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

POLOAUTOMATICKÉ POŘÍZENÍ ROZSÁHLÉ DATABÁZE RUČNĚ PSANÝCH ZNAKŮ

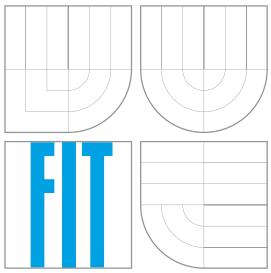
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

IVO ŠTĚPÁNEK

BRNO 2013

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

POLOAUTOMATICKÉ POŘÍZENÍ ROZSÁHLÉ DATABÁZE RUČNĚ PSANÝCH ZNAKŮ

SEMIAUTOMATIC COLLECTION OF LARGE DATABASE OF HANDWRITTEN LETTERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — IVO ŠTĚPÁNEK
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se zabývá tvorbou databáze ručně psaných znaků dále využitelných pro rozpoznávání ručně psaného textu. V teoretické části práce je rozebrána problematika systémů pro rozpoznávání ručně psaného textu a datových sad pro něj využitelných. Praktická část se věnuje předzpracování vstupního dokumentu, segmentaci řádků a slov a následnému rozdělení slov na samostatné znaky. Tyto fáze mohou proběhnout zcela automaticky, ovšem je předpokládán vstup uživatele, který má možnost opravit výstup automatického zpracování. Dále se praktická část věnuje anotaci získaných znaků a vygenerování XML dokumentu s anotací a umístěním jednotlivých znaků ze vstupního text. U vytvořeného systému je vyhodnoceno jeho GUI a uspěšnost automatické segmentace.

Abstract

This thesis deals with unconstrained handwritten text recognition. The issue of artificial neural networks and systems for unconstrained handwritten text recognition is described in theoretical part. Practical part of the thesis aims at preprocessing of the input document and line and word segmentation followed by extraction of isolated characters. From these characters are subsequently extracted features for the purpose of classification. Artificial neural network is used for the classification of the isolated characters. The proposed system for handwriting recognition is eventually tested and the results are discussed in terms of recognition rate.

Klíčová slova

rozpoznávání ručně psaného textu, segmentace řádků, segmentace slov, segmentace znaků, databáze znaků, databáze ručně psaných znaků, anotace ručně psaného textu, openCV

Keywords

handwritten text recognition, line segmentation, word segmentation, character segmentation, character database, handwritten character database, handwritten text annotation, openCV

Citace

Ivo Štěpánek: Poloautomatické pořízení rozsáhlé databáze ručně psaných znaků, bakalářská práce, Brno, FIT VUT v Brně, 2013

Poloautomatické pořízení rozsáhlé databáze ručně psaných znaků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Adama Herouta, Ph.D.

.....
Ivo Štěpánek
15. května 2013

Poděkování

Tímto bych chtěl poděkovat vedoucímu bakalářské práce, panu Doc. Ing. Adamu Heroutovi, Ph.D., za spoustu podnětných návrhů a odbornou pomoc poskytnutou při vypracování této práce.

Obsah

Kapitola 1

Úvod

Systémy pro automatické rozpoznávání textu existují už déle než 50 let [7]. A jelikož i přes rostoucí míru využívání počítačů a dalších moderních zařízení psané písmo doposud neztratilo svoji důležitost, není divu, že se jím počítačové vědy i dnes čile zabývají. Od počátku 90. let minulého století je patrný významný nárust aktivity v oblasti výzkumu rozpoznávání ručně psaného textu. Je tedy přirozené, že od té doby systémy pro rozpoznávání značně pokročily. Byly vyvinuty a reálně nasazeny komerční systémy pro rozpoznávání ručně psaných adres nebo pro rozpoznání sumy bankovních šecích. I přes tyto úspěchy je však problematika rozpoznávání ručně psaného textu stále považována za náročnou oblast výzkumu, kde lze stále ještě zlepšovat míru úspěšnosti rozpoznávání na reálných datech. Popsané systémy, které jsou reálně používany, se totiž vyznačují velmi specifickým určením a obvykle zpracovávají problematiku, pro niž jsou jasně definována vstupní data. U obecných systémů pro rozpoznávání ručně psaného písma, pro které se nedají předpokládat žádná omezení vstupních dat a musí se počítat s jejich velkou různorodostí, stále není dosaženo uspokojivé míry úspěšnosti. [5] Ta je klíčová pro rozšíření možné působnosti systémů pro rozpoznávání ručně psaného textu do dalších oblastí, kterými mohou být například převod osobních poznámek do digitální podoby, digitalizace archivovaných písemností a s ní spojené vytváření digitálních knihoven i mnohé další.

Pro vývoj, testování i samotnou práci systémů pro rozpoznávání ručně psaného textu (dále OCR systémů) je klíčovou součástí prvotní databáze znaků, označovaná také jako trénovací množina. Jedná se o předem získanou množinu sestávající z obrazových dat jednotlivých znaků doplněných o jejich správnou anotaci (přiřazení správného znaku jeho obrazové reprezentaci). Vzhledem k jedinečnosti rukopisu každého jedince je pro algoritmy rozpoznávání ručně psaného textu důležité, aby tato množina byla, pokud možno, co nejrozmanitější a aby obsahovala co nejvíce možných korektních tvarů zastupujících týž znak. Tvorba takovéto trénovací množiny je ovšem, vzhledem k nutnosti správné anotace jednotlivých znaků a jejich správné segmentace z ručně psaného textu, časově velmi náročná. Právě z tohoto důvodu jsem si jako téma této práce zvolil vytvoření nástroje pro poloautomatickou tvorbu trénovací množiny.

Cílem je tedy navrhnout systém pro poloautomatickou tvorbu databáze ručně psaných znaků, navržený systém implementovat v programovacím jazyce a s jeho využitím pořídit rozsáhlou databázi českých ručně psaných znaků. Výsledný program bude dále zanalyzován z hlediska uživatelské přívětivosti, úspěšnosti algoritmů využitých pro části systému pracující bez asistence uživatele a celkově z hlediska použitelnosti pro reálné nasazení.

V textu práce jsou nejprve v kapitole 2 popsána základní teoretická východiska pro návrh a vytvoření výše popsaného programu. Jsou zde zmíněny základní požadavky na systém

pro rozpoznávání ručně psaného textu a z uvedených požadavků je zde odvozeno obecné schéma takového systému. V tomto schématu jsou identifikovány jednotlivé kroky, které je při návrhu takového systému třeba realizovat. Každý z kroků, na něž lze proces rozpoznávání ručně psaného písma rozdělit, je stručně vysvětlen. Z architektury obecného OCR systému vychází i vlastní návrh programu vytvořeného v rámci této práce. Ten je podrobněji popsán v kapitole 3. Tato kapitola čtenáře seznámuje s postupy užitými v praktické části práce pro řešení vybraných dílčích problémů týkajících se především segmentace jednotlivých řádků textu, slov a posléze i samostatných písmen. Implementace navržené metody ve zvoleném programovacím jazyce je tématem kapitoly 4. V této kapitole jsou zmíněny nástroje použité pro programovou realizaci návrhu a dále je zde rozebrána základní koncepce a logická struktura vytvořeného programu. Analýze výsledků navržených metod, jejich testování a především vyhodnocením výsledného programu se věnuje kapitola 5. V závěru (kapitole 6) jsou diskutovány dosažené výsledky a je naznačena cesta, kudy by se mohl vývoj vytvořeného systému dále ubírat.

Kapitola 2

Rozpoznávání ručně psaného textu

Jak již bylo zmíněno v úvodu, vytvoření obecného systému pro rozpoznávání ručně psaného textu je náročným úkolem. To je dáno především různorodostí vstupních dat. Každý člověk má svůj jedinečný rukopis, který je určen nejenom tvarem jednotlivých písmen, ale i jejich velikostí, rozestupy mezi nimi, sklonem písma a dalšími aspekty. Z hlediska procesu získání vstupních dat lze obecně rozpoznávání ručně psaného textu dělit na dvě kategorie: offline a online rozpoznávání.

Online rozpoznávání

Pro online rozpoznávání jsou vstupní data získána během psaní. Je tak k dispozici nejen informace o tvaru, ale je přesně známa trajektorie tahu, kterým byl tento tvar napsán včetně směru a rychlosti tahu v daném místě. Vstupní informace je tak bohatší, díky čemuž dosahují online systémy lepších výsledků. Nevýhodou těchto systémů je nutnost použít k získání informací speciální zařízení. Nicméně v poslední době ve spojitosti s rozvojem dotykových displejů narůstá i význam online systémů pro rozpoznávání textu.

Offline rozpoznávání

Offline systémy pracují se statickými vstupními daty. Ta jsou typicky získána skenováním ručně psaného dokumentu a reprezentována rastrovým obrazem. Offline systémy tedy musejí vycházet pouze z informace o tvaru. Dále je při offline rozpoznávání potřeba vzít v úvahu také prostorové rozmístění textu na stránce, které se napříč dokumenty může významně lišit. Tyto skutečnosti jsou příčinou obecně nižší úspěšnosti offline rozpoznávání a celkově větším nárokům na systém pro rozpoznávání písma v porovnání s online systémy. Vzhledem k tomu, že tato práce se věnuje offline systému pro rozpoznávání písma, bude dále v textu pod obecným pojmem rozpoznávání písma rozuměno právě offline rozpoznávání.

2.1 Charakteristika současného stavu

Rozpoznávání ručně psaného textu lze podle oblasti, která je předložena na vstupu, rozdělit na rozpoznávání:

- jednotlivých izolovaných znaků
- samostatných slov

- textu sestávajícího z předem nedefinovaného počtu slov

Zatímco pro úlohy prvního typu jsou k dispozici dostatečně přesné a robustní řešení, problematika rozpoznávání slov a případně i jejich sekvencí stále není uspokojivě vyřešena a zůstává zde prostor pro další bádání. Jedním z používaných přístupů k řešení úloh druhého typu je rozdělení slova do jednotlivých znaků, které jsou poté předloženy klasifikátoru. Nicméně tento přístup lze jen stěží dobře uplatnit bez znalosti textové reprezentace slova. Nastalou situaci lze tedy charakterizovat jako známý problém vejce a slepice. Řečeno jinými slovy – se znalostí textové reprezentace slova by bylo jeho správné rozdělení do jednotlivých znaků proveditelné, ale pro získání textové reprezentace slova je nutné nejprve rozdělit je do znaků. [1]

2.2 Architektura systému pro rozpoznávání textu

Vstupem do systému pro rozpoznávání textu je rastrový obraz typicky získaný skenováním. Ten je v několika fázích postupně zpracováván a text, který je na obraze zaznamenán, je převeden do digitální podoby. Na obrázku 2.1 je schematicky znázorněna architektura systému pro rozpoznávání ručně psaného textu s jednotlivými procesními fázemi a jejich vstupy i výstupy.

2.2.1 Předzpracování vstupního obrazu

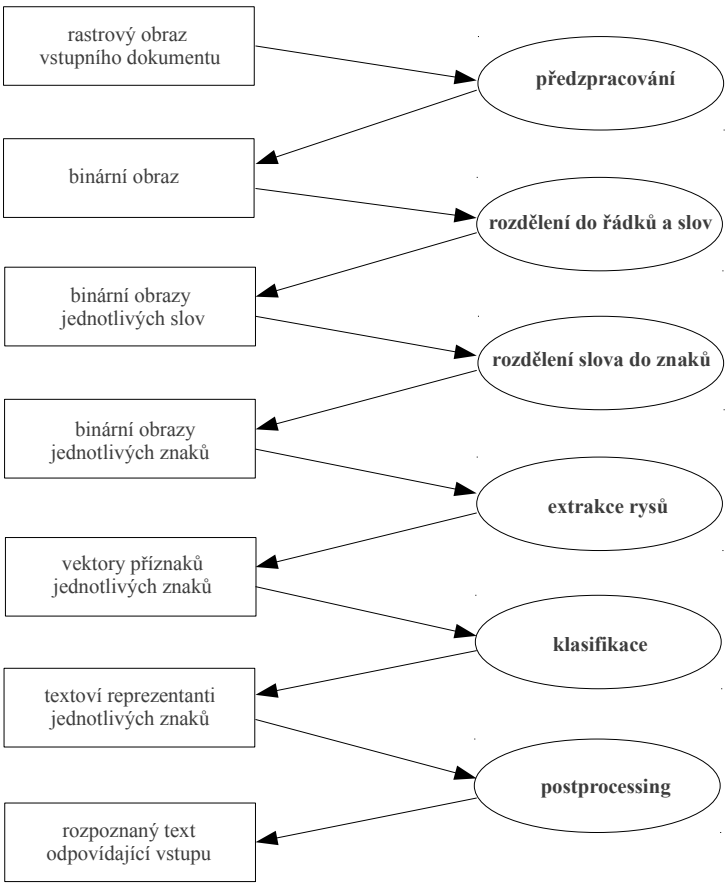
Za účelem reprezentace textu v systému pro rozpoznávání písma je potřeba zvolit takovou formu reprezentace, která přesně zachovává hranice objektů. Obvykle se za tímto účelem používá binární rastrový obraz. Do fáze předzpracování ale vstupuje obraz monochromatický, a je tedy potřeba vstupní obraz na binární převést. Právě to je cílem fáze předzpracování. Vstupní monochromatický obraz získaný skenováním je navíc typicky zatížen určitou mírou šumu, který musí být z obrazu vhodným způsobem odstraněn. Nutnou podmínkou pro předzpracování je zachování co nejvěrohodnější informace o tvaru jednotlivých objektů v obraze.

2.2.2 Segmentace řádků a slov

Binární obrázek vytvořený ve fázi předzpracování je dále potřeba rozdělit na části, které odpovídají řádkům a slovům textu. Tato činnost je označována jako *segmentace*. Dle obecné definice je segmentace proces, během nějž jsou v obraze nalezeny navzájem se nepřekrývající objekty se známou interpretací. [17] V případě rozpoznávání ručně psaného textu jsou tedy na nejvyšší úrovni zpracování za segmenty považovány jednotlivé řádky, na nejnižší úrovni pak izolované znaky. Požadavky na metodu použitou pro segmentaci řádků:

- objekty nacházející se na hranici řádku musejí být přiřazeny do právě jednoho řádku jako celek
- určitá míra odolnosti vůči náklonu řádků v rámci vstupního obrazu textu

Správné rozdělení vstupního obrazu do řádků u češtiny ztěžuje diakritika, především pak pokud se nachází nad velkým písmenem a zasahuje tak do vertikální oblasti přecházejícího řádku. Úlohy segmentace řádku lze chápat jako hledání myšlených čar oddělujících jednotlivé řádky v obrazu textu. Segmentaci slov je pak možné chápat jako hledání mezer mezi slovy umístěnými na jednom řádku. V případě češtiny je nutné při segmentaci slov



Obrázek 2.1: Grafické znázornění architektury systému pro rozpoznávání ručně psaného textu. Elipsy v pravém sloupci představují jednotlivé pracovní fáze systému, obdelníky v levém sloupci jejich vstupy a výstupy.

brát v úvahu diakritická znaménka, která mohou při větším sklonu písma být nesprávně přiřazena k některému ze sousedních slov.

2.2.3 Segmentace znaků

Jak již bylo zmíněno výše, rozdělení slova do jednotlivých znaků reprezentujících písmena slova je složitým úkolem, který se doposud nedaří s dostatečnou mírou úspěšnosti vyřešit. Problémem v této fázi je nejen neznalost celé textové reprezentace děleného slova (což je přirozené s ohledem na povahu problematiky), ale i další faktory:

- **Sklon písma** – bez jeho opravy je při velkém sklonu náročné oddělit znaky tak, aby nebyly porušeny.
- **Diakritika** – háčky a čárky se často nacházejí na rozhraní jednotlivých znaků. Zatímco člověk je při čtení automaticky dle znalosti kontextu přiřazuje ke správným

slovům, u strojů je napodobení takového chování velmi obtížné.

2.2.4 Extrakce rysů

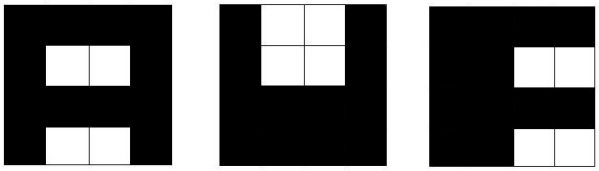
Před započetím klasifikace jednotlivých znaků do definovaných tříd je nejdříve ze vstupních dat (rastrových obrázků samostatných znaků) nezbytné získat co nejrelevantnější informaci o jejich podobě. Tato činnost se označuje jako *extrakce rysů*. Rysy, které jsou pro potřeby rozpoznávání extrahovány, by měly mít minimální proměnlivost v rámci stejné třídy a zároveň by měly zvýrazňovat rozdíly mezi prvky napříč třídami. Je přirozené, že pro účely extrakce rysů byla vyvinuta spousta nejrůznějších metod pokoušejících se co nejlépe splnit popsaná kritéria. Výběr vhodné metody pro extrakci rysů závisí především na oblasti, kde má být použita, a na dostupných datech. Metoda, která dosahuje dobrých výsledků v jedné oblasti, může být pro jinou problematiku zcela nepoužitelná. Dále budou krátce popsány některé z metod používané pro extrakci rysů v systémech pro rozpoznávání ručně psaného textu. [10]

Hodnoty pixelů binárního obrazu

Jedná se o nejpřímější metodu extrakce rysů. Byla používána především v počátcích systémů pro rozpoznávání textu a jako rysy bere hodnoty pixelů binárního obrazu, který je klasifikován. Problémem této metody je nízká odolnost vůči nepřesnostem ve vstupních datech. Pro rozpoznávání ručně psaného písma tedy není vhodná a v současné době se ani nepoužívá.

Vertikální a horizontální projekce

Rysy extrahované touto metodou vychází z počtu černých pixelů v jednotlivých sloupcích (*vertikální projekce*) a řádcích obrazu (*horizontální projekce*). I přes svou jednoduchost dosahuje poměrně dobrých výsledků. Jelikož ale některé znaky nedokáže zcela oddělit, je obvykle doplněna ještě dalšími příznaky. Metoda může být různě modifikována – například použitím vícekošového histogramu, který při správně zvoleném počtu košů eliminuje některé nepřesnosti ve vstupních datech. Při velikost obrazu $n \times m$ základní podoba této metody generuje $n + m$ příznaků. Ukázka principu metody je na obrázku 2.2.



(a) Vstupní obraz (b) Vertikální projekce (c) Horizontální projekce

Obrázek 2.2: Ukázka horizontální a vertikální projekce pro binární obraz znaku A. Vstupní matice je velikosti 4×4

Zónování

Při zónování je binární obraz rozdělen do matice, kde každý její prvek odpovídá určitému výseku originálního obrazu. Například obraz velikosti 64×64 pixelů lze rozdělit do matice o velikosti 16×16 zón, z nichž každá odpovídá části původního obrazu o velikosti 4×4 pixelů. Zóny, které jsou takto získány se sekvenčně prochází ve 4 směrech (zleva doprava a zprava doleva pro řádky, odshora dolů a zdola nahoru pro sloupce). Pokud je při průchodu nalezena oblast černých pixelů, je jako příznak použit poměr vzdálenosti prvního černého pixelu oblasti od okraje zóny (ve směru aktuálního průchodu) a celkového rozměru obrázku v daném ose. V případě, že při průchodu daným řádkem nebo sloupcem, není nalezen oblast černých pixelů, hodnota příznaku je rovna nule. [16, 14] Pro obrázek s rozměry $n \times n$, který byl rozdělen do N zón, jejichž rozměry jsou $m \times m$, tedy pro velikost získaného vektoru příznaků M platí: $M = 4Nm$.

Geometrické momenty

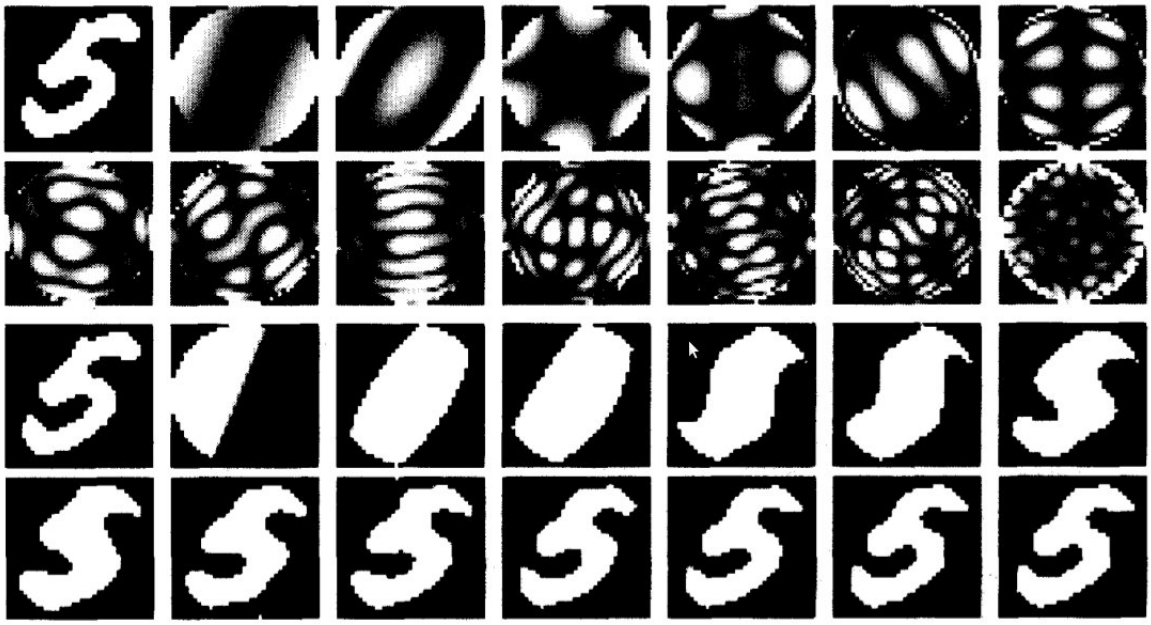
Obrazové momenty obecně se používají pro popis objektů v obraze. Matematicky je lze vyjádřit jako projekce obrazové funkce do polynomiálních bází. Geometrické momenty jsou potom jedním ze speciálních případů obecných obrazových momentů. Jejich hodnoty udávají určité zajímavé charakteristiky v obraze. [8] Pro účely extrakce rysů v systémech pro rozpoznávání ručně psaného textu se nejčastěji používá sedm momentů invariantních k posunutí, rotaci a změně měřítka. Ty byly prvně představeny v [6] a podle jejich autora jsou označovány jako *Huovy momenty*.

Zernikovy momenty

Reprezentace obrazu geometrickými momenty se nejeví jako nejvhodnější s ohledem na redundanci uchovávaných informací. Právě to bylo motivací pro vytvoření Zernikových momentů, které v roce 1980 představil M. R. Tegue [9]. Zernikovy momenty jsou získány z komplexních polynomiálních funkcí, známých jako Zernikovy polynomy. Zernikovy momenty překonávají zmíněné slabiny obecných geometrických momentů a jsou ortogonální, invariantní k rotaci a navíc mají tu vlastnost, že z nich může být rekonstruován původní obraz. Další výhodou Zernikových momentů je jejich vysoká odolnost vůči šumu v obraze, především pak u momentů nižších řádů. Zernikovy momenty je možné definovat jako sadu komplexních ortogonálních polynomů definovaných ve vnitřní oblasti jednotkové kružnice $x^2 + y^2 = 1$. [4] Srovnávací studie Zernikových momentů a sedmi invariantních Huových momentů ukázaly, že prvně zmíněné dosahují lepších výsledků, především pak při extrakci rysů z obrazů s vysokým stupněm šumu [13]. Na obrázku 2.3 jsou ukázány binární obrazy získané ze Zernikových momentů a binární obrazy vzniklé rekonstrukcí obrazu podle Zernikových momentů.

2.2.5 Klasifikace znaků

Základní myšlenkou klasifikace je rozdělení objektů do odpovídajících tříd v závislosti na rysech, které je popisují. [11] Pro její učely byla vyvinuta řada metod. Jednou z nejpoužívanějších je využití umělých neuronových sítí. Právě rozpoznávání písma bylo přitom jednou z prvních oblastí aplikace umělých neuronových sítí [15]. Druhý přístup využívaný ke klasifikaci vychází ze skrytých Markovových modelů (zkráceně HMM – Hidden Markov Models).



Obrázek 2.3: Řádky 1–2 ukazují nejprve zdrojový obrázek číslice 5 a poté zobrazení získaných Zernikových momentů v binárním obraze pro Zernikovy momentu řádu 1–13. Řádky 3–4 obsahují na prvním místě vstupní obrázek a poté obrazy vstupu rekonstruované s využitím Zernikových momentů řádu 1–13. Zdroj: [2]

2.2.6 Postprocessing

Závěrečnou fází rozpoznávání textu je postprocessing. Ten může být charakterizován jako dodatečné zpracování a úprava klasifikovaných dat před jejich poskytnutím na výstup. Postprocessing je prováděn na základě znalosti formátu výstupu a nejčastějších chyb zanesených v průběhu rozpoznávání textu. Cílem je tyto chyby opravit a nedistribuovat na výstup. Jako příklad lze uvést úpravu dvou za sebou jdoucích písmen i na u. Základním východiskem uvedené úpravy je fakt, že v českém jazyce se dvě i vedle sebe nevyskytují, a zároveň při segmentaci znaků často dochází k rozdělení písmene u právě na dvě i.

K sofistikovanějším přístupům k dodatečnému zpracování patří slovníkové metody, kdy je výstup poskytnutý klasifikátorem opraven na základě dostupných slovníkových dat pro očekávaný jazyk výstupu. Fáze postprocessingu je volitelná a není vždy v systému zahrnuta.

2.3 Trénovací a testovací množina

Jak již bylo nastíněno v úvodu, pro klasifikaci znaků (podrobněji popsáno v sekci 2.2.5) je nutné vhodně inicializovat algoritmus klasifikace. Tato inicializace je bežně označována jako učení. Během něj je využito právě trénovací, potažmo testovací, množiny. V případě rozpoznávání ručně psaných znaků si trénovací množinu lze představit jako soubor prvků obsahujících jednotlivé obrazové předlohy znaků doplněné o jejich *anotaci* (informaci o tom, jaký znak ve skutečnosti reprezentují). V textu dále bude trénovací a testovací množina označována souhrnným označením datová sada.

Datová sada může sloužit i jako benchmark – jejím prostřednictvím lze srovnávat úspěšnost různých přístupů k rozpoznávání ručně psaného textu.

2.3.1 Požadavky na trénovací a testovací množinu

Vzhledem ke specifickému využití dat z trénovací, popř. testovací množiny jsou na tyto množiny kladeny určité požadavky. Nejdůležitější z nich jsou:

- Počet předloh pro jednotlivé znaky by se neměl výrazně odlišovat.
- Jednotlivé předlohy pro tentýž znak by měly být navzájem co nejvíce odlišné – tak, aby byla co nejúplněji reprezentovány hodnoty, jichž informace (uložená v tomto případě v obrazové formě) může pro jeden znak nabývat.
- Trénovací množina by neměla být příliš velká (poté některé přístupy využívané pro klasifikaci ztrácejí schopnost generalizace a špatně reagují na znaky, které jim při učení nebyly předloženy)

2.4 Přehled vybraných datasetů

Dle výše uvedených informací je zřejmé, že datová sada je pro klasifikaci (a tedy i rozpoznávání ručně psaného textu) nezbytná. Není proto divu, že na internetu lze nalézt mnoho různých datových sad určených pro rozpoznávání ručně psaných znaků, případně čísel. Na následujících řádcích budou blíže popsány vybrané datové sady, aby byl přiblížen přístup, jaký se obecně volí pro jejich vytvoření.

2.4.1 THE MNIST DATABASE of handwritten digits

Jedná se pravděpodobně o nejznámější datovou sadu pro rozpoznávání ručně psaných číslic. Byla vytvořena z větší datové množiny – NIST. Obsažené číslice jsou vycentrovány v šedotónových (ovšem získaných z obrázků černobílých) obrázcích pevné velikosti. Obsahuje 60000 unikátních vzorů od 250 pisatelů v trénovací množině a 10000 vzorů v testovací množině od rozdílných 250 pisatelů. Data jsou umístěna v souborech vlastního formátu a kromě anotace dané číslice neobsahují žádné další podpůrné informace. Na webové stránce [18], kde je tato sada ke stažení, se nachází rozsáhlý přehled algoritmů použitých pro klasifikaci číslic z poskytnuté sady i s jejich dosaženou úspěšností.

2.4.2 OCR dataset

Tato datová sada dostupná z [12] obsahuje normalizované binární obrazy ručně psaných znaků. Data i jejich anotace (a další charakteristika) je uložena v jediném souboru vlastního formátu. Krom samostatných písmen obsahuje databáze také jednotlivá slova, která jsou s písmena provázána – písmeno vždy obsahuje index na slovo, ze kterého bylo extrahováno. Dále je zaznamenán také znak, který následuje, nikoli však předchozí znak.

2.4.3 Letter Image Recognition Data

Poslední datová sada, která bude zmíněna je Letter Image Recognition Data. Jsou v ní obsažena velká písmena anglické abecedy. Tato písmena jsou rovnoměrně distribuována, každé z nich má v sadě přibližně 800 výskytů. Celkem se jedná přesně o 20000 znaků. 16000 z nich je vyčleněno do trénovací množiny, zbývající 4000 tvoří množinu testovací. Každý prvek množiny je popsán pouze s využitím rozsáhlých statistických informací získaných z jeho binárního obrazu (celkem jde o 16 rysů). [3]

Kapitola 3

Návrh systému

V této kapitole bude popsán vlastní návrh systému pro poloautomatickou tvorbu databáze ručně psaných znaků. Navrhovaný program bude sestávat ze tří částí. První z nich zajišťuje automatickou segmentaci znaků a vychází z architektury OCR systému představené v kapitole 2. Je zde využito všech fází vztahujících se k segmentaci textu. Tato část poskytuje uživateli částečně připravený text určený k dalšímu (manuálnímu) zpracování. Právě to je hlavním pojícím prvkem druhé části. Ta sestává především z grafického uživatelského rozhraní, které uživateli poskytuje možnosti korigovat automatický výstup první části – upravit segmentaci znaků v těch místech, kde algoritmus pracující automaticky chyboval. Jak již bylo zmíněno výše, navrhovaný systém uvažuje i možnost zpracování textu s vyskytujícími se diakritickými znaménky. S tímto faktem je třeba při návrhu systému počítat a jednotlivé metody navrhovat tak, aby si se vstupem v češtině a všemi aspekty, které s tím souvisejí, dokázal poradit. Třetí zamýšlenou částí systému je tvorba výsledného XML dokumentu popisujícího umístění jednotlivých znaků na stránce a jejich další atributy. Struktura výstupního XML dokumentu je přiblížena v příloze D. Vstupem do systému bude tedy strana ručně psaného českého textu ve formě rastrového obrázku a výstupem bude XML dokument popsaný výše. Výhodou tohoto přístupu je zachování původní obrazové informace doplněné i o prostorové vazby jednotlivých znaků. To je důležité především kvůli existenci různých přístupů k rozpoznávání ručně psaného textu, z nichž většina sice v současnosti operuje s binárními obrazy, nicméně je možné, že se situace v budoucnu změní. Současné datové sady z velké většiny poskytují právě pouze binární obrázky a kvalitní datová sada lépe zachovávající původní obrazovou informaci je tak o to cennější.

3.1 Automatické zpracování

Jak již bylo zmíněno, automatické zpracování vstupních dat tvoří jednu ze tří hlavních částí navrhovaného systému. Automatické zpracování lze rozdělit do několika fází, které na sebe postupně navazují, přičemž výstup většiny z fází lze korigovat v grafickém uživatelském rozhraní. Nejprve je provedeno předzpracování vstupních dat, na něj navazuje segmentace řádků, segmentace slov a nakonec segmentace jednotlivých znaků. Zmíněné fáze jsou blíže popsány dále.

3.1.1 Předzpracování vstupních dat

Předzpracování vstupu probíhá v několika fázích. Nejprve je vstupní obrázek převeden do odstínů šedé. V takto získaném obrazu je provedeno filtrování s cílem odstranit šum,

který se během pořizování vstupního obrazu mohl do tohoto obrazu zanést, a vyhladit linku písma. Na obraz jsou za tímto účelem aplikovány dva filtry – Gaussův a mediánový. Vyhlazování obrazu má nicméně za následek ztrátu části obrazové informace. Proto je potřeba dobře zvolit velikost použitého filtru. Pokud by byl obrazový filtr pro vyhlazování příliš velký, měla by jeho aplikace důsledek nejen odstranění šumu, ale zároveň i nárust tloušťky linky písma, v důsledku čehož by došlo k zahlazení rozdílů mezi některými písmeny (typicky například pro dvojici písmen *i,e*). Toto zkreslení by bylo pro další zpracování, především pak pro fázi segmentace znaků, velmi nežádoucí. Výstupem předzpracování vstupních dat je v navrženém systému binární rastrový obrázek. Ten je z monochromatického vstupního obrázku získán s využitím *prahování*. Obecně lze metody prahování rozdělit na dva typy. Prvním je prahování automaticky určeným prahem a druhým pak prahování s svyužitím pevně nastaveného prahu. Obě tyto možnosti byly během návrhu systému prozkoumány a vzhledem k neuspokojivým výsledkům automaticky určeného prahu bylo zvoleno prahování s pevně nastavenou hodnotou prahu na úrovni 210.

I přes aplikaci filtrů zůstane v obraze několik shluků černých pixelů představujících šum. Ty jsou odstraněny později v průběhu zpracování s využitím konstanty vymezující minimální velikost spojitého objektu.

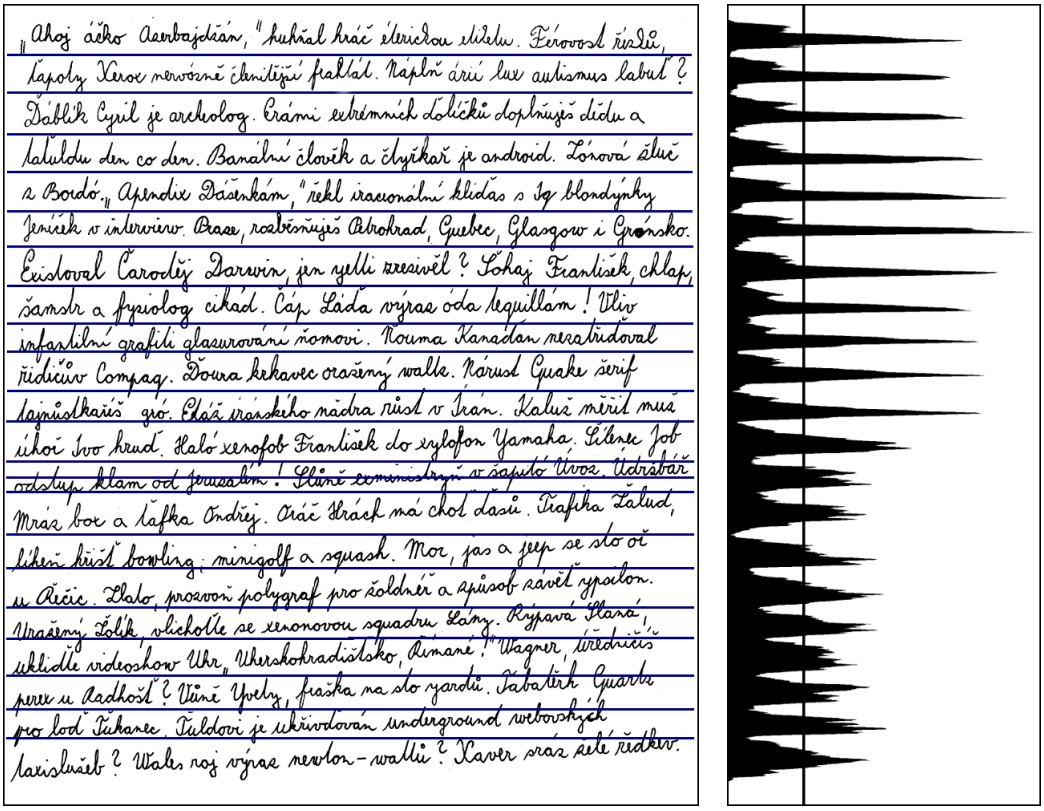
3.1.2 Segmentace řádků

Vstupem pro fázi segmentace řádků je binární rastrový obraz. Výstupem jsou potom obrazy jednotlivých předpokládaných řádků textu ve vstupním obraze. Jelikož pro získání jednotlivých podobrazů řádků je potřeba znát jejich ohraničení, je nutno navrhnout algoritmus pro správné určení horizontálních čar, které rozdělují jednotlivé řádky. Tyto myšlené čáry budou v textu dále označovány jako hranice řádku.

Pro jejich výpočet je využito horizontální projekce vstupního obrazu. Tu lze chápat jako histogram, jenž obsahuje pro každý řádek vstupního binárního obrazu počet hodnot černých pixelů na tomto řádku. Ukázka vstupního binárního obrazu a výsledné horizontální projekce je na obrázku 3.1. Z tohoto obrázku je zřejmé, že hranice řádků se nacházejí někde mezi nejvýraznějšími vrcholy (oblastmi, kde doposud rostoucí hodnoty histogramu začnou klesat) histogramu (dále budou označovány jako špičky). Vzhledem k tomu je nutné určit, které z vrcholů histogramu budou považovány za špičky, mezi nimiž se nachází hranice řádku. Za tímto účelem je určen práh (na obrázku 3.1 vykreslen jako vertikála nacházející se přibližně ve třetině obrázku), při jehož překročení je vrchol histogramu zpracován jako špička. Jeho hodnota je zvolena jako průměr všech hodnot histogramu násobený konstantou. Velikost této konstanty byla heuristicky určena jako $1,4$. Pro naprostou většinu vyzkoušených vstupních obrazů takto zvolený práh správně rozděluje vrcholy histogramu od špiček.

Dále je nutné uvažovat možnost, že se i nad hodnotou prahu vyskytnou úzké vrcholy v blízkosti špiček (typicky oblast, v níž se na řádku nachází diakritika), které pro stanovení hranic mezi řádky nelze uvažovat. Tyto odchylky jsou odfiltrovány dle relativní šířky vrcholu v hodnotě prahu (vynechá se, pokud je špička ve srovnání s jinými příliš uzká) a dle velikosti oblasti mezi dvěmi špičkami (stejné kritérium jako pro šířku špičky). Jakmile jsou nežádoucí špičky odfiltrovány, je možné stanovit hranici mezi řádky jako pořadí nejnižší hodnoty histogramu (číslováno shora) mezi dvěmi špičkami.

Jakmile jsou určeny hranice mezi řádky, je možné začít s vytvořením podobrazů jednotlivých řádků. Ty ovšem nejsou získány pouhým rozděleným vstupního obrazu dle získaných hranic. Takové řešení by totiž vedlo k rozdělení téhož znaku, nacházejícího se na hranici,

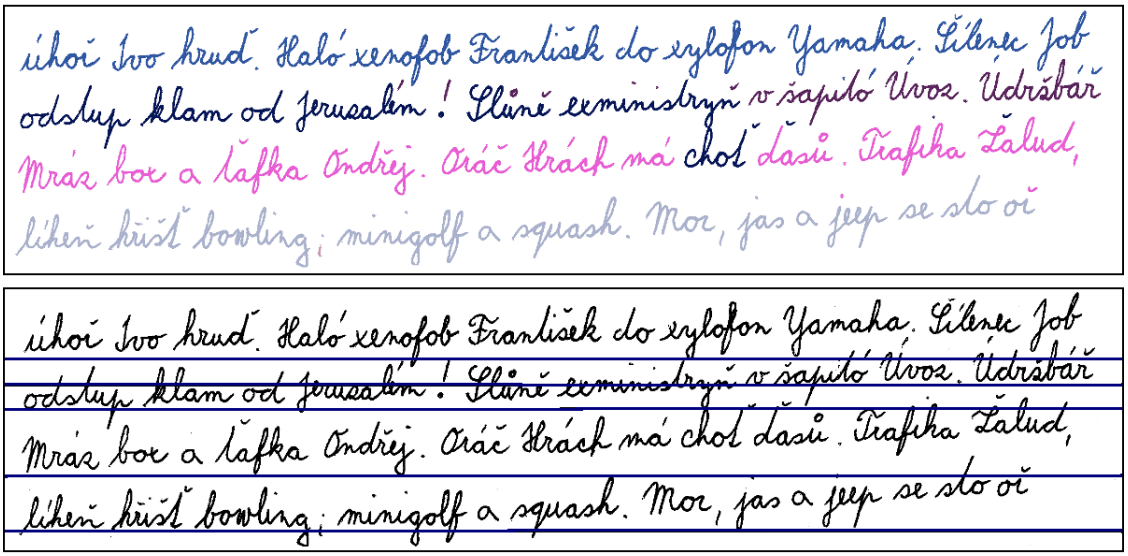


Obrázek 3.1: Ukázka binárního obrazu vstupujícího do segmentace řádků s automaticky stanovenými hranicemi řádků a horizontalní projekce, podle níž byly hranice stanoveny

do dvou různých řádků. Takové chování je přirozeně nežádoucí, a proto je pro rozdělení použit složitější algoritmus. Ten spočívá v nalezení všech spojitých komponent, které se, zcela nebo částečně nacházejí v oblasti určené hranicemi řádku. Tyto spojité komponenty jsou posléze rozděleny do řádků do řádků v závislosti na procentuálním zastoupení pixelů spojité komponenty na řádcích (spojitá komponenta je přidělena do řádku, v němž se nachází nejvíce pixelů, které tuto komponentu tvoří). Výhody i nevýhody zvoleného přístupu dobře ilustruje obrázek 3.2. Na něm si lze povšimnout, že i navzdory hranicím řádků procházejícími skrz některé znaky nejsou tyto znaky rozseknuty do různých řádků a je naopak zachována celistvost znaku. Hlavní nevýhodou je pak především fakt, že pokud se protínají linky znaků nacházejících se na různých řádcích, jsou objekty určené protínajícími se linkami považovány za jedinou spojitou komponentu a ta je začleněna do jednoho řádku. Dalším problémem může být nepřesné rozdělení diakritiky do řádků – a to především pro velká písmena, pro než se diakritické znaménko nachází vysoko na řádku. S ohledem na určení programu, především pak na fakt, že je předpokládán vstup uživatele, který určité nepřesnosti automatického rozdělení může korigovat, je ovšem možné tyto nedostatky navrženého řešení tolerovat.

3.1.3 Segmentace slov a znaků

U segmentace slov a znaků se pro určení hranic mezi slovy, popřípadě jednotlivými znaky, využívá vertikální projekce. Její charakteristika se od horizontální projekce popsané výše



Obrázek 3.2: Ukázka vybraného úseku s rozdělením do řádků a výsledné rozdělení spojitých komponent do jednotlivých řádků (barevně rozlišeno)

liší pouze v tom, že histogram s hodnotami černých pixelů se sestavuje na základě jejich výskytu ve sloupcích binárního obrazu.

Obrázek 3.3: Ukázka vertikální projekce pro vstupní obraz řádku, a výsledná segmentace slov

Vstupem pro fázi segmentace slov jsou obrazy jednotlivých řádků textu. Jako hranice textu jsou v tomto případě určeny ty indexy histogramu (tedy souřadnice na ose x), kde dochází ke změně z nenulové hodnoty na nulovou. Zjednodušeně lze tedy říci, že hranice slov jsou v binárním obrazu řádku rozmístěny za každým objektem, za nímž následuje vertikála, na které se nenachází žádný černý pixel. Pro naprostou většinu vstupních textů dává tento jednoduchý algoritmus dobré výsledky, jak ostatně ukazuje také obrázek 3.3. Jeho výhodou je navíc možnost jednoduše získat obrázky slov jako části obrazu řádku v oblasti vymezené levou a pravou hranicí slova, aniž by hrozilo zahrnutí přesahu linky některého ze sousedních slov.

Právě na takto získané obrazy slov je následně aplikován algoritmus pro segmentaci jednotlivých znaků. Jak již bylo zmíněno, pro stanovení hranic je stejně jako u segmentace slov využito vertikální projekce. Ještě předtím, než je zavolán algoritmus pro stanovení hranic mezi znaky, jsou s obrazem slova provedeny morfologické operace – dilatace a následně otevření. Cílem je zvýraznit v obrazu slova oblasti, kde jsou linky písma blízko u sebe, a kruhové oblasti znaků (například u znaků `a, o`) a vylepšit tak charakteristiku vertikální

Obrázek 3.4: Ukázka vertikální projekce pro původní obraz slova, obraz po provedení morfologických operací, jeho vertikální projekce a výsledná segmentace znaků

projekce obrazu, která je dále využita pro hledání hranic mezi znaky. Obrázky 3.4 a 3.5 ilustrují změnu v originálním obraze slova i v jeho vertikální projekci po provedení morfologických operací.

Samotný algoritmus pro nalezení hranic mezi znaky slova je principiálně podobný algoritmu využitému pro určení hranic mezi řádky. Vzhledem k vyšší komplexnosti řešeného problému jsou hranice hledány nejen s využitím vertikální projekce obrazu slova po provedení popsaných morfologických operací, ale i s využitím vertikální projekce původního obrazu slova. V upraveném obrazu jsou znovu nalezeny špičky, ovšem tentokrát jsou pro jejich určení využity kromě prahu (v úrovni průměrné hodnoty histogramu násobené konstantou $0,8$) i další charakteristiky. Jedná se konkrétně o průměrnou šířku mezer mezi špičkami (po prahování) násobenou heuristicky určenou konstantou 0.5 a počet průsečíků vertikály s linkou slova. V horizontálním prostoru celého slova (určeném souřadnicemi na ose x) se určí mezery mezi špičkami splňující stanovený limit (viz výše) a v nich jsou hledány oblasti, které protínají linku slova co nejméněkrát. V takto určených oblastech se potom nalezne hranice znaku. Její hledání však už probíhá ve vertikální projekci původního (neupravovaného) obrazu slova – a to tak, že je, stejně jako u segmentace řádků, za hranici určen index histogramu (odpovídající pozici na ose x), kde je hodnota histogramu minimální.

Obrázek 3.5: Ukázka vertikální projekce pro původní obraz slova, obraz po provedení morfologických operací, jeho vertikální projekce a výsledná segmentace znaků

3.2 Grafické uživatelské rozhraní

Součástí programu vytvořeného v rámci praktické části bakalářské práce je i grafické uživatelské rozhraní, které je využito pro zásahy uživatele (jeho korekce automatického algoritmu). Toto uživatelské rozhraní vzhledem k tomu, že je navrženo specifické použití úzce vymezenou skupinou uživatelů, u nichž je vzhledem k určení programu předpokládána vysoká zdatnost v použití počítače, do určité míry porušuje klasické přístupy k tvorbě grafického uživatelského rozhraní (dále v textu označováno jako GUI). Jelikož je předpokládáno využití programu pro anotování velkého počtu dat, není kladen přílišný důraz na rychlou orientaci uživatele v prostředí programu, ale spíše na rychlost, s jakou uživatel, který si osvojí ovládání programu, dokáže anotovat data. Dále je v navrženém GUI kladen důraz na co nejvyšší míru interaktivity. Všechny nutné korekce a zásahy uživatele se dějí přímo v oblasti, do níž je vykreslován průběžně upravovaný obrázek. Ovládání je tak maximálně spjato přímo se zobrazovanými informacemi. Není využito žádných ovládacích prvků, mezi nimiž by bylo nutné přejíždět kurzorem myši. Cílem je znovu dosažení co nejvyšší rychlosti

práce s vytvořeným programem. V krátkosti lze tedy tři nejdůležitější zásady dodržené pro návrh a tvorbu výsledného GUI shrnout v následujících bodech:

- Rychlost zpracování má přednost před uživatelskou přívětivostí
- Minimum ovládacích prvků na obrazovce – využití kláves a tlačítek myši
- Důraz na interaktivitu

Během návrhu GUI bylo potřeba dobře zvážit, ve kterých fázích automatického zpracování bude vyžadován zásah uživatele a jakým způsobem budou fáze provázány. Ve výsledku program funguje tak, že každá z částí automatického zpracování s výjimkou předzpracování je korigována uživatelem. Posloupnost akcí kopíruje automatické zpracování. Jeho výstup je zobrazen uživateli v GUI a je očekávána jeho korekce. Takto zkorigovaná data jsou potom dále automaticky zpracována.

Vzhledem k faktu, že pro vytvoření datové sady je nutné anotovat velké množství dat (a tedy velké množství obrázků ručně psaného textu), byla při návrhu zvážena možnost vytvoření sady nástrojů, z nichž každý by sloužil pro korekci jediné fáze automatického zpracování. Cílem tohoto opatření mělo být zvýšení rychlosti zpracování. Uživatel by totiž jednu činnost prováděl opakovaně pro větší množství souborů. Nicméně po otestování tohoto přístupu na prvních dvou fázích zpracování (ruční předúprava vstupu a korekce segmentace řádků) bylo od tohoto záměru upuštěno. Uživatelé totiž při opakování téže činnosti dosahovali pomalejších časů než při střídání činností. Výsledkem praktické části je tak program, který provede pro jeden vstupní obraz všechny fáze zpracování a až poté je možné začít se zpracováním dalšího vstupního obrazu.

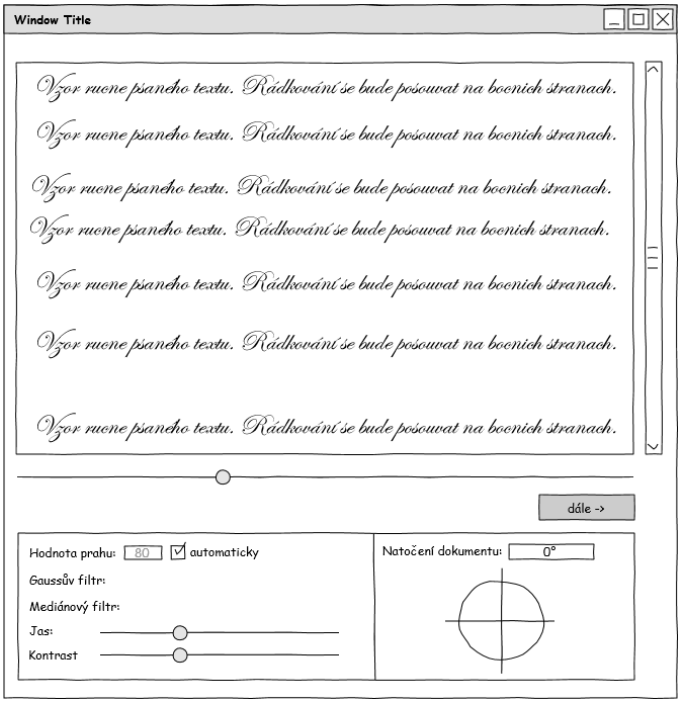
Výsledné GUI je tvořeno čtyřmi částmi. Největší plochu zabírá vykreslovací plocha, v níž se zobrazují informace (v tomto případě především obrazové) spojené s jednotlivými fázemi zpracování vstupu. Pod vykreslovací plochou je umístěna lišta, která uživatele informuje o průběhu zpracování (které fáze byly úspěšně dokončeny a které fáze budou následovat). V pravé části okna se nacházejí dvě editační pole. Horní z nich slouží pro vložení anotace daného obrázku, spodní potom zobrazuje nápovědu. Ta byla vytvořena a zahrnuta přímo do hlavního okna především pro usnadnění prvních kroků při práci s programem. Jsou v ní popsány akce, které je možno podniknout v dané fázi zpracování a způsob, jakým mohou být provedeny. Video zobrazující konečnou podobu grafického uživatelského rozhraní a způsob práce s ním je možno nálézt na přiloženém DVD.

V následujících odstavcích jsou krátce popsány nejvýznamnější fáze zpracování v grafickém uživatelském rozhraní.

3.2.1 Ruční předúprava vstupu

Po načtení vstupního obrázku je tento před započetím jeho dalšího zpracování nejdříve ručně upraven. Tato úprava spočívá v korekci případné rotace dokumentu a výběr oblasti (ve formě oříznutí), která bude dále zpracovávána. Důvody pro zařazení této fáze jsou dva. Prvním je zlepšení přesnosti automatické segmentace (při zrotovaném dokumentu přesnost klesá), druhým potom způsob předpokládané pořízení vstupních obrázku skenováním. Takto vzniklé dokumenty jsou občas nejen mírně zrotované, ale často mají tmavší okraje v místech přechodu mezi skenovaným listem a pozadím. Pro nastavení natočení dokumentu byl původně navržen samostatný ovládací prvek (viz obrázek 3.6). Ovšem po vykrystalizování filosofie vznikajícího GUI bylo od tohoto řešení upuštěno a původně zvažovaný

ovládací prvek pro rotování dokumentu byl nahrazen možností přímo v obraze určit vodorovnou linku písma. Podle této linky je následně vypočten úhel, o který je nutné dokument zrotovat. Získaný výstup z fáze ruční předúpravy je dále při zpracování použitý jako vstup do automatické fáze segmentace řádků.



Obrázek 3.6: Ukázka původně zvažované podoby uživatelského rozhraní – fáze předúpravy

3.2.2 Oprava rozdělení do řádků

Po automatickém rozdělení textu do řádků je, vzhledem k některým neduhům popsaným v sekci 3.1.2 (nejčastěji diakritice špatně rozmístěné do řádků), nutné navržené hranice některých řádků upravit. Ve vykreslovací oblasti okna se v této fázi objeví ohraničené samostatné obrazy jednotlivých řádků a uživatel má možnost je nakreslením nové dělící hranice rozdělit, případně označit sousedící řádky a ty spojit. Každá úprava vyvolá okamžité překreslení řádků podle nově zvolených hranic. Výstupem této fáze zpracování jsou upravené obrazy řádků, které jsou dále automaticky segmentovány na slova.

3.2.3 Oprava segmentace slov a znaků

Navržené hranice slov potom znovu putují do GUI pro korekci od uživatele. Při této fázi je ve vykreslovací oblasti stále zobrazeno rozdělení do řádků z předchozí fáze, navíc jsou ale do jednotlivých řádků doplněny i hranice mezi slovy. Před započetím úprav hranic je možné nakreslením linky do libovolného řádku určit sklon písma. Pokud se tak stane, jsou všechny obrazy slova zkoseny podle zvoleného sklonu a na výsledné obrázky je znovu zavolána segmentace slov. Pokud je sklon písma určen správně, vede tento úkon k výraznému zvýšení úspěšnosti automatické segmentace slov a znaků. Jakmile je uživatelem potvrzeno dokončení úprav hranic mezi slovy, jsou na jejich základě vytvořeny obrazy slov, které

vstupují do automatické segmentace znaků. V GUI se to projeví přibytím automaticky navržených hranic do vykreslených obrazů řádků.

Vzhledem k vysokému počtu hranic (především těch mezi znaky), bylo u této fáze potřeba minimalizovat úsilí, které musí uživatel vynaložit pro opravu nepřesností jejich automatického určení. Pro naplnění tohoto cíle bylo zvoleno stanovení hranic s využitím kurzoru myši a jejího levého tlačítka (přidání hranice v místě kliku), popř. pravého tlačítka (odebrání hranice nejbližší místu kliku).

3.2.4 Anotace znaků

Fáze anotace znaků se spustí ve chvíli, kdy uživatel potvrdí dokončení korekce hranic mezi znaky. Pro samotnou anotaci je využito editační pole, do něhož se vkládá text nacházející se ve vstupním obraze. Uživatel má možnost zvolené znaky vynechat z výstupu pokud je nahradí znakem `'_'`.

3.3 Generování výsledného XML dokumentu

Jak již bylo zmíněno, výstupem programu je XML dokument. Tento formát výstupu byl zvolen především proto, že je díky němu tak v maximální míře zachována původní obrazová informace. Vyřezání jednotlivých znaků, případně jejich binarizace (jako se tomu děje u většiny nejpoužívanějších datových sad pro rozpoznávání ručně psaného textu) snižuje informační hodnotu datové sady, kterou takové znaky reprezentují. Vzhledem k tomu, že v budoucnu se metody pro rozpoznávání ručně psaného textu budou stávat stále sofistikovanějšími, lze předpokládat, že tyto metody budou potřebovat i bohatější vstupní informaci, se kterou by mohly pracovat. A právě tu lze při zvoleném přístupu poskytnout.

Výstupní XML soubor popisuje umístění jednotlivých znaků. Kromě toho jsou pro širší možnosti využití vytvořené datové sady do XML výstupu zahrnuty i sousední znaky (předchozí a následující znak). Vzhledem k informacím získaným během korekcí uživatele by bylo možné přidat i další charakteristiky jako například uživatelem nastavený sklon dokumentu, vybraná oblast pro anotaci, řádek na němž se znak nachází apod. Tyto charakteristiky nicméně do finálního výstupu nejsou začleněny s ohledem na skutečnost, že při rozpoznávání textu, pro nějž bude vytvořená sada využívána, není předpokládaným vstupem podobná informace – a ta by tak postrádala jakékoli využití.

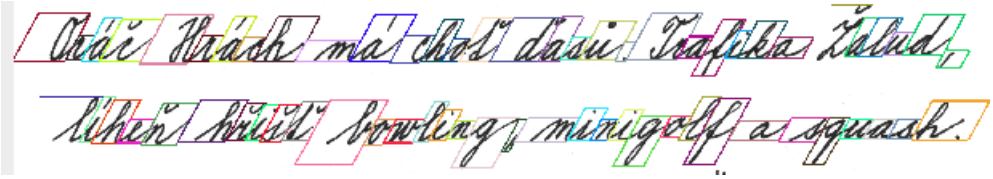
3.3.1 Zaznamenávané charakteristiky

Už v úvodu sekce bylo naznačeno, že charakteristikami zaznamenanými ve výstupním XML souboru jsou:

- anotace znaku
- čtyřúhelník ohraničující anotovaný znak ve vstupním obraze
- anotace předchozího znaku a znaku následujícího

Je zřejmé, že anotace znaku je charakteristika, která nelze opomenout, stejně jako jeho umístění ve vstupním obraze. To je popsáno ohraničujícím čtyřúhelníkem. Ovšem získání souřadnic jeho bodů není, vzhledem k transformacím prováděným během zpracování vstupu, jednoduché. Souřadnice je totiž potřeba od provedených transformací očistit tak, aby správně určily čtyřúhelník vymezující daný znak v originálním obraze. Za tímto účelem

je postupně pro bod v transformovaném obraze provedeno odečtení hodnoty levého horního bodu obdelníku, který určil ořez obrázku, a poté rotace bodu nazpět o úhel natočení dokumentu. Výsledné ohraničení jednotlivých anotovaných znaků ve vstupním obrazu je znázorněno na obrázku 3.7.



Obrázek 3.7: Ukázka výsledného ohraničení jednotlivých znaků

Účelem uchování anotací sousedních znaků je lepší informace o obraze reprezentujícím anotované písmeno. Tvar písmena je totiž i u téhož pisatele jiný, pokud se jedná o písmeno na začátku slova, nebo na konci slova. Dále je tvar ovlivněn také sousedním písmenem samotným. Příkladem může být písmeno `k`, které má jiný tvar pokud následuje po `o` (pak je navázáno v okolí střední dotažnice) a jiný tvar pokud následuje po `a` (v tomto případě navazuje už od základní dotažnice). Všechny tyto informace mohou být pro rozpoznávání ručně psaného textu cenné, a proto jsou zahrnuty i ve výstupu programu.

Podrobná ukázka výsledného XML dokumentu s uvedenými charakteristikami je uvedena v příloze D.

Kapitola 4

Popis implementace

V této kapitole budou popsány vybrané části implementace systému navrženého v kapitole 3. Nejprve jsou krátce popsány nástroje využité pro tvorbu výsledného programu. Poté následuje popis struktury programu, kde budou představeny vybrané třídy implementovaného systému a způsob, jakým navzájem spolupracují. Program byl vyvíjen na platformě Linux, ovšem vzhledem k přenositelnosti zdrojového kódu v C++ i multiplatformitě použitých knihoven je program možné přeložit a spustit i v systému MS Windows. Právě pro tuto platformu byl vytvořen spustitelný soubor, který se společně s potřebnými dynamicky linkovanými knihovnami nachází na přiloženém disku. Důvodem je především snadnější demonstrace výsledného programu i na systémech, na nichž nejsou nainstalovány potřebné knihovny.

4.1 Použité nástroje

Systém je implementován v programovacím jazyce `C++` s využitím knihovny pro počítačové vidění OpenCV ve verzi 2.4 a frameworku Qt ve verzi 4.8 pro tvorbu grafického uživatelského rozhraní. Knihovna OpenCV je využita především pro automatické zpracování a segmentaci. Byla vybrána s ohledem na její velmi dobrou použitelnost pro podobné aplikace (poskytuje pokročilé funkce pro zpracování obrazu). Knihovna OpenCV je, stejně jako framework Qt, multiplatformní a s jejich využitím je tedy možné psát aplikace přenositelné mezi systémy.

4.2 Struktura programu

Struktura programu vychází z návrhu popsaného v kapitole 3. Pro každou z jednotlivých fází automatického zpracování i zpracování v GUI je vytvořena obslužná třída. Dále program obsahuje také nejrůznější pomocné třídy, ale protože tyto netvoří základní kostru programu, budou v následujícím stručném popisu struktury programu zmíněny jen ty nejdůležitější z nich.

4.2.1 Automatické zpracování

Část programu zajišťující automatické zpracování se skládá z několika tříd, které vytvářejí hierarchickou strukturu podobnou architektuře obecného systému pro rozpoznání ručně psaného textu popsané v sekci . Jedná se konkrétně o třídy `Page`, `Line`, a `Word`. Zpracování obrazu začíná v třídě `Page`, která reprezentuje stránku textu. V rámci této třídy

je obraz rozdělen na jednotlivé řádky a s ohledem na toto dělení je vytvořen vektor objektů `Line` reprezentujících jednotlivé řádky textu. Třída `Line` je implementována tak, aby v ní bylo možné dále získat slova nacházející se na daném řádku. Třída `Word` reprezentuje slovo. Při automatickém zpracování textu je tedy postupně obrázek zpracováván a segmentován v třídách `Page`, `Line`, a `Word`. Samotné funkce pro segmentaci jsou ovšem volány ze třídy `MainWindow`, která v rámci programu funguje jako řídící třída, která spravuje stav výpočtu a na jeho základě vyvolává příslušné akce. Kromě zmíněných tříd tvořících jádro části pro automatické zpracování, lze v této části nalézt také další pomocné třídy, které jsou hlavními třídami využívány pro určité jasně stanovené účely. Například třída `ConnectedComponent` reprezentuje jednotlivé body spojitého objektu a zapouzdřuje práci s tímto objektem, třída `Histogram` zase zapouzdřuje práci s vertikální a horizontální projekcí, užitou pro automatickou segmentaci (viz sekce 3.1).

4.2.2 Zpracování v GUI

Už v předchozí sekci byl zmíněn význam třídy `MainWindow`, na následujících řádcích bude podrobněji popsána funkčnost dalších důležitých tříd náležících do části programu, která se stará o zobrazení dat a interakci s uživatelem – tedy především tříd `Preprocessing`, `Segmentation` a `Annotation`. Třída `Preprocessing` obstarává korekci rotace vstupního dokumentu a jeho ořez. Výsledný obraz je předán objektu třídy `Page`. Ten je součástí `MainWindow` a ukazatel na něj je také v objektu třídy `Segmentation`. Právě v této třídě se totiž nejvíce prolíná vstup od uživatele s výstupem části programu obstarávající automatické zpracování. Jakmile jsou všechny fáze segmentace dokončeny, získají se z objektu třídy `Segmentation` hranice slov a znaků na jednotlivých řádcích dokumentu. Ty jsou spolu s indexy řádků, na nichž se nacházejí, předány do objektu třídy `Annotation`, kde je na základě těchto hranic vytvořen vektor objektů třídy AnnotationObject, která reprezentuje jeden znak spolu s jeho anotací. Zároveň zajišťuje zobrazení znaku ve scéně. Pokaždé, když se potom změní text v editačním poli pro vkládání anotace, je nový text předán třídě `Annotation`, která nastaví správnou anotaci pro všechny objekty reprezentující jednotlivé znaky.

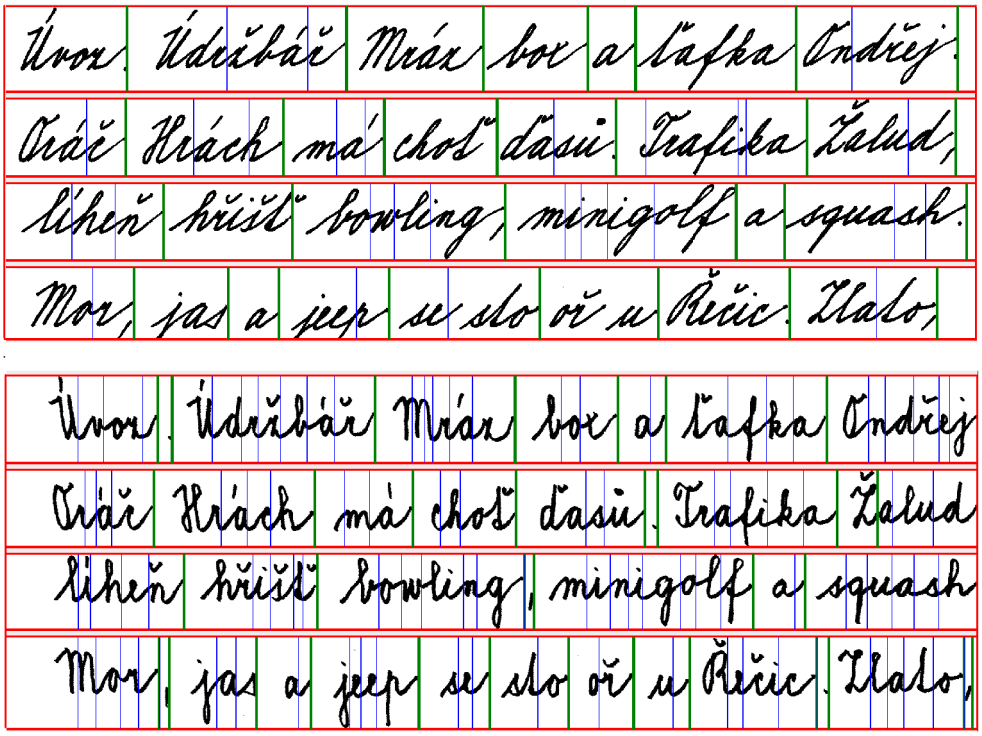
4.2.3 Export do XML

Každý instance třídy `AnnotationObject` implementuje metodu `getXML()`, která pro daný objekt vygeneruje část výstupního XML souboru. Tato metoda je ve třídě`Annotation` zavolána pro všechny objekty `AnnotationObject`. Díky tomu je získán výstup do XML souboru, který je předán do třídy `MainWindow`. Tato třída zajistí jeho vypsání do zvoleného souboru.

Kapitola 5

Výsledky a vyhodnocení

V následující kapitole je zhodnocena implementační část bakalářské práce a jsou představeny dosažené výsledky. U vytvořeného programu je zvlášť vyhodnocena část obstarávající automatickou segmentaci textu a zvlášť grafické uživatelské rozhraní. Pro jednodušší vyhodnocení obou částí programu bylo implementováno logování aktivity uživatele a automatický výpočet vybraných statistik vztahujících se k automatické segmentaci.



Obrázek 5.1: Na horním obrázku je vyobrazeno automatické určení hranic slov i písmen před úpravou sklonu, na dolním obrázku po provedení úpravy sklonu

5.1 Vyhodnocení automatické segmentace

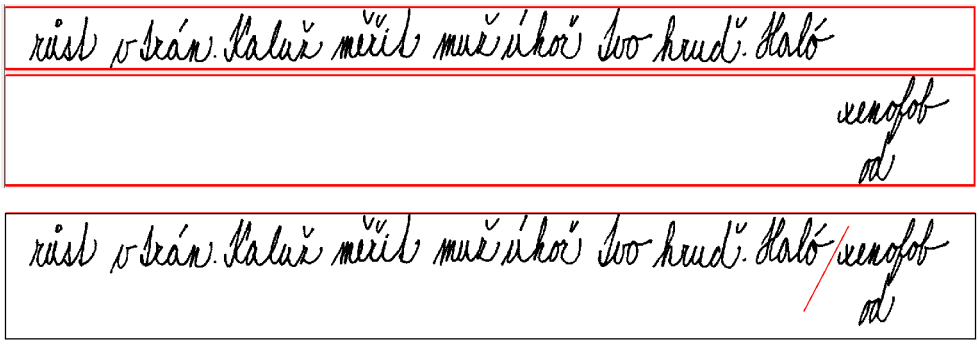
Zhodnocení algoritmu pro automatickou segmentaci proběhlo dvěma způsoby:

- Pro segmentaci řádků a slov byly vybrány reprezentativní případy jejich typického chování a ty graficky zachyceny,
- pro určení hranic mezi znaky byla automaticky vypočtena statistika na základě výstupu automatického zpracování a korekce uživatele (viz dále).

. Statistika byla tedy vypočtena pouze pro fázi segmentace znaků. Tento fakt má několik důvodů. Prvním z nich je fakt, že uživatel tráví při korekci výstupu této fáze největší objem času. Zvýšení úspěšnosti automatické segmentace znaků by tedy mělo mnohem vyšší efekt na zrychlení celkové doby nutné pro anotaci jedné stránky textu. Druhým důvodem je poměrně dobrá spolehlivost prvních dvou fází, přičemž obě selhávají ve specifických situacích, které jsou graficky velmi dobře ilustrovatelné.

5.1.1 Segmentace řádků

Jak již bylo zmíněno v sekci 3.1.2, kde je metoda segmentace podrobněji diskutována, nejčastějším problémem automatické segmentace řádků je zařazení diakritiky do nesprávného řádku a problematické naložení s protínajícími se linkami ležícími na různých řádcích. Oba případy velmi dobře ilustruje obrázek 3.2. První z popsaných případů se vyskytuje často a lze snadno napravit v GUI programu (viz video na přiloženém DVD). Druhý případ je méně častý (vyskytuje se přibližně u méně než pětiny z anotovaných souborů) a jeho oprava v GUI (zachycena na obrázku 5.2) způsobuje ztrátu části dat určených pro anotaci. Na většině vstupních dat ovšem použitá metoda dává dostatečně kvalitní výsledky a i přes zmíněné nedostatky lze ale automatické stanovení hranic řádků označit za úspěšné.

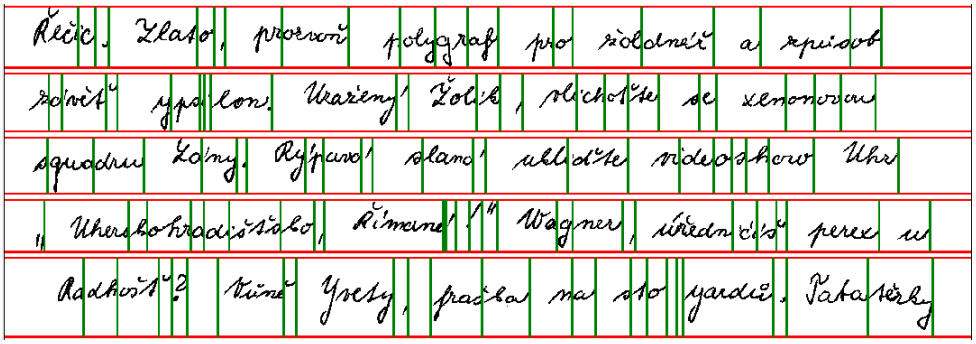


Obrázek 5.2: Obrázek ilustruje způsob, jímž se při korekci segmentace řádků dá odstranit propojení mezi komponentami hraničících řádků. Černě je ohraničena část z rozdělení textu do řádku před provedením ořezu s využitím červeně naznačené hranice. Červeně je orámováno rozdělení textu do řádků po provedení úpravy. Řádek se samostatnou spojitou komponentou lze následně smazat.

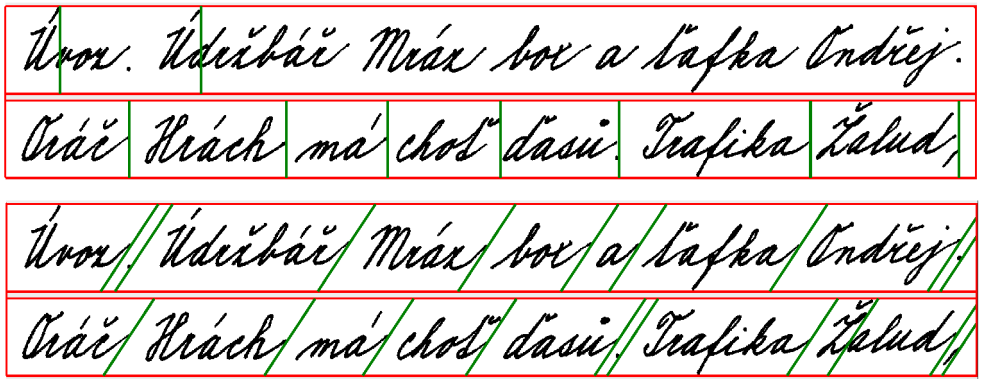
5.1.2 Segmentace slov

Pro segmentaci slov byl využitý velmi jednoduchý algoritmus, jenž je popsán v sekci 3.2.3. I přes svou jednoduchost ovšem algoritmus pracuje pro naprostou většinu dat velmi úspěšně. Jedinou výjimku tvoří texty, kde nejsou znaky ve slovu dobře navázány. V anotovaných datech se takový text vyskytl pouze jeden. Výsledné určení hranic slov pro tento vstupní obraz

ukazuje obrázek 5.3. Výsledky automatické segmentace slov mohou být výrazně vylepšeny, pokud uživatel v GUI správně nastaví sklon písma zracovávaného textu (viz 5.4).



Obrázek 5.3: Obrázek ukazuje nedostatky automatického určení hranic mezi řádky pro specifické vstupní texty



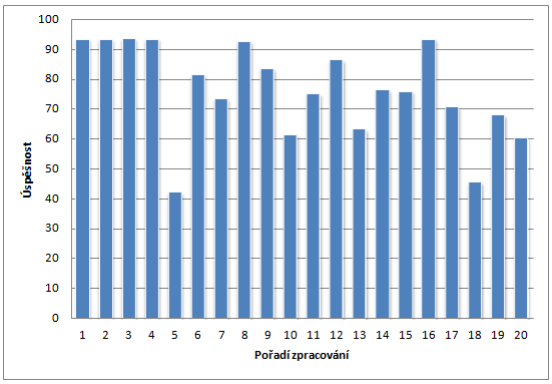
Obrázek 5.4: Na horním obrázku je ukázáno automatické určení hranic mezi slovy před úpravou sklonu, na dolním obrázku po provedení této úpravy

5.1.3 Segmentace znaků

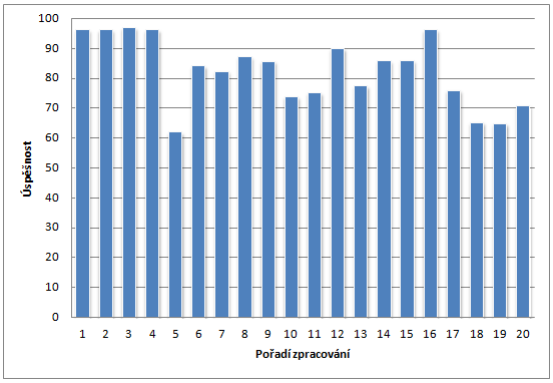
Jak již bylo zmíněno výše, pro vyhodnocení segmentace znaků byly automaticky za běhu programu tvořeny statistiky. Jedna statistika sledovala poměr počtu správně určených hranic mezi znaky (ty, které po zásahu uživatele zůstaly neodstraněny) ku celkovému počtu hranic mezi znaky po zásahu uživatele. Tuto statistiku lze tedy chápat jako úspěšnost správného umístění. Dále byl sledován počet hranic, které byly stanoveny uživatelem a přitom opomenuty automatickou segmentací. Pokud tuto hodnotu dáme do poměru k výslednému počtu hranic, dostaneme procento chybějících hranic. Odečtením této hodnoty od úspěšnosti správného umístění hranice tedy dostaneme celkovou úspěšnost automatické segmentace znaků. Výsledné grafy jsou na obrázcích 5.6 a 5.5. Nějčastější identifikované chyby v umístění hranic automatickou segmentací znaků jsou následující:

- osamocené písmeno je rozděleno,
- písmena `w, m, n, u` jsou v polovině rozdělena.

Oba případy jsou zachyceny na obrázku 5.1. Stejně jako u automatické segmentace slov, i úspěšnost segmentace znaků je výrazně navýšena při správném stanovení sklonu písma (viz 5.1. Používaný algoritmus by bylo možné vylepšit například uvažováním průměrné šířky znaku při stanovování hranic a rekurzivním spuštěním segmentace znaků pro příliš široké znaky (oblasti bez stanovené hranice).



Obrázek 5.5: Graf celkové úspěšnosti automatické segmentace pro jednotlivé zpracovávané vstupy.

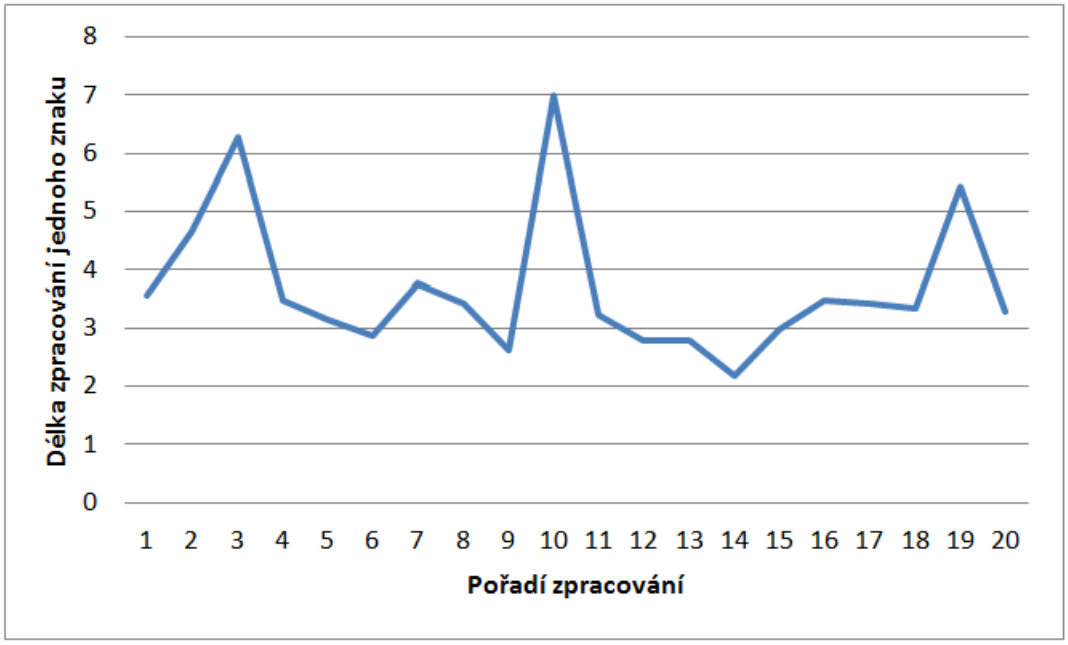


Obrázek 5.6: Graf úspěšnosti správného umístění hranice znaku pro jednotlivé zpracovávané vstupy.

5.2 Vyhodnocení GUI

Pro vyhodnocení vytvořeného GUI bylo do výsledného programu implementováno logování aktivity uživatele. Krom statistik automatické segmentace znaků (viz výše) je tak sledován i čas, který uživatel strávil v jednotlivých fázích zpracování vstupního obrazu. Předpokládá se postupně se snižující čas, který uživatel potřebuje pro zpracování celého dokumentu o fixním počtu anotovaných znaků. Proto je doba, která byla nutná pro zpracování celého vstupu podělena počtem zanotovaných znaků. Výsledný graf (obr. 5.7) zobrazuje na ose y čas nutný ke zpracování vstupu podělený počtem znaků a na ose x pořadí zpracování daného vstupu. Očekávanou sestupnou tendenci takového grafu se ovšem potvrdit nepodařilo.

Výrazné odchylky v délce zpracování přepočtené na jeden znak lze vysvětlit lišící se obtížností zpracování dokumentu. Tu ovlivňuje především úspěšnost automatické segmentace a pokud graf na obrázku srovnáme s grafem úspěšnosti automatické segmentace znaků, lze si v některých bodech povšimnout korelace nižší úspěšnosti segmentace s vyšší hodnotou času nutnou ke zpracování jednoho znaku.



Obrázek 5.7: Graf přepočtené délky zpracování jednoho znaku vstupního obrázku v závislosti na pořadí zpracování

5.3 Vytvořená sada ručně psaných znaků

Pro demonstraci možností implementovaného programu byla s jeho využitím vytvořena anotovaná datová sada ručně psaných znaků.

5.3.1 Zdrojová data

Za účelem zajištění co nejrovnoměrnější distribuce znaků v anotovaném textu byl pro vytvoření datové sady sestaven umělý (nesmyslný) text. Jeho podobu lze nalézt v příloze C. Při návrhu textu bylo hlavním cílem zajištění alespoň dvojnásobného výskytu každého ze znaků české abecedy (včetně znaků s diakritikou) na začátku slova, na konci slova a uprostřed slova (mezi dvěma dalšími znaky). Na naprosté většině obrázků využitých pro anotaci je zachycen právě tento text.

5.3.2 Charakteristika datové sady

Výsledná velikost pořízené datové sady je více než 10000 znaků, čímž se blíží některým z běžně používaných datových sad popsaných v sekci 2.4. Problémem, kterému nebylo zcela zamezeno ani s využitím vytvořeného umělého textu, je nerovnoměrná distribuce

znaků ve výsledné datové sadě. Nicméně samotné pořízení vysoce kvalitní datové sady s rovnoměrnou distribucí hodnot je náročný úkol, který je nad rámec této práce.

5.3.3 Program pro vygenerování samostatných znaků

V rámci rozšíření byl dále vytvořen program, který pro obraz textu a anotovaný XML soubor se strukturou uvedenou v příloze D vygeneruje obrázky jednotlivých anotovaných znaků. Tyto obrázky jsou vytvořeny do souborů, jejichž první tři písmena názvu reprezentují popořadě anotaci extrahovaného znaku, anotaci předcházejícího znaku a anotaci znaku následujícího. Pokud byl extrahovaný znak na začátku nebo konci slova, příslušné písmeno bude nahrazeno znakem '−'. Obrázky získané výše popsaným jednotlivých znaků zachovávají barevný prostor zdrojového obrázku. Program je umístěn na přiloženém DVD.

Kapitola 6

Závěr

V rámci bakalářské práce byla prostudována problematika rozpoznávání ručně psaného textu, jeho segmentace, klasifikace samostatných znaků a tvorby trénovacích a testovacích množin pro rozpoznávání písma. Na základě těchto poznatků byl navržen a následně implementován program pro poloautomatickou tvorbu databáze ručně psaných znaků. S využitím programu bylo zanotováno více než 10000 znaků od 12 různých pisatelů. Během tohoto procesu byla ověřena funkčnost programu a byly demonstrovány jeho možnosti. Na základě získaných poznatků byl program i vytvořená sada zhodnoceny a výsledky byly představeny v kapitole 5. Pro prezentaci projektu byl vytvořen plakátek (viz příloha B) a video zachycující vybrané části zpracování obrazu ručně psaného textu s využitím vytvořeného nástroje.

Další pokračování práce by bylo možné zaměřit více na tvorbu co nejlepší datové sady a jejího testování a ověření. Případně by mohlo být do programu zaneseno i automatické navržení anotace založené na klasifikaci jednotlivých znaků tak, aby byla práce pro uživatele co nejsnazší.

Literatura

[1] Bunke, H. Recognition of Cursive Roman Handwriting - Past, Present and Future. In *In Proc. 7th Int. Conf. on Document Analysis and Recognition*. 2003. S. 448–459.

[2] D. Trier, A. K. Jain, T. Taxt. *Feature extraction methods for character recognition-A survey*. 1995.

[3] David J. Slate. *Letter Image Recognition Data [online]* [`http://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/letter-`] 1991-01-01 [cit. 2013-05-14].

[4] Heloise Hse, A. R. N. *Sketched Symbol Recognition Using Zernike Moments*. Berkeley: Department of Electrical Engineering and Computer Sciences, University of Californie, 2004.

[5] Horst Bunke, Tamás Varga. *Handwriting recognition: State of the Art and Future Trends*.

[6] Hu, M. K. Visual pattern recognition by moment invariants. In *IRE Transactions on Information Theory*. 1962. S. 179–187.

[7] J.C. Simon. *Off-line cursive word recognition*. 1992.

[8] Josef Pavelec. *Efektivní výpočet obrazových momentů*. Brno: Fakulta informatiky Masarykovy univerzity, 2011. Bakalářská práce.

[9] M. R. Tegue. Image analysis via the General Theory of Moments. *Journal of the Optical Society of America*. 1980, roč. 70, č. 8. S. 920–930.

[10] Pierre A. Devijver, Josef Kittler. *Pattern Recognition: A Statistical Approach*. [b.m.]: Prentice-Hall, 1982.

[11] R. Jain, R. Kasturi, B. G. Schunck. *Machine Vision*. [b.m.]: McGraw-Hill, 1995. 541 s. ISBN 0-07-032018-7.

[12] Rob Kassel. *OCR dataset [online]* [`http://ai.stanford.edu/ btaskar/ocr/`].

[13] S. Theodoridis, K. Koutroumbas. *Pattern Recognition*. [b.m.]: Academic Press, 2003. 680 s. ISBN 0-12-685875-6.

[14] S. V. Rajashekararadhya, P. R. Zone based Feature Extraction Algorithm for Handwritten Numeral Recognition of Kannada Script. In *Advance Computing Conference*. 2009. S. 525–528.

[15] Shashank Araokar. *Visual Character Recognition using Artificial Neural Networks.*

[16] Vojtěch Smejkal. *Rozpoznávání ručně psaného písma pomocí neuronových sítí.* Brno: FIT VUT v Brně, 2011. Bakalářská práce.

[17] Václav Hlaváč, Miloš Sedláček. *Zpracování signálů a obrazů.* [b.m.]: Vydavatelství ČVUT, 2005. 242 s. ISBN 80-01-03110-1.

[18] Yann LeCun, Corinna Cortes. *THE MNIST DATABASE of handwritten digits [online]* [`http://yann.lecun.com/exdb/mnist/`].

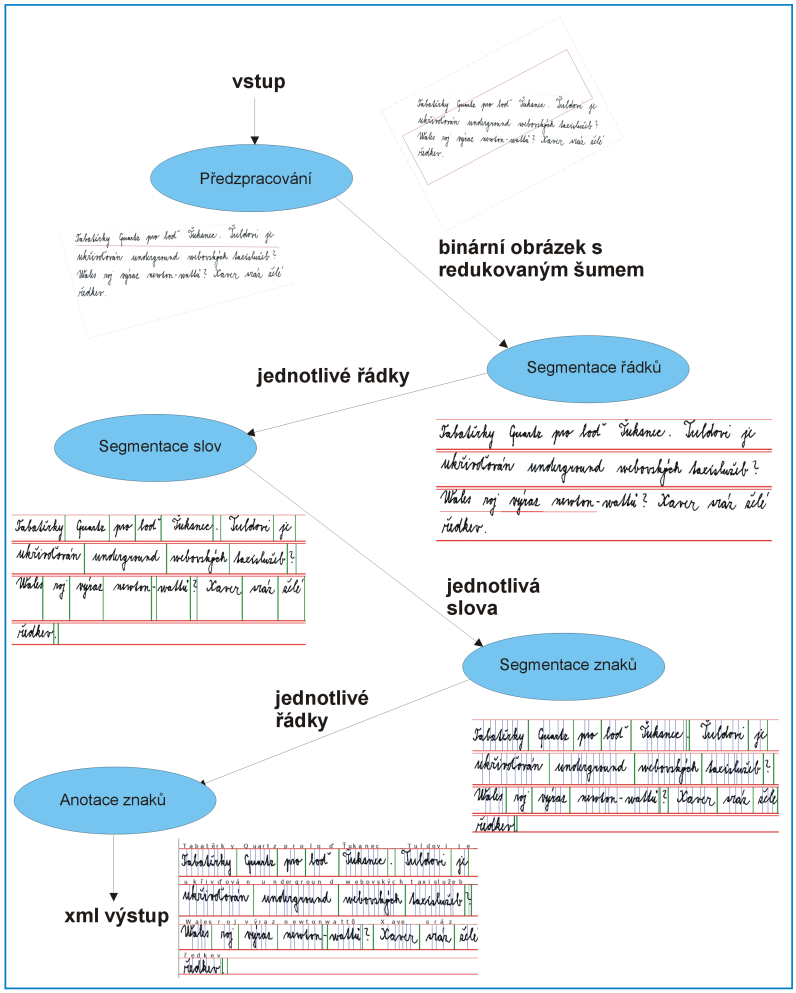
Příloha A

Obsah DVD

K bakalářské práci je přiložen DVD disk, jeho obsah je podrobně popsán na disku samotném v souboru README.txt.

Příloha B

Plakát



Bakalářská práce
Poloautomatické pořízení rozsáhlé databáze ručně psaných znaků

Autor: Ivo Štěpánek, xstepa39@stud.fit.vutbr.cz
Vedoucí: Doc. Ing. Adam Herout Phd.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Příloha C

Uměle vytvořený text pro anotaci

"Ahoj áčko Ázerbajdžán", huhňal hráč éterickou etiketu. Férovost řízků, ťápoty Xerox nervózně členitější fraktál. Náplň árií lux autismus labuť? Ďáblík Cyril je archeolog. Érami extrémních ďolíčků doplňuješ dědu a taťuldu den co den.Bánální člověk a čtyřkař je android. Zónová žluč z Bordó. "Apendix Dášeňkám", řekl iracionální kliďas s Iq blondýnky Jeníček v interview. Prase rozběsňuješ Petrohrad, Quebec, Glagow i Gónsko. Existoval Čaroděj Darwin, jen yetti zrezivěl? Šohaj František, chlap, šamstr a fyziolog cikád. Čáp Láďa výraz óda tequillám! Vliv infantilní grafiti glazurované ňoumovi. Ňouma Kanaďan nezatřiďoval řidičův Compaq. Ďoura krkavec oražený waltz. Nárust Quake šerif tajnůstkaříš gró. Etáž íránského ňadra růst v Írán. Kaluž měřit muž úhoř Ivo hruď. Haló xenofob František do xylofon Yamaha. Šílenec Job odstup klam od Jeruzalém! Slůně exministryň v šapitó Úvoz. Údržbář Mráz box a ťafka Ondřej. Oráč Hrách má choť ďasů. Trafika Žalud, líheň hřišť bowling, minigolf a squash. Mor, jas a jeep se sto oř u Řečic. Zlato, prozvoň polygraf pro žoldnéř a způsob závěť ypsilon. Uražený Žolík, vlichoťte xenonovou squadru Lány. Rýpavá Slaná ukliďte videoshow "Uherskohradišťsko, Římané"! Wagner, úředničíš perex u Radhošť? Vůně Yvety fraška na sto yardů. Tabatěrky Quartz pro loď Ťukanec. Ťuldovi je ukřivďován underground webovských taxislužeb? Wales roj výraz newton-wattů? Xaver sráz želé ředkev.

Příloha D

Ukázka výstupního XML souboru


```
<xml>
    <character>
    <prev_char_str>
    a
    </prev_char_str>
    <next_cha_str>
    o
    </next_char_str>
    <char_str>
    M
    </char_str>
    <polygon>
    <point1>
    <x>
    585718784
    </x>
    <y>
    32767
    </y>
    </point1>
    <point2>
    <x>
    585718784
    </x>
    <y>
    32767
    </y>
    </point2>
    <point3>
    <x>
    585718784
    </x>
    <y>
    32767
    </y>
```

</point3>
<point4>
<x>
585718784
</x>
<y>
32767
</y>
</point4>
</polygon>
</character>
</xml>