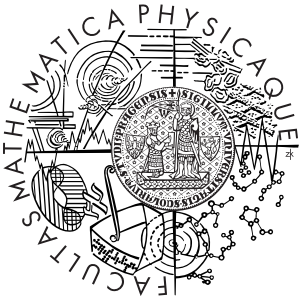
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Hubert Kindermann

Vytěžování textu ze strojově psaných dokumentů

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Jan Blažek
Studijní program: Informatika, Obecná informatika

2011

Chtěl bych poděkovat svému vedoucímu práce Mgr. Janu Blažkovi za cenné připomínky a všem, kteří při mně stáli během mého studia.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 22. května 2011 Hubert Kindermann

Obsah

Název práce: Vytěžování textu ze strojově psaných dokumentů
Autor: Hubert Kindermann
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: Mgr. Jan Blažek
e-mail vedoucího: blazek@utia.cas.cz

Abstrakt: V předložené práci řešíme problém extrakce a rozpoznání znaků z tištěných dokumentů digitalizovaných skenerem nebo fotoaparátem. Uvádíme způsob normalizace osvětlení dokumentů rezistentní vůči šumu. Pokračujeme extrakcí jednotlivých znaků z dokumentu a následně jejich rozpoznáním pomocí systému vícevrstvých neurálních sítí s dopředným šířením. Okrajově se zabýváme zpracováním výsledné množiny rozpoznaných symbolů, které je nezbytné pro další práci s vytěženým textem. Posledním krokem je korekce výstupu založená na okolích jednotlivých znaků. Podařilo se nám implementovat automatický systém obsahující všechny zmíněné komponenty.

Klíčová slova: Optické rozpoznávání znaků, Extrakce textu, Normalizace osvětlení

Title: Character recognition of machine-written documents
Author: Hubert Kindermann
Department: Název katedry či ústavu v angličtině
Supervisor: Mgr. Jan Blažek
Supervisor's e-mail address: blazek@utia.cas.cz

Abstract: In the present thesis we solve the problem of symbol extraction and recognition from printed documents digitized by the scanner or camera. We introduce a noise resistant algorithm of document lighting normalization. We continue with the extraction of individual characters from the document and their recognition with a system of feedforward multilayer neural networks. We also focus on processing of the resulting set of recognized characters, which is necessary for further use of the extracted text. The last step is correction of the output based on surrounding letters of each character. We have successfully implemented an automatic system containing all the above components.

Keywords: Optical Character Recognition, Text Extraction, Lightning Normalization

Kapitola 1

Úvod

V dnešní době je snaha co největší množství informací digitalizovat. Tento trend je hlavní motivací pro mou bakalářskou práci. Do kategorie digitalizovaných informací spadají například knihy a zejména tištěné dokumenty. Zdroje informací uchovávané na papíře jsou nepraktické z důvodu objemnosti, nemožnosti kvalitního vyhledávání nebo snadného odkazování. Tyto dokumenty dnes nejčastěji digitalizujeme pomocí skenerů nebo fotoaparátů do formy bitmap. Proces digitalizace s sebou nese zcela zásadní problémy. Je to například nemožnost snadného postprocessingu digitalizovaného textu (formátování, kopírování, ...) nebo neúměrně velká prostorová náročnost zdaleka neodpovídající množství informací, které se doopravdy snažíme uchovat. Přínosy současného způsobu zpracování klasických archivů jsou zejména v zpřístupnění dokumentů prostřednictvím internetu a jejich snadné zálohování. Přirozeným požadavkem na digitalizovaný archiv je především možnost vyhledávání dat a jejich zpracování jako textu. Dalším přínosem digitalizovaného archivu v textové formě je řádové zmenšení jeho prostorové náročnosti (namísto megabytů dat stačí uchovávat kilobyty). Jako řešení zmíněných problémů vznikají systémy pro detekci a rozpoznávání textu v bitmapách, které převádějí obrazová data do textové formy.

Vytěžování textu z naskenovaných, či jinak zdigitalizovaných dokumentů přináší následující problémy:

- odšumění a zaostření dat,
- vyrovnání nerovností v osvětlení plochy dokumentů,
- zakřivení řádků nebo jiná celková deformace textu,
- splynutí znaků v důsledku rasterizace.

1.1 Specifikace zadání

V této práci popíšeme systém pro extrakci a rozpoznávání textu z digitálně pořízených dokumentů psaných strojem. To mohou být například fotografie knih nebo data pořízená ze skeneru. Pořízené dokumenty převádíme do textové podoby zachovávající pozice znaků, čitelnost a rozšiřující o možnost vyhledávání. Pracovat budeme pouze s abecedou obsahující znaky a-zA-Z0-9, speciální a diakritiku (viz příloha na CD). Ve vytěžovaných dokumentech nepředpokládáme výskyt obrázků.

Kapitola 2

Analýza problému

2.1 Základní pojmy a terminologie

V celé práci budeme předpokládat, že všechny bitmapy mají kladné celočíselné a konečné rozměry.

Definice 1. *Pixelem budeme rozumět uspořádanou trojici*

$$\rho = (\rho_x, \rho_y, \rho_{sat}) \in (\mathbb{Z}_0^+)^2 \times [0,1],$$

kde ρ_x, resp. ρ_y, resp. ρ_{sat} značí x-ovou souřadnici, resp. y-ovou souřadnici, resp. saturaci pixelu ρ a $[0,1]$ značí uzavřený interval.

Saturace pixelu je aritmetický průměr jeho barevných RGB komponent, přičemž pixel ρ považujeme za černý, pokud $\rho_{sat} = 0$ a za bílý, je-li $\rho_{sat} = 1$.

Definice 2. *Bitmapa B je matice*

$$\begin{pmatrix} \rho_{sat}^{0,0} & \rho_{sat}^{1,0} & \cdots & \rho_{sat}^{B_w-1,0} \\ \rho_{sat}^{0,1} & \rho_{sat}^{1,1} & \cdots & \rho_{sat}^{B_w-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{sat}^{0,B_h-1} & \rho_{sat}^{1,B_h-1} & \cdots & \rho_{sat}^{B_w-1,B_h-1} \end{pmatrix}$$

bitmapy B velikosti $B_w \times B_h$, kde $B_w \in \mathbb{Z}^+$, resp. $B_h \in \mathbb{Z}^+$ značí šířku, resp. výšku a $\rho^{i,j}$ je pixel bitmapy B na pozici (i,j), pro $i \in \{0,\ldots,B_w\}$ a $j \in \{0,\ldots,B_h\}$. Saturaci pixelu v bitmapě B na pozici (x,y) budeme značit $B[x,y]$.

Definice 3. *Množina Π_B je reprezentující množinou bitmapy B, jestliže platí*

$$\Pi_B = \{(x, y, B[x,y]) \,|\, x \in \{0,\ldots,B_w - 1\}, y \in \{0,\ldots,B_h - 1\}\}.$$

Pro zjednodušení terminologie budeme-li v práci hovořit o bitmapě Π_B, budeme mít na mysli množinu Π_B reprezentující bitmapu B. Řekneme-li, že pro dvě bitmapy Π_A a Π_B platí

$$|\Pi_A| = |\Pi_B|,$$

budeme tím myslet, že

$$A_w = B_w \ \wedge \ A_h = B_h.$$

Definice 4. *Mějme bitmapu Π_B a pixel $\rho \in \Pi_B$. Pak budeme psát*

$$\rho = \Pi_B[x, y],$$

jestliže

$$\rho_x = x \ \wedge \ \rho_y = y \ \wedge \ \rho_{sat} = B[x, y].$$

O pixelu ρ pak řekneme, že leží na pozici (x, y) v bitmapě Π_B.

Definice 5. *Vzdálenost* $\operatorname{dist}(\rho, \omega)$ *dvou pixelů ρ a ω v bitmapě budeme brát jako jejich euklidovskou vzdálenost v rovině. Tedy*

$$\operatorname{dist}(\rho, \omega) = \sqrt{(\rho_x - \omega_x)^2 + (\rho_y - \omega_y)^2}.$$

Definice 6. *Řekneme, že X_{Π_B} je diskrétní náhodná veličina reprezentující bitmapu Π_B, jestliže*

$$\forall \rho \in \Pi_B \colon P(X_{\Pi_B} = \rho_{sat}) = \frac{|\{\pi \in \Pi_B | \pi_{sat} = \rho_{sat}\}|}{|\Pi_B|}.$$

Definice 7. *Existuje-li střední hodnota $\mathbb{E}\, X_{\Pi_B}$, resp. rozptyl $\operatorname{var} X_{\Pi_B}$, resp. n-tý moment $\mathbb{E}\, X^n_{\Pi_B}$ diskrétní náhodné veličiny X_{Π_B} reprezentující bitmapu Π_B, pak označíme střední hodnotu, resp.rozptyl, resp. n-tý moment bitmapy Π_B jako $\mathbb{E}\, \Pi_B = \mathbb{E}\, X_{\Pi_B}$, resp. $\operatorname{var} \Pi_B = \operatorname{var} X_{\Pi_B}$, resp. $\mathbb{E}\, \Pi^n_B = \mathbb{E}\, X^n_{\Pi_B}$.*

Definice 8. *Definujme následně na množině Π_B reprezentující bitmapu B uspořádání $\leq$:*

$$\forall \rho, \omega \in \Pi_B \colon \rho \leq \omega \Leftrightarrow \rho_{sat} \leq \omega_{sat}.$$

Bude-li dále řečeno, že pixel ρ je **tmavší** než pixel ω, budeme tím mít na mysli, že $\rho < \omega$.

Definice 9. *Mějme množinu Π_B reprezentující bitmapu B. Potom označíme zobrazení $\tau_B \colon \Pi_B \to \{0, 1\}$ definované předpisem*

$$\tau_B(\rho) = \begin{cases} 0 & \text{když } \rho \in \Pi_B \text{ tvoří pozadí bitmapy } B, \\ 1 & \text{když } \rho \in \Pi_B \text{ tvoří text v bitmapě } B, \end{cases}$$

jako indikátor textu v bitmapě B.

2.2 Idea řešení

Hlavním problémem, před který jsme postaveni ještě před samotným rozpoznáváním písmen, je jejich extrakce z bitmapy. Pokud se jedná o fotografie, je tato extrakce netriviální a bude jí proto věnována velká část této práce.

Pro úspěšné vytěžení dokumentu budeme řešit následující problémy:

- extrakce znaků (problém zejména u fotografií),
  - odšumění
  - normalizace osvětlení
  - vlastní extrakce
- rozpoznání jednotlivých znaků,
- kompozice znaků do písmen (především písmena s diakritikou),
  - jak poznat, které znaky jsou diakritickými znaménky a ke kterým písmenům patří
  - vzhledem k malé velikosti diakritických znamének po rasterizaci originálních dokumentů s nedostatečně vysokým rozlišením jsou tyto extrahované symboly velmi podobné
- rozložení do slov a řádků.

Při extrakci znaků popsané v kapitole 3.1 je nutné dokument nejdříve vyčistit pro snadné vyhledání jednotlivých symbolů, které pak můžeme dle kapitoly 3.2 rozpoznat. Kompozice znaků popsaná v sekci 3.3.1 je nezbytná, jelikož například diakritická znaménka rozpoznáváme jako samostatné znaky, které nejsou spárované s příslušejícími písmeny.

Kapitola 3

Metodika řešení

3.1 Extrakce znaků

Cílem této sekce je nalezení minimálních obdélníkových oblastí obsahujících jednotlivé znaky, které se vyskytují ve zkoumaných dokumentech.

3.1.1 Extrakce z ideálního dokumentu

Definice 10. *Jako ideální dokument označme bitmapu Π_B, právě když splňuje následující podmínky*

$$\{\rho_{sat} | \rho \in \Pi_B\} \subseteq \{0, 1\}$$
$$\forall \rho \in \Pi_B \colon \rho_{sat} = 0 \Leftrightarrow \tau_B(\rho) = 1$$
$$\forall \rho \in \Pi_B \colon \rho_{sat} = 1 \Leftrightarrow \tau_B(\rho) = 0.$$

Jinými slovy, ideálním dokumentem je bitmapa obsahující pouze černé a bílé pixely. Černé pixely tvoří v bitmapě text a bílé pixely pozadí.

Definice 11. *Nechť Π_B je bitmapa. Oblast $\Delta \subseteq \Pi_B$ označíme jako 4-souvislou, právě když pro každé dva pixely $\rho_1, \rho_2 \in \Delta$ existuje cesta mezi ρ_1 a ρ_2 složená pouze z horizontálních a vertikálních kroků po vnitřních pixelech oblasti Δ.*

Náš extrakční algoritmus je navržený pro extrakci znaků právě z ideálních dokumentů. Funguje na bázi procházení všech pixelů bitmapy a vrací seznam pozic a rozměrů minimálních obdélníků obsahujících celé jednotlivé znaky. V okamžiku, kdy extrakční algoritmus narazí na černý pixel, spustí se procedura, která nalezne všechny černé pixely tvořící 4-souvislou plochu a následně vrátí nejmenší obdélník obsahující právě nalezenou oblast. Ukázka výstupu je na obrázku 3.1.

(a) Původní dokument (b) Segmentovaný dokument

Obrázek 3.1: Ukázka práce segmentovacího algoritmu.

Pokud si důkladně prohlédneme tištěné dokumenty, zjistíme, že určité dvojice písmen se v některých částech mohou (ve smyslu euklidovské vzdálenosti) velmi přibližovat, což může po převedení do digitální podoby způsobit spojení dvou či více písmen dohromady (ve smyslu 4-souvislosti), jak je vidět na obrázku 3.2. Pokud ke spojení dojde, náš extrakční algoritmus takovou dvojici extrahuje pouze jako jeden znak. Tento jev je pro náš rozpoznávací algoritmus nežádoucí, protože rozpoznávací systém je primárně navržený na rozpoznávání jednotlivých symbolů.

Vyřešit problém omezený pouze na bitmapu obsahující dva znaky spojené jedním nebo více černými pixely, je obtížné. Problém proto nebudeme řešit v extrakční fázi algoritmu, ale pokusíme se jej řešit až ve fázi rozpoznávání znaků.

Obrázek 3.2: Dvojice "ty", "vý", "kl".

3.1.2 Extrakce z reálného dokumentu

Reálně pořízené dokumenty nesplňují definici ideálního dokumentu. Procedura vyhledávající pixely, jež náležejí jednotlivým znakům, tedy nebude fungovat. Pokud bychom předpokládali, že saturace pixelů reprezentujících písmena je vždy menší než saturace pixelů reprezentujících pozadí, lze jednoduše extrakční algoritmus pro ideální dokumenty upravit. Mějme tedy práh $\eta \in [0, 1]$. Předpokládejme platnost následujícího tvrzení pro každý reálný dokument Π_B:

$$\forall \delta \in \Pi_B \colon \tau_B(\delta) = 1 \Leftrightarrow \delta_{sat} < \eta,$$

z něhož snadno plyne i komplementární tvrzení

$$\forall \delta \in \Pi_B \colon \tau_B(\delta) = 0 \Leftrightarrow \delta_{sat} \geq \eta.$$

Pak stačí algoritmus upravit tak, že pixely se saturací menší než η by se rozpoznávaly jako černé a naopak.

Hraniční saturaci η bohužel nemůžeme vždy ani pevně určit kvůli různému osvětlení dokumentů. Saturační hranice se může pro různé dokumenty lišit, jak vidíme na obrázku 3.3(a) a 3.3(b). Může dojít i ke stavu, kdy v jedné části dokumentu bude písmo světlejší než pozadí v jiné části dokumentu, což je vidět na obrázku 3.3(c). Problému se pokusíme předejít tím, že ještě před vlastní extrakcí znaků se pokusíme dokument převést do podoby splňující předpoklady ideálního dokumentu, což znamená černý text na bílém pozadí.

(a) Tmavé pozadí (b) Světlý text (c) Proměnná saturace

Obrázek 3.3: Dokumenty s různou hraniční saturací pro extrakci znaků.

3.1.3 Determinace šumu v reálných dokumentech

Jak vidíme v následující sekci 3.1.4 o čistícím algoritmu, budeme pro každý reálný dokument potřebovat určit, zda je nebo není zašuměný, a to z důvodu některých odlišností v přístupu k čištění dokumentu. Tento závěr nelze učinit na základě odhadu rozptylu šumu obsaženého v dokumentu, neboť běžné metody pro určení tohoto odhadu divergují od správného řešení, pokud se rozptyl šumu blíží nule. Jako příklad uveďme metodu popsanou v práci [3].

Pro zkoumaný dokument předpokládáme uniformní rozdělení aditivního šumu nezávislé na obsahu dokumentu.

Definice 12. *Entropii H diskrétní náhodné veličiny X s možnými hodnotami $\{x_1, x_2, ..., x_n\}$, pro které platí*

$$P(X = x_i) \in (0, 1], \textit{ pro } i \in \{1, \ldots, n\},$$

definujeme podle [2] jako

$$H(X) = -\sum_{i=1}^{n} P(X = x_i) \log P(X = x_i).$$

Entropie ve své podstatě určuje míru nepředvídatelnosti hodnoty náhodné veličiny X.

Důsledek 1. *Nechť X je diskrétní náhodná veličina s možnými hodnotami $\{x_1, x_2, \ldots, x_n\}$ pro které platí*

$$P(X = x_i) \in (0, 1], \text{ pro } i \in \{1, \ldots, n\}.$$

Pak platí, že $H(X) \geq 0$.

Definice 13. *Entropií H_{Π_B} bitmapy Π_B nazvěme entropii H diskrétní náhodné veličiny X_{Π_B} reprezentující bitmapu Π_B.*

Náš algoritmus rozhodující o tom, zda je dokument zašuměný nebo ne, bude spočívat v nalezení nejstabilnější oblasti pevně dané velikosti v dokumentu a následném rozhodnutí, zda je tato oblast zašuměná. Stabilitou oblasti dokumentu zde myslíme míru prediktibility saturací jednotlivých pixelů v ní obsažených. To, že budeme hledat nejstabilnější oblast dokumentu, je přirozený důsledek požadavku nalezení oblasti neobsahující žádný text, jelikož ten působí jako element zvyšující entropii celé oblasti. Formálně tedy nejdříve nalezneme souvislou čtvercovou oblast $\Delta \subseteq \Pi_B$ s konstantní velikostí C takovou, že

$$\Delta \in \operatorname*{argmin}_{\substack{\Omega \subseteq \Pi_B \\ |\Omega| = C}} H_\Omega. \tag{3.1}$$

Přirozeným předpokladem, který platí podle důsledků 1 a 2, je to, že oblast pixelů, kde každý pixel má stejnou saturaci, bude mít maximální možnou stabilitu, a tudíž minimální entropii.

Důsledek 2. *Nechť množina Ω reprezentující bitmapu splňuje podmínku*

$$\forall \rho, \omega \in \Omega \colon \rho_{sat} = \omega_{sat}. \tag{3.2}$$

Potom platí, že

$$H_\Omega = 0.$$

Máme-li bitmapu Π_B, tak dle důsledků (1) a (2) přímo dostáváme následující implikaci:

$$\exists \Delta \subseteq \Pi_B \colon |\Delta| = C \wedge \ \forall \rho, \omega \in \Delta \colon \rho_{sat} = \omega_{sat}$$

$$\Rightarrow$$

$$\exists \Delta \subseteq \Pi_B \colon |\Delta| = C \wedge \Delta \in \operatorname*{argmin}_{\substack{\Omega \subseteq \Pi_B \\ |\Omega| = C}} H_\Omega.$$

Jinými slovy, existuje-li v dané bitmapě oblast velikosti C obsahující všechny pixely se stejnou saturací, tak ji v (3.1) najdeme.

Rozhodnutí o zašuměnosti dokumentu určíme na základě rozptylu náhodné veličiny

$$Y = X_\Delta - \mathbb{E}\, X_\Delta,$$

kde X_Δ je diskrétní náhodná veličina reprezentující bitmapu $\Delta \subseteq \Pi_B$ splňující podmínku (3.1). Jelikož je rozptyl invariantní k translaci, tak je vidět, že nám stačí zkoumat

$$var\, Y = var\, (X_\Delta - \mathbb{E}\, X_\Delta) = var\, X_\Delta.$$

Po nalezení nejstabilnější oblasti Δ v bitmapě Π_B tedy učiníme rozhodnutí o zašuměnosti dokumentu na základě rozptylu právě oblasti Δ.

Abychom věděli, vůči jakým hodnotám máme $var\, X_\Delta$ porovnávat, namodelujme náhodnou veličinu X_Δ pomocí náhodné veličiny $\widetilde{X}_\Delta$ takové, že

$$\widetilde{X}_\Delta = X_{\Delta_0} + Q_\Delta,$$

kde X_{Δ_0} je náhodná veličina reprezentující nezašuměnou bitmapu Δ a Q_Δ je náhodná veličina reprezentující šum v bitmapě Δ. Vzhledem k vysoké stabilitě oblasti Δ podle (3.1) odhadneme

$$var\, X_{\Delta_0} \approx 0$$

a dostaneme

$$var\, \widetilde{X}_\Delta \approx var\, Q_\Delta.$$

Nyní nám stačí pouze zvolit vhodnou hraniční hodnotu σ_Q^2 rozptylu šumu Q_Δ, pro kterou budeme dokument ještě považovat za nezašuměný. Jelikož jsme našli odhad

$$var\, X_\Delta \approx var\, Q_\Delta,$$

tak náš algoritmus bude rozhodovat následovně:

- jestliže $var\, X_\Delta \leq \sigma_Q^2$, pak dokument Π není zašuměný,

- jestliže $var\, X_\Delta > \sigma_Q^2$, pak dokument Π je zašuměný.

Nejlepší chování algoritmu bylo experimentálně ověřeno pro hodnotu $\sigma_Q^2 = 2.5 \times 10^{-3}$ a velikost oblasti $C = 30 \times 30$.

3.1.4 Čistící algoritmus

Nyní představíme algoritmus pro vyčištění reálného dokumentu. Vyčištěním zde myslíme převod na dokument, který se co nejvíce svými vlastnostmi blíží k dokumentu ideálnímu. Vzhledem k problému s osvětlením reálných dokumentů popsanému u obrázku 3.3(c) nemůžeme použít žádný jednoduchý konvoluční algoritmus, který by dokument prošel pixel po pixelu a znormalizoval ho. Proto volíme postup, kdy dokument převedeme do podoby ideálního dokumentu a pro extrakci symbolů použijeme postup popsaný v sekci 3.1.1.

Definice 14. *Nechť Π_B je bitmapa a*

$$p_B = \frac{|\{\rho \in \Pi_B | \tau_B(\rho) = 1\}|}{|\Pi_B|}$$

je pravděpodobnost, že náhodně zvolený pixel z bitmapy B bude reprezentovat text. Pak predikční bitmapa $\Pi_{\widetilde{B}}$ pro bitmapu Π_B je určena následujícími vlastnostmi:

(1) $|\Pi_{\widetilde{B}}| = |\Pi_B|$,

(2) $\{\widetilde{\rho}_{sat} | \widetilde{\rho} \in \Pi_{\widetilde{B}}\} \subseteq \{0, 1\}$,

(3) pro náhodné $\rho \in \Pi_B$: $P(\tau_B(\rho) = 1 \,|\, \widetilde{\rho}_{sat} = 1) \geq p_B$, *kde* $\widetilde{\rho}_{sat} = \widetilde{B}[\rho_x, \rho_y]$,

(4) pro náhodné $\rho \in \Pi_B$: $P(\tau_B(\rho) = 0 \,|\, \widetilde{\rho}_{sat} = 0) \geq 1 - p_B$,
kde $\widetilde{\rho}_{sat} = \widetilde{B}[\rho_x, \rho_y]$.

Definice 15. *Definujme množinu Π_B^+, resp. Π_B^- reprezentující text, resp. pozadí v bitmapě Π_B jako*

$$\Pi_B^+ = \{\rho \,|\, \rho \in \Pi_B \wedge \tau_B(\rho) = 1\},$$
$$\Pi_B^- = \{\rho \,|\, \rho \in \Pi_B \wedge \tau_B(\rho) = 0\}.$$

Následující postup bude spočívat v přípravě predikční bitmapy $\Pi_{\widetilde{B}}$, která bude vypovídat, kde můžeme na odpovídajících pozicích v bitmapě Π_B s vysokou pravděpodobností očekávat text a kde pozadí. S její pomocí pak nalezneme

množiny $\widetilde{\Pi}_B^+ \subseteq \Pi_B^+$ a $\widetilde{\Pi}_B^- \subseteq \Pi_B^-$. Následně tyto dvě množiny využijeme při vlastním čištění našeho dokumentu. To bude probíhat na principu odhadu saturace pixelu na dané pozici, pokud by reprezentoval pozadí, resp. text, na základě množin $\widetilde{\Pi}_B^+$ a $\widetilde{\Pi}_B^-$. Nakonec následuje určení, kterému odhadu se více přibližuje skutečná hodnota saturace daného pixelu.

Definice 16. *Označme bitmapu B, resp. Π_B, která byla Gaussovsky rozmazána konvoluční maticí o velikosti 5×5 jako B^*, resp. Π_B^*.*

Poznámka. *Pro Gaussovské rozmazání používáme normalizovanou konvoluční matici*

$$\frac{1}{273}\begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix}.$$

Postup konstrukce predikční bitmapy je založený na podobné myšlence jako u determinace šumu v reálných dokumentech. Vypočteme rozptyl Gaussovsky rozmazané oblasti $\Omega_B \subseteq \Pi_B$ právě procházeného pixelu v bitmapě Π_B a porovnáme ho s určitou prahovou hodnotou. Následně určíme saturaci odpovídajícího pixelu v predikační bitmapě $\Pi_{\widetilde{B}}$. Oblast před určením jejího rozptylu Gaussovsky rozmažeme z důvodu redukce míry ovlivnění rozptylu šumem v dané oblasti. Zmíněný postup vychází z přirozené představy podobnosti zkoumané oblasti Ω_B^* s oblastí tvořenou pixely se saturací rovnou střední hodnotě diskrétní náhodné veličiny X_{Ω^*} reprezentující oblast Ω_B^*. Tuto podobnost budeme určovat pomocí euklidovské vzdálenosti vektorů tvořených saturacemi pixelů obou zmíněných oblastí. Bude-li tato vdálenost rovna 0, budeme oblasti považovat za stejné. Označíme-li danou vzdálenost jako $\delta(\Omega_B^*)$, dostáváme následující vztah

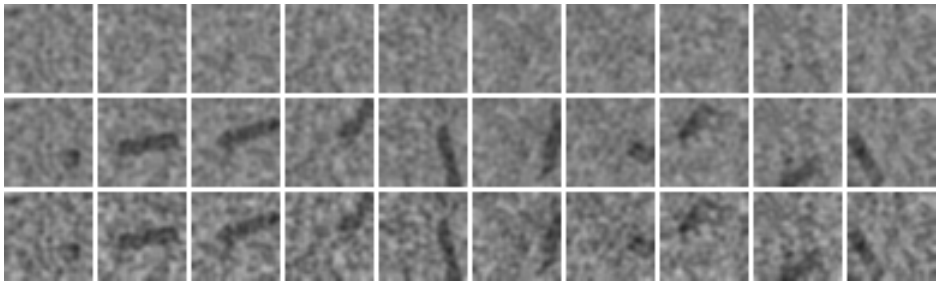
$$\begin{aligned} \delta(\Omega_B^*) &= \sqrt{\sum_{y\in[\Omega_h^*]}\sum_{x\in[\Omega_w^*]}(\Omega_B^*[x,y]_{sat} - \mathbb{E}\, X_{\Omega^*})^2} = \\ &= \sqrt{|\Omega_B^*|\,\mathbb{E}\,(X_{\Omega^*} - \mathbb{E}\, X_{\Omega^*})^2} = \\ &= \sqrt{|\Omega_B^*|}\sqrt{var\, X_{\Omega^*}}, \end{aligned} \tag{3.3}$$

kde Ω_h^*, resp. Ω_w^* značí výšku, resp. šířku oblasti Ω_B^* a $[n]$ značí množinu $\{1, \dots n\}$. Jestliže hodnotu $\delta(\Omega_B^*)$ porovnáváme s prahovou hodnotou $\widehat{\delta}(\Omega_B^*)$, pak z rovnosti

(3.3) plyne, že stejného výsledku dosáhneme, když budeme porovnávat rozptyl $var\, X_{\Omega^*}$ s prahovou hodnotou

$$\widehat{\sigma}^2(\Omega_B^*) = \frac{\widehat{\delta}(\Omega_B^*)^2}{|\Omega_B^*|}. \tag{3.4}$$

Popsaná představa porovnávání hodnoty $\delta(\Omega_B^*)$ s určitou prahovou hodnotou $\widehat{\delta}(\Omega_B^*)$ je tedy ekvivalentní porovnávání rozptylu oblasti Ω_B^* s vhodně zvolenou hodnotou $\widehat{\sigma}^2(\Omega_B^*)$ podle (3.4).



Obrázek 3.4: V prvním řádku vidíme oblasti Ω_1^* a v druhém, resp. třetím řádku, oblasti Ω_2^* pro $\sigma_Q^2 = 0.03$, resp. $\sigma_Q^2 = 0.06$, a poměr saturace textu a pozadí 0.6.

Použitelnou hodnotu $\widehat{\sigma}^2(\Omega_B^*)$ odhadujeme za pomoci metody typu Monte Carlo. Generujeme desítky tisíc oblastí fixní velikosti 11×11 pixelů. Zmíněná velikost je použita i jako velikost okolí procházených pixelů při generování predikční bitmapy. Generujeme dva následující typy oblastí:

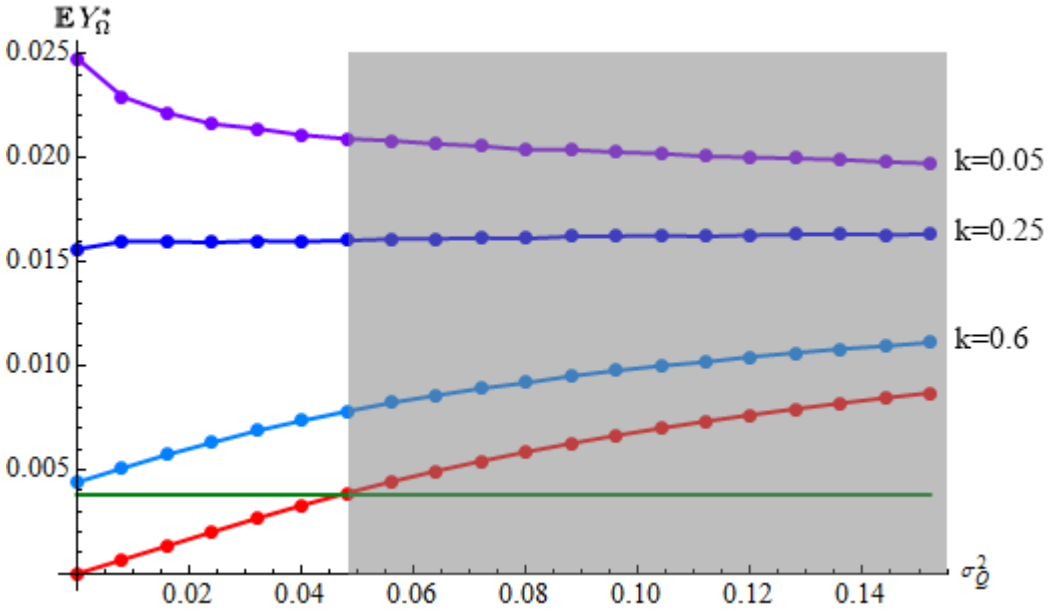
- oblasti Ω_1 reprezentují pozadí dokumentu s konstantní saturací pixelů, do kterých byl následně přidán šum s normálním rozdělením $\mathcal{N}(0, \sigma_Q^2)$,
- oblasti Ω_2 jsou stejné jako předchozí, ale před zašuměním jsme do nich přidali tmavší úsečku s náhodnými pozicemi koncových bodů, která reprezentuje fragment znaku.

Vzorek nagenerovaných dat můžeme vidět na obrázku 3.4. Takto nasimulované oblasti využijeme k odhadu prahové hodnoty $\widehat{\sigma}^2(\Omega_B^*)$.

Pro oblasti Ω_i mějme diskrétní náhodné veličiny $Y_{\Omega_i}^*$ definované jako

$$Y_{\Omega_i}^* = (X_{\Omega_i^*} - \mathbb{E}\, X_{\Omega_i^*})^2,$$

pro $i \in \{1, 2\}$. Náhodné veličiny $Y_{\Omega_i}^*$ jsme definovali tímto způsobem, protože nás zajímá odhad hodnot rozptylu náhodné veličiny $X_{\Omega_i^*}$. Označme konstantou k poměr saturace pixelů textu a pixelů pozadí v oblasti Ω_2. Z nagenerovaných oblastí



Obrázek 3.5: Graf zobrazuje odhadnuté hodnoty $\mathbb{E}\,Y^*_{\Omega_1}$ (červeně) a $\mathbb{E}\,Y^*_{\Omega_2}$ (pro $k = 0.05, k = 0.25, k = 0.6$). Zeleně je vyznačen náš odhad prahové hodnoty $\widehat{\sigma}^2(\Omega^*_B)$.

jsme spočetli odhady hodnot $\mathbb{E}\,Y^*_{\Omega_1}$ a $\mathbb{E}\,Y^*_{\Omega_2}$ v závislosti na hodnotě rozptylu σ^2_Q použitého šumu a hodnoty k. Tyto odhady jsou zaneseny v grafu 3.5. Šedivě vyznačená část grafu určuje oblast s hodnotou rozptylu zaneseného šumu natolik vysokou, že se jí již nebudeme zabývat, protože ji pro rozpoznávání považujeme za příliš zašuměnou. Náš odhad prahové hodnoty $\widehat{\sigma}^2(\Omega^*_B)$ je konstantní, jelikož nejsme schopni dobře určit hodnotu rozptylu šumu zaneseného v dokumentu. To platí především pro hodnoty rozptylu šumu blížící se nule.

Jelikož víme, že znaky v bitmapě Π_B se nacházejí s velkou pravděpodobností v oblastech, v nichž jsou na odpovídajících pozicích v predikční bitmapě $\Pi_{\widetilde{B}}$ pixely se saturací rovnou 0, jsme schopni nálezt množinu $\widetilde{\Pi}^-_B \subseteq \Pi^-_B$ pixelů ležících na pozadí dokumentu a množinu $\widetilde{\Pi}^+_B \subseteq \Pi^+_B$ pixelů tvořících hledané znaky[1]. Množiny $\widetilde{\Pi}^-_B$ a $\widetilde{\Pi}^+_B$ nalezneme tak, že bitmapu Π_B rozdělíme do čtvercové sítě a pro každé její pole nalezneme jeden pixel z pozadí a pokusíme se nalézt pixel reprezentující text.

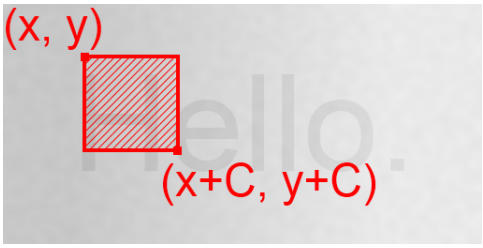
Definice 17. *Nechť Π_B je bitmapa, $x, y \in \mathbb{Z}$ a $C \in \{1, 2, \ldots, \min(B_w, B_h)\}$.*

[1]Nechceme a ani nebudeme hledat maximální možné množiny $\widetilde{\Pi}^-_B$ a $\widetilde{\Pi}^+_B$

Množinou $\Delta_C^B(x, y)$ *označíme množinu* $\Delta_C^B(x, y) \subseteq \Pi_B$*, pro kterou platí*

$$\Delta_C^B(x, y) = \{\rho \,|\, \rho \in \Pi_B \,,\; x_0 \leq \rho_x < x_0 + C \;\wedge\; y_0 \leq \rho_y < y_0 + C\},$$

kde $x_0 = \max(0, \min(B_w - C, x))$ *a* $y_0 = \max(0, \min(B_h - C, y))$*. Ukázku vidíme na obrázku 3.6.*



Obrázek 3.6: Ukázka množiny $\Delta_C^B(x, y)$.

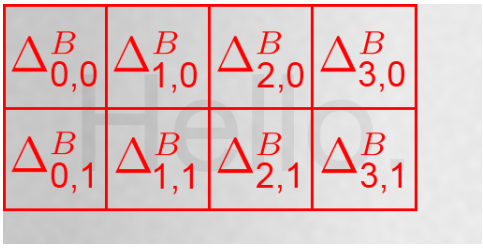
Ve zbylém textu bude $\Delta_{x,y}^B$ značit množinu $\Delta_C^B(x, y)$, kde

$$x \in \{0, C_B, 2C_B, \ldots \left(\left\lfloor \frac{B_w}{C_B} \right\rfloor - 1\right) C_B\} = \Delta_{G_w}^B,$$

$$y \in \{0, C_B, 2C_B, \ldots \left(\left\lfloor \frac{B_h}{C_B} \right\rfloor - 1\right) C_B\} = \Delta_{G_h}^B,$$

$C_B \in \{2, \ldots, \min(B_w, B_h)\}$ je určitý koeficient dělení bitmapy Π_B.

Tyto množiny reprezentují jednotlivá pole zmíněné čtvercové sítě v bitmapě Π_B. Její ukázka je na obrázku 3.7.



Obrázek 3.7: Ukázka množin $\Delta_{x,y}^B$.

Předpokládáme, že

$$\forall x \in \Delta_{G_w}^B \forall y \in \Delta_{G_h}^B \exists \rho \in \Delta_{x,y}^B \colon \tau_B(\rho) = 0.$$

To znamená, že každé pole naší čtvercové sítě musí obsahovat alespoň jeden pixel z pozadí. Tento předpoklad se zdá být přirozený, ale přináší jedno omezení. Nebudeme moci vytěžovat dokumenty Π_B se znaky obsahujícími čtvercové oblasti velikosti C_B^2 složené pouze z pixelů, které mají hodnotu indikátoru textu rovnu 1.

Potom budeme $\widetilde{\Pi}_B^-$ a $\widetilde{\Pi}_B^+$ konstruovat tak, aby pro $\widetilde{\Pi}_B^-$ platilo, že

$$\forall x \in \Delta_{G_w}^B \forall y \in \Delta_{G_h}^B \exists! \rho \in \Delta_{x,y}^B : \rho \in \widetilde{\Pi}_B^- \tag{3.5}$$

a pro množinu $\widetilde{\Pi}_B^+$

$$\begin{aligned}&\forall x \in \Delta_{G_w}^B \forall y \in \Delta_{G_h}^B : \\ &\quad (\exists \rho \in \Delta_{x,y}^B : \tau_B(\rho) = 1) \Rightarrow (\exists! \rho \in \widetilde{\Pi}_B^+ \colon \rho \in \Delta_{x,y}^B \wedge \tau_B(\rho) = 1).\end{aligned}$$

To znamená, že jednotlivé prvky množiny $\widetilde{\Pi}_B^-$, resp. $\widetilde{\Pi}_B^+$ hledáme tak, že postupně procházíme všechny množiny $\Delta_{x,y}^B$ a pro každou nalezneme jeden pixel reprezentující pozadí (ten jistě existuje podle (3.5)) a v případě existence i pixel reprezentující text.

Hledání prvku v $\Delta_{x,y}^B$ náležícímu $\widetilde{\Pi}_B^+$:

(1) Nalezneme pixel $\widetilde{\rho} \in \Delta_{x,y}^{\widetilde{B}}$, pro který platí, že $\widetilde{\rho}_{sat} = 0$. Pokud takový pixel neexistuje, tak jsme pro danou oblast $\widetilde{\Pi}_B^+$ vhodný pixel nenalezli a pokračujeme v hledání pixelu z pozadí pro množinu $\widetilde{\Pi}_B^-$.

(2) Nalezneme pixel $\rho^* \in \Delta_{x,y}^{B^*}$ tak, aby platilo

$$\rho^* \in \operatorname*{argmin}_{\pi \in \Omega} \pi_{sat},$$

kde $\Omega = \Delta_E^B(\widetilde{\rho}_x, \widetilde{\rho}_y)$ a $E \in \{1, \ldots, C_B\}$ je konstanta pro bitmapu Π_B určující velikost okolí, v kterém se má hledat.

(3) Pixel $B[\rho_x^*, \rho_y^*]$ zvolíme jako výsledný pixel reprezentující pozadí v oblasti $\Delta_{x,y}^{B^*}$.

Hledání prvku v $\Delta_{x,y}^B$ náležícímu $\widetilde{\Pi}_B^-$:

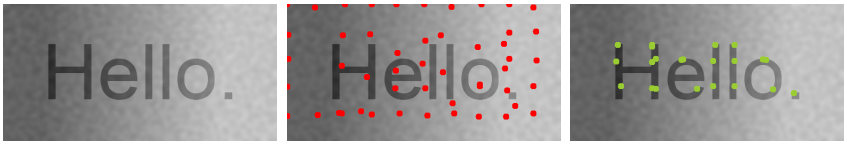
(1) Jestliže jsme v oblasti $\Delta_{x,y}^B$ nenalezli vhodný prvek pro $\widetilde{\Pi}_B^+$, tak prvek náležející množině $\widetilde{\Pi}_B^-$ nalezneme analogicky jako v předchozím postupu. V opačném případě nalezneme pixel $\widetilde{\rho} \in \Delta_{x,y}^{\widetilde{B}}$, pro který platí, že $\widetilde{\rho}_{sat} = 1$.

(2) Nalezneme pixel $\rho^* \in \Delta_{x,y}^{B^*}$ tak, aby platilo

$$\rho^* \in \operatorname*{argmax}_{\pi \in \Omega} \pi_{sat},$$

kde Ω a E jsou definovány stejně jako při konstrukci $\widetilde{\Pi}_B^+$. Sekundárním požadavkem na pixel ρ^* je jeho vzdálenost od pixelu nalezeného pro množinu $\widetilde{\Pi}_B^+$ v oblasti $\rho^* \in \Delta_{x,y}^{B^*}$, aby byla maximální. Tento požadavek klademe z důvodu snížení pravděpodobnosti chybného označení pixelu reprezentujícího text jako pixelu reprezentujícího pozadí.

(3) Pixel $B[\rho_x^*, \rho_y^*]$ zvolíme jako výsledný pixel reprezentující text v dané oblasti.



(a) Původní dokument (b) Množina $\widetilde{\Pi}_B^-$ (c) Množina $\widetilde{\Pi}_B^+$

Obrázek 3.8: Ukázkový dokument s nalezenými množinami $\widetilde{\Pi}_B^-$ a $\widetilde{\Pi}_B^+$.

Ve chvíli, kdy známe množiny $\widetilde{\Pi}_B^-$ a $\widetilde{\Pi}_B^+$, určíme podle sekce 3.1.3, zda je právě zpracovávaný dokument Π_B zašuměný. Pokud ano, nahradíme saturace pixelů původní bitmapy Π_B saturacemi pixelů Gaussovsky rozmazané bitmapy Π_B^*. Tedy

$$\Pi_B := \{\rho \,|\, \rho \in \Pi_B^*\}.$$

Jestliže bychom toto nahrazení provedli i pro nezašuměný dokument, vedlo by to k nežádoucímu rozšíření jednotlivých symbolů ve výsledné bitmapě. Nyní máme vše potřebné připraveno a můžeme přejít k vlastnímu čistícímu algoritmu.

Pro každý pixel ρ právě čištěné bitmapy Π_B nalezneme k němu nejbližší pixely $\omega^1, \omega^2 \in \widetilde{\Pi}_B^-$ a $\delta^1, \delta^2 \in \widetilde{\Pi}_B^+$. To znamená, že

$$\omega_1 \in \operatorname*{argmin}_{\pi \in \widetilde{\Pi}_B^-} \operatorname{dist}(\rho, \pi) \quad \text{a} \quad \omega_2 \in \operatorname*{argmin}_{\pi \in \widetilde{\Pi}_B^- \setminus \{\omega^1\}} \operatorname{dist}(\rho, \pi),$$

$$\delta_1 \in \operatorname*{argmin}_{\pi \in \widetilde{\Pi}_B^+} \operatorname{dist}(\rho, \pi) \quad \text{a} \quad \delta_2 \in \operatorname*{argmin}_{\pi \in \widetilde{\Pi}_B^+ \setminus \{\delta^1\}} \operatorname{dist}(\rho, \pi).$$

Nyní pro každý pixel $\rho \in \Pi_B$ odhadneme hodnoty $\widehat{\omega}, \widehat{\delta} \in [0,1]$. Hodnota $\widehat{\omega}$, resp. $\widehat{\delta}$ tvoří odhad saturace pixelu ρ na základě hodnot ω^1, ω^2, resp. δ^1, δ^2 za předpokladu, že $\tau_B(\rho) = 1$, resp. $\tau_B(\rho) = 0$. Budeme předpokládat, že hodnoty saturací textu a pozadí se mění spojitě. Hodnotu $\widehat{\omega}$, resp. $\widehat{\delta}$ odhadneme pomocí váženého průměru saturací pixelů ω^1, ω^2, resp. δ^1, δ^2 závislého na vzdálenosti od pixelu ρ. Tedy

$$\widehat{\omega}_{sat} = \frac{\omega^1_{sat} \operatorname{dist}(\rho, \omega^1) + \omega^2_{sat} \operatorname{dist}(\rho, \omega^2)}{\operatorname{dist}(\rho, \omega^1) + \operatorname{dist}(\rho, \omega^2)},$$

$$\widehat{\delta}_{sat} = \frac{\delta^1_{sat} \operatorname{dist}(\rho, \delta^1) + \delta^2_{sat} \operatorname{dist}(\rho, \delta^2)}{\operatorname{dist}(\rho, \delta^1) + \operatorname{dist}(\rho, \delta^2)}.$$

Nakonec algoritmus projde bitmapu Π_B a pro každý její pixel zjistí zda je svou saturací blíže odhadu saturace pozadí $\widehat{\omega}$ nebo saturace textu $\widehat{\delta}$ a podle toho mu přiřadí novou saturaci a to buď 1 pro pozadí nebo 0 pro text. Výstup algoritmu vidíme na obrázku 3.9.

(a) Původní dokument (b) Vyčištěný dokument

Obrázek 3.9: Ukázka práce čistícího algoritmu.

3.2 Rozpoznávání symbolů

3.2.1 Základní algoritmus

V okamžiku, kdy máme extrahované jednotlivé znaky z dokumentu, můžeme se začít soustředit na vlastní rozpoznávání. To bude založené na vícevrstvých neurálních sítích s dopředným šířením [1]. V našem případě používáme sítě s dvěma skrytými vrstvami. Neurální síť je funkce

$$f\colon \mathbb{R}^m \to \mathbb{R}^n, \text{ kde } m, n \in \mathbb{N},$$

která nám na základě vstupního vektoru velikosti m vydá výsledek o velikosti n. Jelikož budeme našim neurálním sítím předávat data obsažená v extrahovaných

znacích, je nutné abychom sjednotili jejich velikosti. Ještě před vlastním rozpoznáváním extrahované znaky zmenšíme popřípadě zvětšíme na nějakou předem danou velikost. V našem případě jsme jako postačující velikost zvolili 15×18 pixelů. Výška je větší než šířka, neboť většina písmen rozpoznávané abecedy je vyšší než širší. Při použití větších rozměrů již nedochází ke zlepšení rozpoznávání neurálními sítěmi. Změnu velikosti provádíme pomocí algoritmu nejbližšího souseda. Tedy pro bitmapu B' reprezentující znak se znormalizovanou velikostí a původní bitmapu B s originálním znakem platí, že

$$\forall \rho' \in B' \colon \rho'_{sat} = B\left[\left\lfloor \frac{\rho'_x B_w}{B'_w} \right\rfloor, \left\lfloor \frac{\rho'_y B_h}{B'_h} \right\rfloor\right].$$

Vektorem zkonstruovaným z bitmapy Π_B pro neurální síť budeme rozumět vektor jehož složky jsou tvořeny jednotlivými hodnotami saturací pixelů bitmapy Π_B a popřípadě dalšími hodnotami, jak bude popsáno níže.

Definice 18. *Nechť Π_B je bitmapa a $f \colon \mathbb{R}^{B_N} \to \mathbb{R}^n$, kde $B_N, n \in \mathbb{N}$. Zavedeme následující značení:*

- *B_N je velikost konstruovaného vektoru z bitmapy B pro neurální síť,*
- $\mathrm{RI}_f(\Pi_B) = \underset{i \in \{1,\ldots,n\}}{\operatorname{argmax}} (f(\Pi_B))_i$,
- $\mathrm{RV}_f(\Pi_B) = \max_{i \in \{1,\ldots,n\}} (f(\Pi_B))_i$,

kde $f(\Pi_B)$ značí hodnotu funkce f ve vektoru velikosti B_N zkonstruovaném z Π_B pro neurální síť reprezentovanou funkcí f.

Poznámka. *Hodnota $\mathrm{RI}_f(\Pi_B)$, resp. $\mathrm{RV}_f(\Pi_B)$ určuje index složky vektoru, resp. hodnotu složky s maximální hodnotou v daném vektoru.*

Použití mechanismu rozpoznávajícího jednotlivé znaky právě na základě hodnot $\mathrm{RI}_f(\Pi_B)$ je následující. Budeme mít jednu neurální síť a množinu Σ reprezentující námi rozpoznávanou abecedu znaků. Budeme mít extrahovaný znak s již znormalizovanou velikostí reprezentovaný bitmapou Π_B a neurální síť popsanou funkcí

$$f : \mathbb{R}^{|\Pi_B|} \to \mathbb{R}^{|\Sigma|},$$

přičemž hodnota $\mathrm{RI}_f(\Pi_B)$ bude určovat index symbolu v abecedě Σ. Při testování tohoto přístupu jsme došli k závěru, že vzhledem k velikosti množiny Σ, kterou chceme rozpoznávat, to je poměrně vysoký nárok na jednu neurální síť,

která potom nedostatečně kvalitně klasifikuje jednotlivé znaky. Pokud bychom síť trénovali pouze na jeden font, tak bez větších problémů zvládne rozpoznávat celou abecedu. K horší klasifikaci začne docházet až ve chvíli, kdy jednu síť začneme trénovat pro větší množství různých fontů. Z toho důvodu v sekci 3.2.2 uvádíme způsob, kterým jsme zredukovali negativní vliv množství trénovaných fontů na rozpoznávání.

V části 3.1.1 jsme zmínili problém, ke kterému může dojít během extrakčního algoritmu. Extrahujeme oblast, která obsahuje více spojených znaků. Nejčastěji se s tímto problémem setkáváme, při spojení dvou písmen. Ke spojení více znaků dochází výrazně méně častěji a proto se budeme zaobírat pouze řešením rozpoznávání oblastí s maximálně dvěma znaky. Pro lepší představu po vyčištění obrázku 4.2 v něm dochází ke spojení dvou písmen osmkrát a ke spojení více než dvou písmen ani jednou. V extrahované oblasti Π_B se pokusíme rozpoznat dva znaky jestliže

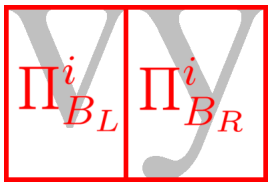
$$B_w > B_h.$$

Potom postupně konstruujeme posloupnost dvojic oblastí $\Pi^i_{B_L}, \Pi^i_{B_R} \subseteq \Pi_B$, kde

$$\Pi^i_{B_L} = \{\rho \,|\, \rho \in \Pi_B \ \wedge \ \rho_x \leq i\},$$

$$\Pi^i_{B_R} = \{\rho \,|\, \rho \in \Pi_B \ \wedge \ \rho_x > i\},$$

pro $i \in \{1, \ldots, B_w - 1\}$, které reprezentují všechna možná vertikální dělení oblasti Π_B. Můžeme je vidět na obrázku 3.10. Ve všech oblastech $\Pi^i_{B_L}$ a $\Pi^i_{B_R}$ spustíme



Obrázek 3.10: Ukázka množin $\Pi^i_{B_L}, \Pi^i_{B_R}$.

standardní rozpoznávání jednoho symbolu a nalezneme j takové, že

$$j = \underset{i \in \{1,\ldots,B_w-1\}}{\operatorname{argmax}} (\mathrm{RV}_f(\Pi^i_{B_L}) + \mathrm{RV}_f(\Pi^i_{B_R})).$$

Pokud

$$\mathrm{RV}_f(\Pi^j_{B_L}) + \mathrm{RV}_f(\Pi^j_{B_R})) > 2\,\mathrm{RV}_f(\Pi_B),$$

tak pro oblast Π_B jako výsledek vrátíme dva znaky určené v aktuálně rozpoznávané abecedě indexy $\mathrm{RI}_f(\Pi^j_{B_L})$ a $\mathrm{RI}_f(\Pi^j_{B_R})$. V opačném případě vrátíme jeden znak určený indexem $\mathrm{RI}_f(\Pi_B)$.

3.2.2 Vylepšený rozpoznávací proces

Nejvýraznější změnou je zvýšení počtu použitých neurálních sítí a jejich postavení do stromové struktury. To znamená, že pro bitmapu B máme jednu hlavní neurální síť, která je funkcí $f : \mathbb{R}^{|B_N|} \to \mathbb{R}^n$, kde n je počet specializovaných neurálních sítí a číslo $\mathrm{RI}_f(\Pi_B)$ určuje, kterou specializovanou neurální síť použít. Po tomto výpočtu extrahovaný symbol položíme jako vstup vybrané specializované neurální sítě, která je funkcí $g : \mathbb{R}^{|B_N|} \to \mathbb{R}^{|\Sigma|}$, jež určí index písmena v abecedě Σ, nebo $f' : \mathbb{R}^{|B_N|} \to \mathbb{R}^m$, která určí další specializovanou neurální síť, kde $m \in \mathbb{N}$ určuje počet specializovaných sítí pro danou množinu znaků. Struktura sítí je konstruována tak, aby platilo, že pokud máme neurální síť N ve tvaru $f : \mathbb{R}^{|B_N|} \to \mathbb{R}^n$ určující specializovanou síť N_i, kde $i \in \{0, \ldots, n\}$ pro znak z množiny $A \subseteq \Sigma$, potom pro množiny znaků A_i rozpoznávanými sítěmi N_i platí, že

$$A = \bigcup_{k=0}^{n} A_k,$$

$$A_i \cap A_j = \emptyset, \text{ pro } i \neq j \text{ a } i, j \in \{0, \ldots, n\}.$$

My jsme zvolili následující strukturu, kde počáteční rozpoznávaná abeceda je celá Σ a vypsané množiny jsou rozpoznávané specializovanými sítěmi (čím níže v hierarchii je množina umístěna, tím více specializovanou neurální sítí je rozpoznávána):

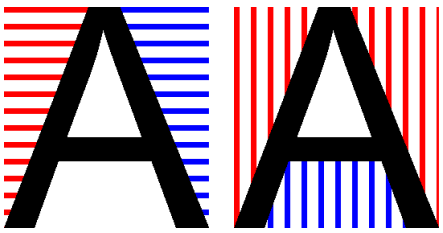
- celá rozpoznávaná abeceda Σ
  - { 'c', 'C', 'o', 'O', '0', 'Q', 'D', 'G', '*', '.', ',', '‘', '´', '˘' }
    - { 'c', 'C', 'G' }
    - { 'o', 'O', '0', 'Q', 'D' }
    - { '*', '.', ',', '‘', '´', '˘' }
  - { 'v', 'V', 'u', 'U', 'w', 'W', 'x', 'X', 'N', 'M', 'm', 'n', '+', '∼', '-' }
  - { '{', '}', '[', ']', '(', ')', 'i', 'I', '/', '\', 'j', 'J', 'l', '1', 'f' }
    - { '[', ']', 'i', 'I', '/', '\', 'l', '1' }
    - { '{', '}', '(', ')', 'j', 'J', 'f' }
  - { 'b', 'B', 'd', 'p', 'P', 'q', 'R', '8', '3', 'a', '4', '6', '9', 'e', 'g', '@' }
  - { 'E', 'F', 'H', 'k', 'K', 'L', 'T', 'A', 't', 'y', 'Y', 'h', 'r', '&', '#' }
  - { '2', 'z', 'Z', 's', 'S', '5', '7', '?', '$', '<', '>', 'ˆ' }

Rozdělení rozpoznávané abecedy do těchto skupin jsme provedli na základě subjektivní podobnosti jednotlivých symbolů. Lepších výsledků by se pravděpodobně dalo dosáhnout s použitím klasifikátoru vektorů vstupních dat neurálních sítí, který by nám vrátil přirozenější rozdělení naší abecedy do určitého počtu skupin.

Dalším postupem, který vede k lepšímu rozpoznávání je navýšení množství informací předávaných neurálním sítím. K vlastním saturacím jednotlivých pixelů extrahovaných znaků navíc přidáme čtyři vektory. Každý tento vektor bude přiřazen jedné hraně obdélníku obsahujícího extrahovaný symbol a bude obsahovat hodnoty, které budou vypovídat o vzdálenosti pixelů s nulovou saturací od dané hrany. Tyto vektory budou jinak řečeno popisovat zevní tvar extrahovaného znaku. Ukázku těchto vzdáleností vidíme na obrázku 3.11. K těmto informacím nakonec ještě přidáme normalizované velikosti stran $B_{w'}$ a $B_{h'}$ extrahovaného symbolu reprezentovaného bitmapou B definované jako

$$B_{w'} = \frac{B_w}{\max(B_w, B_h)}, \; B_{h'} = \frac{B_h}{\max(B_w, B_h)},$$

kde B_w, resp. B_h je šířka, resp. výška bitmapy B. Normalizovaná velikost znaku vypovídá o poměru stran, protože při převodu extrahovaných znaků na jednotnou velikost se informace o poměru stran ztrácí. Důvodem pro výpočet dvou hodnot místo pouhého podílu obou velikostí je, že chceme mít hodnoty předávané neurálním sítím z intervalu $[0, 1]$.



(a) Vzdálenosti od levého a pravého okraje

(b) Vzdálenosti od horního a spodního okraje

Obrázek 3.11: Vektory reprezentující vzdálenosti písmena od okrajů.

3.2.3 Neurální sítě a jejich učení

Pro trénování neurálních sítí automaticky generujeme množství jednotlivých znaků z množin rozpoznávaných symbolů. Data generujeme pro většinu standardních fontů. Navíc pro každý symbol s danou velikostí generujeme čtyři různé verze, které se odvíjejí od toho, zda je písmeno tučné nebo normální a zda je vyhlazené nebo ne. Každý z těchto typů generujeme ve třech různých velikostech. K učení jednotlivých neurálních sítí používáme pro jednoduchost algoritmus zpětného šíření popsaný v [1] i přesto, že existují rychlejší možnosti, které jsou nad rámec naší práce, neboť učení se řeší pouze při přípravě algoritmu. Učící algoritmus nemá vliv na rychlost samotného rozpoznávání.

3.3 Zpracování rozpoznaných symbolů

Tato práce je zaměřena především na extrakci znaků, a proto jsme implementovali pouze jednoduché a provizorní algorimy zajišťující kompozici znaků a dělení symbolů do řádků a slov.

3.3.1 Kompozice symbolů

Všechna písmena s diakritikou a další znaky složené z více symbolů náš extrakční algoritmus rozpozná jako několik nezávislých znaků. Jako výsledek obdržíme pouze pozice s přiřazenými pravděpodobnými symboly, které teprve tvoří jednotlivá písmena. Musíme tedy seznam výsledků extrakce analyzovat a pospojovat patřičné znaky dohromady. V aplikaci máme definován seznam trojic znaků, kde jednotlivé složky reprezentují postupně:

- znak nacházející se nad výsledným písmenem (háček, čárka, tečka, apod.),
- znak, který může být svázán se symbolem z první složky,
- výstupní znak, který vznikne spárováním prvních dvou složek této trojice.

Každé trojici dále náleží příznak o tom, v jaké poloze by se vstupní znaky vůči sobě měly nacházet, aby mohly být spojeny. Vzhledem k nepříliš vysokému množství znaků na standardním listu textu si můžeme dovolit použít jednoduchý algoritmus se složitostí $O(n^2)$, který pro každý rozpoznaný znak projde všechny ostatní znaky a otestuje, zda vyhovuje podmínkám popisujícím složené znaky.

3.3.2 Dělení vytěžených znaků do řádků a slov

Než můžeme vytěžený text exportovat, musíme jednotlivé znaky seskupit do slov a řádků. Nalezneme skupiny znaků patřící do jednotlivých řádků a do řádků přidáme mezery tak, abychom je rozdělili do slov.

- setřídíme znaky podle x-ové souřadnice,
- pro každý znak z^0, který ještě nebyl zpracován:
  - založíme nový řádek w se stejnou pozicí jako má z^0
  - pro každý znak z^1, který ještě nebyl zpracován:
    - pokud se intervaly $[z_y^0, z_y^0 + z_h^0]$ a $[z_y^1, z_y^1 + z_h^1]$ překrývají alespoň z 40% a z^1 je dostatečně blízko konci řádku w, tak z^1 přidáme do slova w a označíme ho jako zpracované
  - z^0 označíme jako zpracované,
- pro každý řádek w:
  - postupně procházíme znaky obsažené v w a na základě vzdálenosti pozice aktuálního znaku a konce předchozího znaku na x-ové ose určíme počet mezer, které mezi tyto dva znaky vložíme.

Poznámka. *Máme-li znak z, tak z_y značí y-ovou souřadnici pozice znaku z a z_h značí výšku znaku z.*

3.3.3 Korekce vytěženého textu

Při rozpoznávání jednotlivých znaků často dochází k chybám u velmi podobných znaků, jako jsou například '1', 'l', 'I'. Proto využíváme tři typy korekcí.

- **Analýza velikosti okolních znaků** - máme-li ve slově malé písmeno z a okolo něj jsou písmena velká, tak je pravděpodobné, že z mělo být taky velké a zaměníme ho za velké písmeno podobné z. Analogicky pro velké písmeno mezi malými. Tento postup pochopitelně provádíme pouze u písmen, kde si malé a velké písmeno je podobné.
- **Analýza typu okolních znaků** - Podobně postupujeme, když se číslo nachází ve slově tvořeném z písmen. Zaměníme ho za písmeno podobné danému číslu. To se týká především čísel '0' a '1'.

- **Znaky složené z jedné části byly rozpoznané jako složený znak** - při kompozici znaků si pamatujeme, které znaky byly změněny a pokud narazíme na znak složený z více částí a zároveň víme, že tento znak nevznikl při kompozici, tak ho změníme na podobný znak složený pouze z jedné části.

Další možnou korekcí by mohla být nějaká slovníková metoda, která by porovnávala vytěžená slova s existujícím slovníkem a na jeho základě prováděla korekce. Syntaktická analýza tohoto typu je nad rámec této práce.

Kapitola 4

Závěr

4.1 Dosažené výsledky

V práci jsme předvedli komplexní řešení problému extrakce a rozpoznávání textu z digitálních dokumentů. Systém úspěšně implementuje:

- hlavní těžiště práce:
  - determinaci šumu v dokumentech
  - čistící algoritmus
  - extrakci symbolů
- plánujeme vylepšení:
  - rozpoznání a kompozice symbolů
  - umístění rozpoznaných znaků v dokumentu.

Tyto funkční části jsou na sebe navázány a pracují automaticky bez nutnosti zásahu uživatele. Nespornou výhodou je možnost zpracování špatně osvětlených a zašuměných dokumentů. Systém jsme otestovali na generovaných datech i na reálných dokumentech a dosáhli jsme uspokojivých výsledků.

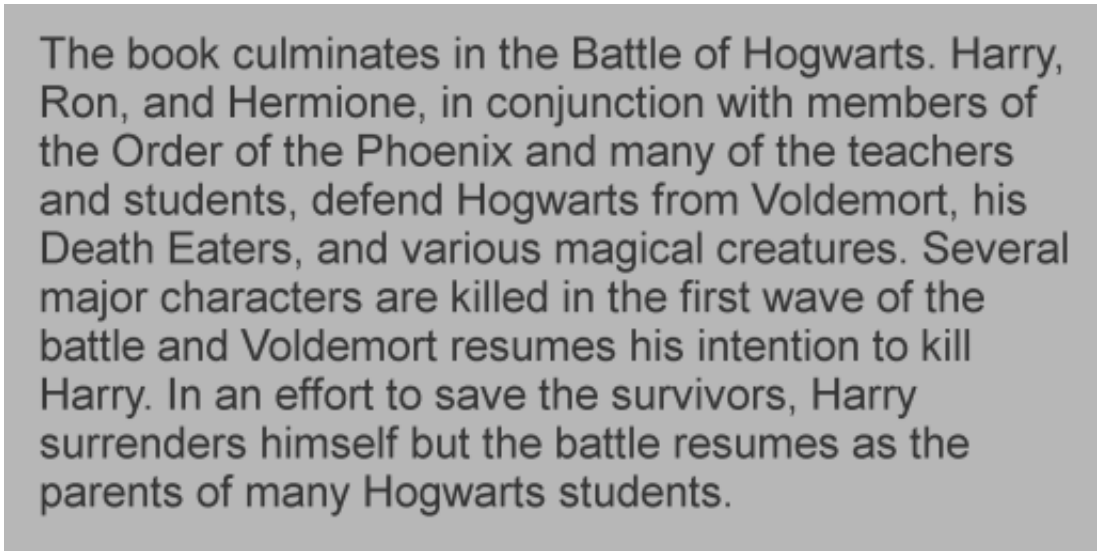
4.2 Úspěšnost rozpoznávání

V tabulce 4.1 vidíme pravděpodobnosti úspěšného rozpoznání jednotlivých znaků v dokumentech 4.1 a 4.2 pro různé hodnoty rozptylu přidaného šumu. Pravděpodobnost není vypočítána na základě exportovaného textu, ale jednotlivých

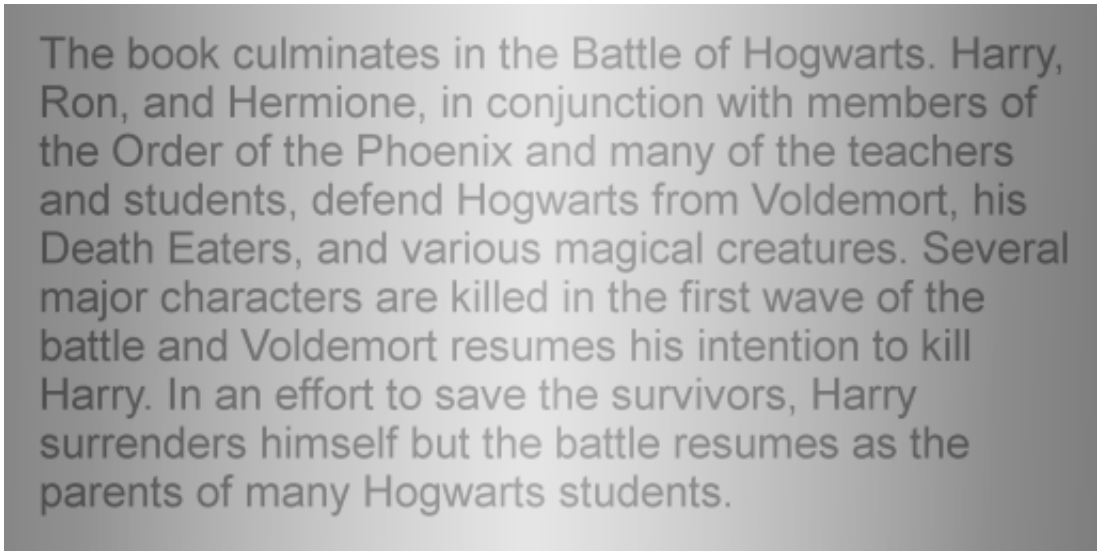
znaků. Zpracování rozpoznaných znaků do slov a řádků jsme v této práci nevěnovali takovou pozornost a předvedený algoritmus je spíše provizorní. Z toho důvodu algoritmus netestujeme a nemáme žádné nároky na jeho výstup.

Tabulka 4.1: Úspěšnost rozpoznání jednotlivých symbolů

Rozptyl šumu	Úspěšnost	
	Dokument 4.1	Dokument 4.2
0.00	92%	93%
0.01	91%	91%
0.02	90%	89%
0.03	90%	88%
0.04	90%	84%
0.05	89%	81%



Obrázek 4.1: Testovaný dokument 1.



Obrázek 4.2: Testovaný dokument 2.

4.3 Budoucí práce

Ve všech prezentovaných postupech je zajisté prostor ke zlepšení. Při extrakci znaků z dokumentů jsme uvedli množství předpokladů, které by bylo vhodné zredukovat. Například bychom se mohli zabývat větší rezistencí vůči obsaženému šumu, nižšími nároky na nezbytný odstup saturace textu od pozadí nebo dynamickou volbou použitých konstant. Vlastní rozpoznávání budeme schopni zlepšit přirozenějším rozdělením rozpoznávané abecedy do skupin a předáváním většího množství relevantních informací vypovídajících o symbolech neurálním sítím. Algoritmy pro rozdělení vytěžených znaků do slov a řádků, kterými jsme se zabývali pouze okrajově, hodláme v budoucnu rozšířit, jelikož jsou pro použitelnost celého systému nezbytné.

Literatura

[1] Michael A. Arbib. *The Handbook of Brain Theory and Neural Networks*. The MIT Press, 2003.

[2] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc, 2006.

[3] Shen-Chuan Tai and Shih-Ming Yang. A fast method for image noise estimation using laplacian operator and adaptive edge detection. *ISCCSP*, 2008.