli manuálně vybrat oblasti s textem, zkontrolovat pootočení atp. Pokud by program po vylepšení vykazoval dobré výsledky, jsou možnosti použití široké. Pomocí dnešních mobilních přístrojů lze pořizovat fotografie, bylo by tedy možné např. v mobilním telefonu obraz okamžitě zpracovat, vyhledat text a dále s ním pracovat (přeložit, odeslat atp.).

Trénovací datová sada by mohla být rozšířena o další typy znaků a jejich různé varianty, což by přispělo k většímu zobecnění tříd klasifikátorem.

Kapitola 6

Závěr

Cílem této práce bylo prostudovat problematiku rozpoznávání textu v obraze a aplikovat zjištěné poznatky při implementaci nástroje pro vyhledání a rozpoznání textu v obraze. Tento úkol se podařilo splnit. V kapitolách 2 a 3 jsou popsány některé metody pro zpracování obrazu, vyhledání textu a extrakce příznaků. Pro klasifikaci znaku na základě získaných příznaků je možné použít strojového učení, nejznámější algoritmy popisuje kapitola 4.

Výběrem metod a implementací nástroje pro vyhledání a rozpoznání textu v obraze se zabývá kapitola 5. Za účelem testování vytvořené aplikace byla vytvořena datová sada znaků, jejím získáním se zabývá sekce 5.5. Dosažené výsledky jsou popsány v sekci 5.6.

Výzkum v oblasti rozpoznávání textu z obrazu probíhá již od roku 1929, kdy si Gustav Tauschek nechal patentovat opticko-mechanický systém na rozpoznávání textu. Po tolika letech je OCR stále aktuální a vyvíjející se oblastí. Současné komerční systémy dosahují velice dobrých výsledků a na jejich vývojem se zabývá velké množství lidí a společností. Mnou zhotovený nástroj proto není možné porovnávat s dlouhodobě vyvíjenými systémy, ale nakročeno má správným směrem. Datová sada i zdrojové kódy programu jsou na přiloženém CD.

Literatura

[1] Bradski, G. R.; Kaehler, A.: *Learning opencv, 1st edition*. O'Reilly Media, Inc., 2008, ISBN 9780596516130.

[2] Breiman, L.; Cutler, A.: Random Forests. [Online; accessed 22-April-2010].
URL http://www.stat.berkeley.edu/users/breiman/RandomForests

[3] Chang, C.-C.; Lin, C.-J.: *LIBSVM: a library for support vector machines*. 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] Cychosz, J. M.: Efficient binary image thinning using neighborhood maps. 1994: s. 465–473.

[5] Ghuneim, A. G.: Přehled metod pro vyhledávání kontur. 2000, [Online; accessed 2-May-2010].
URL http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/moore.html

[6] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008, ISBN 978-0-521-86571-5.

[7] O'Gorman, L.: *Document Image Analysis: An Executive Briefing*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997, ISBN 081867802X.

[8] Pegoraro, V.: Ukázky použití detektorů hran. [Online; accessed 8-February-2010].
URL http://www.cs.utah.edu/~vpegorar/courses/cs7966/Assignment4

[9] Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, druhé vydání, 2003, ISBN 0137903952.

[10] Trier, Ř. D.; Jain, A. K.; Taxt, T.: Feature extraction methods for character recognition-A survey. *Pattern Recognition*, ročník 29, č. 4, 1996: s. 641 – 662, ISSN 0031-3203.
URL http://www.sciencedirect.com/science/article/B6V14-3VSNG9F-T/2/4299d6ec297d5c2835fb74fee5fce588

[11] Španěl, M.; Beran, V.: *Obrazové segmentace: Přehled existujících metod*. 2005, [Online; accessed 4-April-2010].
URL http://www.fit.vutbr.cz/~spanel/segmentace

[12] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika*. Computer Press, 2005, ISBN 80-251-0454-0.