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

ČTEČKA BRAILLOVA PÍSMA

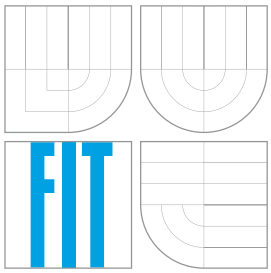
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN VICIÁN

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

ČTEČKA BRAILLOVA PÍSMA

BRAILLE READER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN VICIÁN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ MARŠÍK

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá metodami detekce a dekódování digitalizovaných dokumentů v Braillově písmu do latinky. Popisuje funkci skenerů a jejich použití pro digitalizování Braillova písma. Analyzuje různé metody prahování, detekce rotace a zpracování znaků. Popisuje implementovaný program v jazyku C++, knihovně OpenCV a Qt frameworku. Na závěr navrhuje nejvhodnější postup dekódování.

Abstract

The bachelor's paper deals with methods of detection and decoding digitalized documents in braille to Latin script. It describes the function of scanners and their employment in digitalizing braille. It analyzes various methods of thresholding, rotation detection and sign processing. It describes an implemented program in C++ language, the OpenCV library and Qt framework. Finally it proposes the most feasible procedure of decoding. declaration

Klíčová slova

Braillovo písmo, prahování, rozpoznávání obrazu, zpracování obrazu, C++, OpenCV, Qt framework.

Keywords

Braille system, thresholding, image recognition, image processing, C++, OpenCV, Qt framework.

Citace

Martin Vicián: Čtečka Braillova písma, bakalářská práce, Brno, FIT VUT v Brně, 2013

Čtečka Braillova písma

Prohlášení

.....
Martin Vicián
14. května 2013

Poděkování

Děkuji především Ing. Lukáši Maršíkovi za odborné vedení této bakalářské práce a společnosti Tyflokabinet, která ochotně poskytla všechny testovací dokumenty a umožnila konfrontaci práce s praxí.

Obsah

Kapitola 1

Úvod

V této bakalářské práci se budu zabývat převodem digitalizovaných dokumentů v Braillově písmu do latinky. Navržený program, který je hlavním předmětem mojí práce, usnadňuje práci s těmito dokumenty bez znalosti Braillovy abecedy. Krátce popíšu její historii, důkladněji pak definování rozměrů znaku a jazykové normy abecedy.

Dále se budu zabývat možnostmi digitalizace těchto dokumentů pomocí běžně dostupných skenerů, u kterých na základě funkčnosti budu ověřovat, zda je lze použít pro skenování trojrozměrného Braillova písma a za jakých podmínek.

Následně analyzuji jednotlivé kroky dekódování; dále se budu zabývat možnými metodami převodu obrazu na stupně šedi, Gaussovým rozostřením obrazu, používanými i navrženými thresholdingy včetně jejich srovnání; redukcí šumu, detekcí rotace a metodami vyrovnání dokumentu. Uvedu použitelná vstupní data metod, zvláště se budu zabývat převodem obrazových dat do matematických struktur s následnými korekcemi a eliminacemi detekce znaku.

Pro uvedené metody navrhnu postup detekce, který implementuji pomocí jazyka C++, knihovny OpenCV a Qt frameworku. Program použiji pro zhodnocení metod a navržení vhodných parametrů pro automatické dekódování. V závěru navrhnu budoucí možné modifikace a vylepšení programu.

Kapitola 2

Braillovo písmo

Podle organizace WHO (World Health Organization) bylo v roce 2012 na světě 285 miliónů lidí se zrakovými vadami (39 miliónů nevidomých a 246 miliónů nevidomých částečně). To odpovídalo přibližně 4,5 % celkové populace Země. V roce 2002 to bylo přibližně 161 miliónů (≈2,6 %). [15]

Dorozumívání s nevidomými se obecně považuje za jednodušší než s neslyšícími. Není totiž třeba učit se kvůli němu nějaké zvláštní řeči nebo novému způsobu komunikace; to se ovšem týká jen verbální komunikace. Při textové nebo obrazové komunikaci je situace komplikovanější. Obrazová komunikace je používaná převážně pro děti, slouží k poznávání okolního světa a rozvoji hmatu. To je nezbytnou přípravou a výborným tréninkem k výuce a používání některého z různých druhů písma určených pro nevidomé. [3]

Braillovo písmo je nejrozšířenější a nejuniverzálnější písmo pro nevidomé. Existují různé alternativy, například písmo Moonovo (viz kapitola 2.6).

2.1 Historie Braillova písma

Braillovo písmo nese název podle francouzského učitele Louise Brailla. Jeho vznik však ovlivnil už Napoleon a jeho kapitán Charles Barbier (1767 - 1841). Ten byl pod Napoleonovým vedením pověřen vytvořením tajného písma pro vojenské účely. Mělo jím být usnadněno dorozumívání vojáků v noci, muselo být proto nenápadné - čitelné bez použití světla. Jeho první verze pochází z roku 1976. Papír byl propichován pomocí nože, vzhledem k profilu jeho břitu se tak vytvářely trojúhelníkové tvary dírek.

Barbier nakonec v roce 1815 sestavil tzv. noční písmo tvořené z 12 reliéfních bodů (šest vertikálních a šest horizontálních). Písmo bylo navíc koncipováno jako fonetické. Dekódovalo 36 rozdílných zvuků (obsahovalo většinu znaků francouzské abecedy). Pro vojáky bylo toto písmo příliš složité a i pro nevidomé nebylo snadné, neboť kvůli své rozměrnosti neumožňovalo rozpoznání jednoho znaku jedním dotykem. Viz obrázek 2.1.

I přesto bylo toto písmo v roce 1821 zavedeno v Národním ústavu pro mladé slepce v Paříži, protože žádné lepší neměli v té době k dispozici. [8]

2.1.1 Louis Braille

Narodil se v roce 1809 poblíž Paříže jako syn sedláře. Ve třech letech si v otcově dílně poranil levé oko. Dostal infekci, která se přenesla i na druhé oko, a postupně oslepl. V deseti letech dostal stipendium v již zmíněném Národním ústavu pro mladé slepce, v němž se později stal profesorem.

	1	2	3	4	5	6
1	a	i	o	u	é	è
2	an	in	on	un	eu	ou
3	b	d	g	j	v	z
4	p	t	q	ch	f	s
5	l	m	n	r	gn	ll
6	oi	oin	ian	ien	ion	ieu

Obrázek 2.1: Tajné písmo vynalezené Charlesem Barierem

V patnácti letech vynalezl systém vyražených bodů inspirovaný právě Barbierovým písmem. Redukoval dvanáct bodů na šest a písmo zdokonalil. V roce 1829 vydal první knihu a v roce 1837 písmo ještě vylepšil přidáním znamének pro zápis matematických a hudebních symbolů. V roce 1932 začal toto písmo používat celý svět. [9]

2.2 Definice

Každý znak Braillova písma tvoří mřížka šesti bodů uspořádaných do obdélníku $2 * 3$. Na každém z těchto šesti míst buď bod (vyvýšené místo) je, nebo není. Tímto způsobem je možno zakódovat $2^6 = 64$ různých znaků, z nichž se ovšem úplně prázdný znak používá jako mezera, takže zbývá 63 použitelných znaků.

Základní sada obsahuje znaky pro písmena a interpunkci. Velká písmena a číslice se zapisují pomocí speciálního symbolu a některého z těchto základních znaků.

2.2.1 Varianty Braillova písma

Existují také způsoby zápisu dalších symbolů (jako např. z matematiky či hudby). Později byla k šestibodové mřížce přidána další řada dvou bodů, čímž počet použitelných symbolů vzrostl na 255. Takovou sadou symbolů je možno zapisovat všechny znaky ASCII[1]. Sada všech těchto symbolů je součástí standardu Unicode (Braille Patterns, U+2800 - U+28FF). Osmibodové Braillovo písmo se však v praxi příliš nepoužívá. [14]

2.3 Jazykové odlišnosti

Každý jazyk má Braillovo písmo upravené pro své potřeby. Tato nejednotnost systému je výrazným problémem. Například jen mezi českou a slovenskou normou je už v základních písmenech abecedy sedm rozdílů.

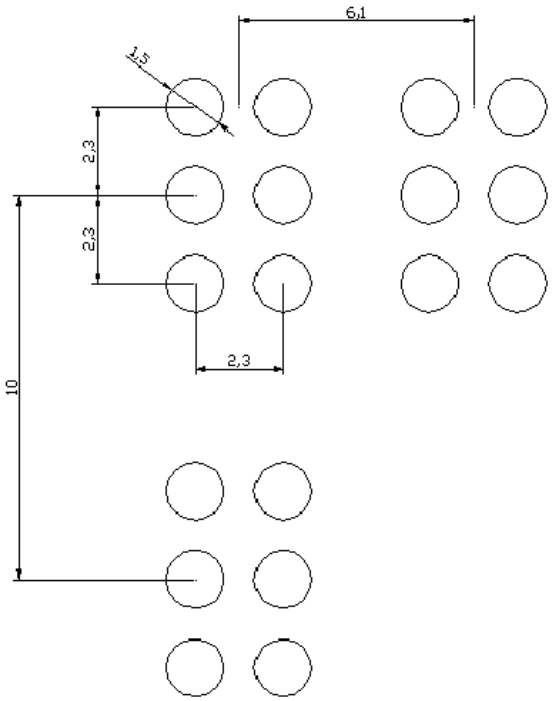
Česká norma Braillova písma je zobrazena v příloze B. Znak, který nemá žádný bod, má význam mezery.

[1] Znaková sada *American Standard Code for Information Interchange*.

2.4 Definice rozměrů

Standardy pro velikosti bodů, vzdálenosti mezi nimi a vzdálenosti mezi znaky jsou zobrazeny na obrázku 2.2.

Základním a společným pravidlem všech norem je, že vzdálenost dvou bodů ve sloupci znaku je vždy stejná jako jejich vzdálenost v řádce daného znaku. Braillův znak se tedy skládá ze dvou čtverců pod sebou.



Obrázek 2.2: Česká norma definice rozměrů

2.5 Nástroje

Braillovo písmo je možné ručně psát pomocí speciální destičky s perem, kterým se zezadu (zrcadlově) přes dírky v destičce vytváří v papíru perforace (tzv. pražská tabulka). K rychlejšímu psaní se užívá Pichtův psací stroj či speciální počítačové tiskárny.

Jako nástroj pro práci s počítačem slouží nevidomým hmatový displej (též braillský řádek), což je zařízení, které převádí část jednoho řádku textu (obvykle z počítačové obrazovky) do hmatové podoby. [18]

2.6 Moonovo písmo

Moonovo písmo je méně univerzální než písmo Braillovo. Používá se jen v anglofonním světě, bylo vytvořeno pro latinku. Někteří jeho zatvrzelí příznivci zastávají názor, že je jednodušší než Braillovo písmo. Faktem je, že ho využívají hlavně lidé, kteří ztratili zrak až v dospělosti, a proto již znají tvary písmen.

Vytvořil ho Dr. William Moon, který v jednadvaceti letech po zápalu plic přišel o zrak. Později se stal učitelem zrakově postižených dětí. Právě díky tomu zpozoroval, že dětem

Obrázek 2.3: Abeceda Moonova písma

dělá velké problémy číst z reliéfních písmen (trojrozměrné verze písmen) a vymyslel vlastní metodu.

Systém Moonova písmo je založený na devíti základních znacích, které mají různý význam podle naklonění. Znaky jsou kvůli čitelnosti větší, což je zároveň jeho hlavní nevýhodou. Výsledný text je vzhledem k původnímu textu rozsáhlý a například obtížně přenosný.

Toto písmo je rovněž vhodné pro nevidomé s menší hmatovou citlivostí a pro děti s dalším fyzickým postižením nebo poruchami učení.

Znaky Moonova písma jsou zobrazeny na obrázku 2.3. [9]

Kapitola 3

Skenování dokumentů

Před počítačovým zpracováním je třeba dokument nějakým způsobem digitalizovat. K obrazové digitalizaci je nejvhodnější specializovaný nástroj: skener.

Skenery můžeme rozdělit do několika typů podle způsobu zpracování dokumentů, možného použití nebo pořizovacích nákladů a s nimi spojené dostupnosti. Pro skenování Braillova písma stačí následující nejzákladnější rozdělení:

- 3D skenery. Ke skenování se používá laserový paprsek odrážený od objektu. Získává ucelené informace o objektu a všech jeho rozměrech: v případě Braillova písma i informace o výšce či hloubce bodu. Tato technologie je ovšem nákladná a není zcela běžně dostupná (často jen ve specializovaných firmách). [13]

- Bubnové skenery (válcové, rotační). Předloha se připíná na skleněný válec, nebo upíná do rámečku (tzv. virtuální buben). Nelze snímat neupínatelné předlohy (knihy, předměty) a často ani neprůsvitné předlohy. Tyto skenery jsou prestižní, kvalitní (5 000 - 15 000 ppi) a extrémně drahé. Používají se převážně u fotografických filmů - negativů a diapozitivů.

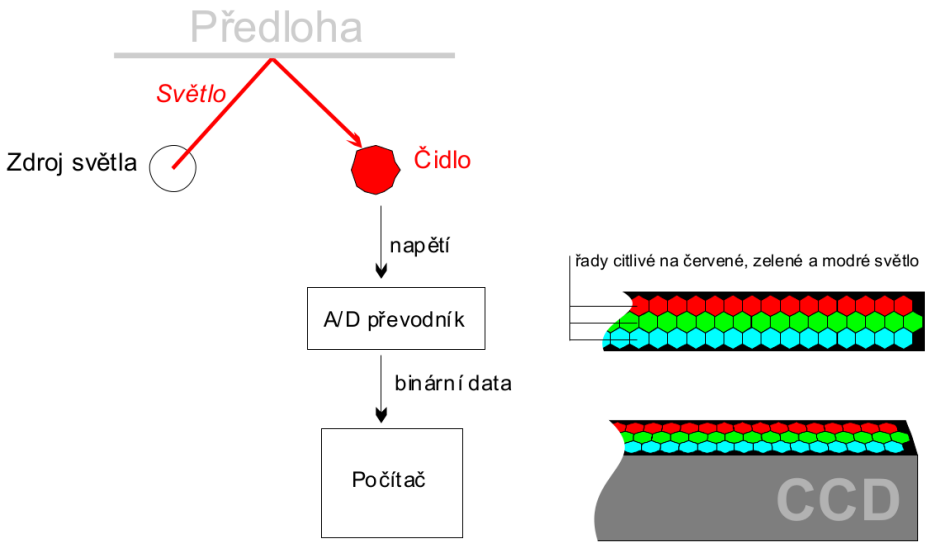
- Plošné skenery. Jsou levnější než předchozí typy, méně kvalitní než bubnové skenery, ale významně rychlejší. Jsou nejrozšířenější a nejdostupnější.

V případě Braillova písma by bylo nejvhodnější používat 3D skenery, ale z hlediska dostupnosti a finanční náročnosti budeme dále uvažovat jen skenování na nejběžnějších, plošných skenerech, které lze pro skenování Braillova písma také použít, jak bude blíže vysvětleno v kapitole 3.3.

3.1 Obecná funkce skenerů

Skenery běžně pracují na podobném principu (viz obrázek 3.1a). Předloha je ozářena zdrojem světla, to se od ní odráží do fotosenzitivních prvků čidla, kde vlivem světla vzniká napětí různé velikosti. To je dále převedeno analogově-digitálním převodníkem na binární informaci, kterou počítač dokáže zpracovat.

Pokud je skenování barevné, liší se konstrukce pouze čidla, které obsahuje tři složky citlivé na různé barvy (obvykle, dle RGB, červená, modrá a zelená).

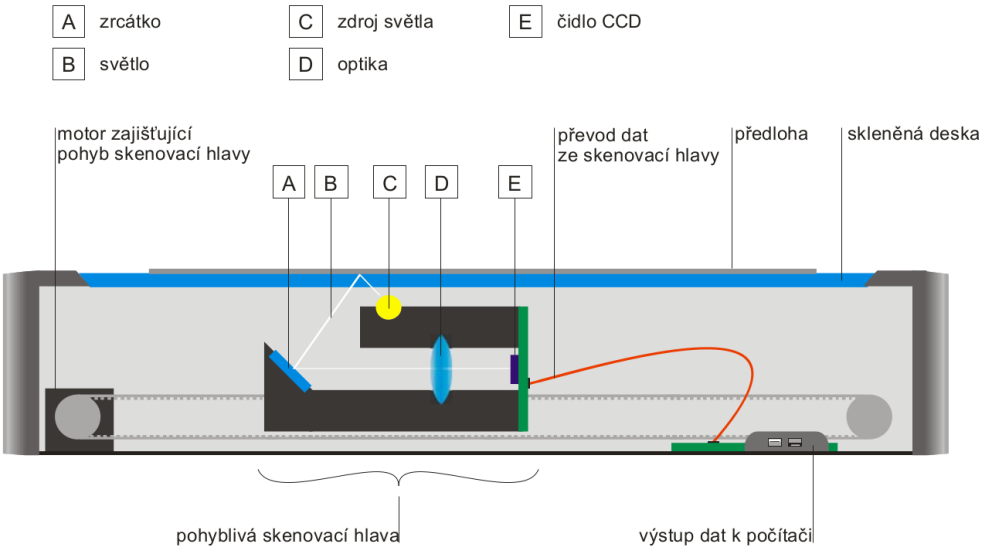


(a) Obecná funkce skeneru (b) CCD čip plošného skeneru

Obrázek 3.1: Obecný sknener a CCD čip

3.2 Plošné skenery

Plošné skenery fungují dle obecného principu, mají skleněnou desku, na kterou se pokládá předloha motivem dolů. Skener snímá zdola pomocí pohybové snímací hlavy, která obsahuje zdroj světla s čidlem. U některých skenerů se nepohybuje snímací hlava, ale předloha. Díky tomuto principu je možné skenovat více dokumentů za sebou bez interakce obsluhy. Některé skenery umožňují i oboustranné skenování dokumentů, ty mají z každé strany předlohy jednu snímací hlavu, což je ovšem finančně náročné. (Dokumenty v Braillově písmu lze oboustranně skenovat na jednostranném skeneru, viz kapitola 3.3.3).



Obrázek 3.2: Obvyklá konstrukce plošného skeneru

Nejčastěji bývá jako čidlo užívaný čip CCD, nebo CIS. Konstrukce CCD čipu ve skenerech je zobrazena na obrázku 3.1b.

Skener současně snímá jeden řádek předlohy a zároveň ho ukládá jako jeden řádek pixelů rastrového obrazu. Obvyklá konstrukce plošného skeneru je na obrázku 3.2. Na synchronizaci krokového motoru a taktu odebírání dat z CCD čipu jsou kladeny velké nároky. [12]

3.3 Braillovo písmo

Z rozdělení skenerů na začátku kapitoly 3 plyne, že nejběžnější a nejdostupnější jsou plošné skenery, které slouží ke skenování dvourozměrných dokumentů. Body Braillova písma jsou oproti běžným dokumentům trojrozměrné a neměly by být těmito skenery naskenovatelné.

3.3.1 Skenování bodů

Plošný skener pracuje na principu odrazu světla (kapitola 3.2). Pokud mu předložíme trojrozměrný bod, bude se od něj světlo odrážet jinak, což významně ovlivňuje intenzitu světla snímaného čidlem. Pro jednostranné dokumenty v Braillově písmu otočené vystouplými body směrem ke skenovací hlavě můžeme vyvodit několik závěrů:

- Pozadí naskenovaného dokumentu nebude bílé (světlé) jako u dvourozměrných dokumentů. Vzdálenost papíru a skleněné plochy je větší, proto světlo dopadá na čidlo pod jiným úhlem a s menší intenzitou. Pozadí je proto tmavší, více šedé.
- Vystouplé body ležící přímo na skleněné ploše nad skenovací hlavou budou nejsvětlejší (nejbělejší), jsou nejblíže zdroji světla; protože jsou zakulacené, určitá část bodu odráží do čidla více světla.
- Zdroj světla ozařuje dokument pod určitým úhlem (kvůli odrazu, obrázek 3.1a); za trojrozměrnými body budou vznikat drobné stíny tmavší než pozadí.

3.3.2 Skenování prohlubní

Pokud předložíme jednostranný dokument otočený vystouplými body směrem od skenovací hlavy, platí následující závěry:

- Situace s pozadím dokumentu je totožná jako v předchozím případě.
- Prohlubně bodu jsou dále od zdroje světla, snímané světlo na čidle má proto menší intenzitu.
- Nejlépe se světlo odráží od míst uvnitř prohlubně bodu, kde bude nejsvětlejší místo.
- Nejhůře se světlo odráží od míst na vnitřní straně prohlubně bodu směrem ke zdroji světla. Kvůli odrazu světlo dopadne na čidlo s nejmenší intenzitou a i zde vznikne menší tmavý stín.

Přestože je plošný skener dvourozměrný, lze na něm díky jeho předchozím uvedeným vlastnostem digitalizovat dokumenty v Braillově písmu bez jakékoliv úpravy skeneru či dokumentu. Vylepšení je možné použitím skenovací fólie, viz kapitola 3.6.

3.3.3 Oboustranné dokumenty

Dokumenty v Braillově písmu jsou často i oboustranné, body jsou na nich vyražené z obou stran papíru. Spojují principy z předchozích dvou kapitol 3.3.1 a 3.3.2, díky kterým je možné jednostranným skenerem skenovat oboustranné dokumenty bez ztráty informací.

Počet stran v Braillově písmu je přibližně třikrát až čtyřikrát větší než u písma tištěného (při běžném formátování). Oboustranné texty tudíž značně šetří množství papíru. Používají se zvláště u knih pro nevidomé, ale i u běžných brožurek.

Text z druhé strany papíru je vůči textu z první strany posunut do mezer mezi body, aby se vzájemně neovlivňovaly.

3.4 Přesvícený obraz

U multifnuknčních kvalitnějších skenerů (častěji skříňových než stolních) je někdy výsledný digitalizovaný dokument značně přesvícený, ztrácí se informace o otočení bodu. Dekódováním na těchto skenerech se budeme zabývat v kapitole 4.8. Pokud nebude výslovně uvedeno jinak, budeme uvažovat nepřesvícené dokumenty, beze ztráty informací o otočení bodu.

3.5 Použité skenery

Skenery s vhodnými ověřenými vlastnostmi pro skenování Braillova písma:

- Canon PIXMA MG5300
- Canon Canoscan 5600F
- HP Scanjet G4050

Teoreticky lze použít většinu stolních kancelářských skenerů. (Pro ostatní lze aplikovat metody pro přesvětlené dokument 4.8.)

3.6 Použití fólie při skenování

Pro zlepšení skenování je možné použít průhlednou fólii, kterou pokládáme mezi skenovací plochu a skenovaný dokument. Fólie by měla zlepšit zpětný odraz světla od bodů bez negativního vlivu na skenování. Je požadována zvláštně při použtí programu OBR [16], kde má funkci kalibračního filtru. Pro běžného uživatele je toto použití nedostupné, proto ho dále nebudeme brát v potaz.

3.7 Chyby skenování

Při skenování dokumentů může nastat několik komplikací:

- Předloha se může nedostatečně přimykat ke skleněné ploše, světlo se odráží odlišně a vznikají tím tmavší místa. To je časté při skenování svázaných knih, nebo dokumentů větších, než je skenovací plocha skeneru, je předlohu nutné skenovat po částech. Tuto situaci řeší adaptivní thresholding (kapitola 4.3.1).

- Dokument při skenování může být špatně otočen. Otočení na ruby nemá vliv, pouze budou dekódované texty ze stran dokumentu v opačném pořadí. Otočením o 180° se zabýváme v kapitole 4.9. Tomuto se nelze vyhnout vyhnout při skenování jednostranných knih ve vazbě, kde by kvůli hřbetu knihy bylo nutno knihu otočit, aby bylo možné zavřít dvířka skeneru.

Kapitola 4

Návrh detekce

Detekce dokumentů v Braillově písmu se skládá z několika bodů:

1. Převod digitalizovaného obrazu na stupně šedi.
2. Předpřipravení obrazových dat (filtry, vyvážení intenzity šedi).
3. Thresholding.
4. Doupravení binárního, nebo ternárního obrazu.
5. Rotace obrazu.
6. Dekódování textu.

Pro detekci jsou nejvhodnější nepřesvícené (kapitola 3.4) jednobarevné, resp. bílé papíry s jednostranným nebo oboustranným textem. U potištěných papírů by mohly tmavší části textu splývat se stíny bodů Braillova písma. Kombinace Braillova písma a tištěného písma není častá, používá se například na obalech léků.

4.1 Šedotónový obraz

Pro detekci Braillova písma stačí mít pouze šedotónový obraz. Jak plyne z kapitoly 3.3, stěžejní jsou jen stíny bodů v naskenovaném obrazu dokumentu, ne barva pozadí papíru. Jinak řečeno detekce se řídí intenzitou odraženého světla (kapitola 3.2). Proto je jako první krok potřeba převést barevnou informaci na informace o intenzitě jasu.

Jak bylo uvedeno v kapitole 3.1, je možné skenovat dokumenty rovnou šedotónově, takové skenování je výrazně rychlejší oproti barevnému.

4.1.1 Konverze na šedotónový obraz

Jedná se o funkci, u které je na vstupu barevná hodnota pixelu a na výstupu hodnota intenzity šedi. Pokud bereme v úvahu reprezentaci barvy v hodnotách RGB, na vstupu máme tři hodnoty pro každou barevnou složku (červená, zelená, modrá), a na výstupu hodnotu jednu. Jednotlivé hodnoty jsou v rozsahu od 0 do 255. Pro barevnou informaci to znamená více než 16 miliónů možných kombinací. [6]

Nejjednodušší konverzí je průměr tří barevných složek, jak je popsáno v rovnici 4.1.

$$I = (r + g + b)/3 \tag{4.1}$$

Další možností je použití tzv. empirického vztahu barev, který je odvozen od schopnosti lidského oka vnímat jednotlivé barevné složky. Tento vztah je popsán rovnicí 4.2. [5]

$$I = 0.299 * R + 0.587 * G + 0.114 * B \tag{4.2}$$

Je možné používat i metodu vyvinutou pro HDTV (digitální televize s vysokým rozlišením), která je popsána rovnicí 4.3.

$$I = 0.216 * R + 0.7152 * G + 0.0722 * B \tag{4.3}$$

Pro detekci Braillova písma je nejvhodnější použít první rovnici, protože barva papíru dokumentů bývá bílá a ideálně již po naskenování jsou všechny tři složky barev jednotlivých pixelů stejné.

Je možné použít i jinou metodu, protože následný thresholding (kapitola 4.3) eliminuje rozdíly, kterými se jednotlivé metody od sebe liší.

4.2 Preparing

Dále je možné obrazová data dokumentu předpřipravit. Tyto metody v kombinaci s metodami některých dalších kroků mohou přinášet kvalitnější výsledky. Lze na nich také demonstrovat možnosti jednotlivých kroků dekódování, ale při obecném automatickém použití s kvalitním obrazovým vstupem nejsou nutné.

4.2.1 Gaussovo rozostření

Slouží k rozmazání obrazu. Tím se částečně odstraňuje šum, ale zároveň se takto mění rozměry a intenzita bodů. Rovněž může mít za následek slití sousedících stínů stejné barvy.

Principem je pro každý pixel obrazu spočítat novou hodnotu intenzity na základě sousedních pixelů. Vychází z rovnice Gaussovy funkce 4.4.

$$f(x) = ae^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.4}$$

Počet sousedních pixelů je třeba zvolit. Lze zvolit i parametr σ (standardní odchylku), který určuje šířku funkce. Pokud není nastaven, spočítá se automaticky dle rovnice 4.5; kde n je horizontální počet sousedních pixelů v okolí pro horizontální jádro a vertikální počet pro vertikální jádro. [2, str. 109 - 114]

$$\sigma = 0.3(n/2 - 1) + 0.8 \tag{4.5}$$

Tuto metodu lze použít pro normování velikosti a tvaru bodů, pokud není možné vyhnout se slití bodů, kapitola 4.5.2.

4.2.2 Rozložení jasu

Tato metoda změní rozložení intenzity jasu v obrazu na základě vypočteného histogramu. Lze ji efektivně použít pro názornost postupu detekce přesvětlených dokumentů. U oboustranných dokumentů se ztrácí informace o vystouplosti bodu, tedy na kterou stranu papíru je bod vryt (řeší metody z kapitoly 4.8).

(a) Originální obrázek (b) Gaussovo rozostření pro 3x3 okolí (c) Gaussovo rozostření pro 9x9 okolí

Obrázek 4.1: Gaussovo rozostření

$$dst(x,y) = (src(x,y) - min_{new}) * \frac{max_{old}}{max_{new} - min_{new}} \tag{4.6}$$

Pro nové rozložení je potřeba vhodně nastavit parametry min_{new} a max_{new}. Maxima a minima původního rozložení jsou hranice rozsahu možných hodnot intenzity $min_{old} = 0$ a $max_{old} = 255$.

Nové parametry se snadno určují na základě histogramu - tím se budeme detailněji zabývat hlavně v kapitole 4.3.3 v souvislosti s thresholdingem). Parametr max_{new} se určí pomocí nejčastější hodnoty intenzity celého obrazu (značí pozadí). Od té je vhodné odečíst určitou toleranci, dobré výsledky jsou při odečtení vhodné konstanty (například deset). Parametr min_{new} je třeba posunout do bodu křivky, v němž začíná značně růst. Je možné například zvolit první nejmenší hodnotu intenzity, při které je počet pixelů, které ji nabývají, větší, něž vhodně zvolená konstanta, například opět číslo deset.

Výstupem této metody je obraz, který má přibližně standardní světlé pozadí jako přesvětlené dokumenty, viz obrázek 4.2.

(a) Originální obrázek (b) Nové rozložení jasu

Obrázek 4.2: Přerozložení jasu

4.3 Thresholding

Thresholding - česky prahování - je nejjednodušší metodou segmetnace obrazu, slouží k rozdělení digitalizovaného obrazu na oblasti se společnými vlastnostmi. Aplikuje se častěji na šedotónový obraz, u barevného obrazu je tato metoda složitější.

Nejčastěji používaný thresholding s jedním globálním prahem znázorňuje rovnice 4.7.[2, str. 135 - 138]

$$dst(x,y) = \begin{cases} 255 & if\ src(x,y) > T \\ 0 & if\ src(x,y) \leq T \end{cases} \tag{4.7}$$

Při detekci Braillova písma (viz kapitola 3.4) je potřeba rozlišit dvě skupiny oblastí, složené ze tří částí:

- Pozadí (vše, co není bod, jakkoliv otočen).
- Bod s jeho světlou i tmavší (stín) částí.

U jednostranných dokumentů stačí dvě složky (pozadí, bod), protože všechny body jsou otočeny na jednu stranu. Pro univerzálnost je vhodnější pracovat s jednostrannými dokumenty stejně jako s oboustrannými, aby nebyla nutná interakce uživatele. Dalšími kroky je zjištěna absence bodů z druhé strany.

Pro oboustranné dokumenty nelze použít thresholding s jedním prahem, který vytváří binární obraz, ale thresholding s dvěma prahy, kdy vzniká obraz ternární (třísložkový). To je znázorněno rovnicí 4.8 pro dva globální prahy L a H.

$$dst(x,y) = \begin{cases} 255 & if\ src(x,y) > H \\ 127 & otherwise \\ 0 & if\ src(x,y) < L \end{cases} \tag{4.8}$$

Prahy nelze empiricky pevně stanovit kvůli rozličnosti skenerů, barvě a kvalitě papíru. Pro každý dokument je nutné prahy individuálně spočítat.

Pro určení hodnoty prahu existuje více metod s různými výsledky.[4]

4.3.1 Adaptivní thresholding pro jednostranné dokumenty

Tato metoda je vhodná, pokud je obraz přesvětlený - neobsahuje informace o otočení bodu (viz kapitola 3.4), výsledkem je binární obraz. Vychází z obecné rovnice 4.7, kde se dynamický práh určuje jako průměr všech hodnot intenzity obrazu, rovnice 4.9.

$$T = \frac{\sum_{x=0}^{width} \sum_{y=0}^{height} src(x,y)}{width * height} \tag{4.9}$$

Pro adaptivní thersholding platí rovnice 4.10.[2, str. 138 - 141]

$$dst(x,y) = \begin{cases} 255 & if\ src(x,y) > T(x,y) \\ 0 & otherwise \end{cases} \tag{4.10}$$

Dynamický práh $T(x,y)$ se pro jednotlivý pixel počítá jako průměr hodnot intenzity v *n-okolí* pixelu.

Adaptivní thresholding oproti jednoduchému dokáže eliminovat nerovnoměrné nasvícení papíru. Srovnání zobrazuje obrázek 4.3.

(a) Originální obrázek (b) Globální práh (c) Adaptivní práh

Obrázek 4.3: Srovnání thresholdingů

4.3.2 Arabic OBR

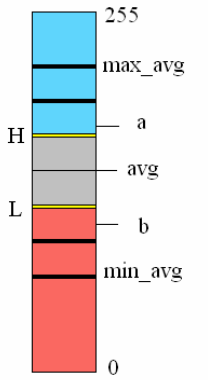
Thresholding pro jednostranné a oboustranné dokumenty, vytvořený pro software OBR[16]. Prahy se počítají z histogramu pomocnými proměnnými a a b, rovnice 4.11.

$$\begin{aligned} a = &\quad mean(max(I) + avg)/2 \\ b = &\quad mean(min(I) + avg)/2 \end{aligned} \tag{4.11}$$

I repreztuje celý obrázek, $max(I)$ a $min(I)$ reprezetnuje největší a nejmenší hodnoty v každém sloupci obrázku I. Prahy se poté vypočtou dle rovnice 4.12.

$$\begin{aligned} L = &\quad avg - (avg - min\_avg)/3 \\ H = &\quad avg + (max\_avg - avg)/3 \end{aligned} \tag{4.12}$$

Parametr min_avg je průměr všech hodnot menších než b a max_avg je průměr všech hodnot větších než a. Thresholding je následně proveden tak, že hodnoty intenzity obrazu větší než parametr H jsou označený za světlé, a hodnoty menší než L jsou tmavé. Rozložení hodnot je viditelné na obrázku 4.4. [1]



Obrázek 4.4: Rozložení hodnot při Arabic OBR threshodlingu

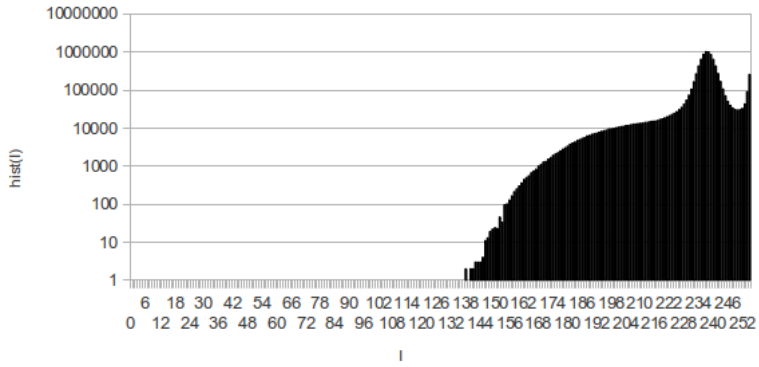
4.3.3 Nejfrekventovanější maximální hodnota

Tato metoda vyplývá z histogramu dokumentů v Braillově písmu, obrázek 4.5.

Logicky lze některým vrcholům přiřadit významy. Nejvyšší vrchol jsou hodnoty pozadí - v celém dokumentu jsou nejčastější. Vrchol při přibližné intenzitě 255 jsou bílé stíny bodů (viz kapitola 3.3.2). Horní práh H se bude nacházet někde mezi nejčastější hodnotou pozadí a nejčastější hodnotou pro světlé body. Ta nemusí být nutně 255, zvláště při použití Gaussova rozostření se vrchol posune více vlevo na nižší hodnoty (bílé stíny se rozostří do okolních pixelů).

Nejjednodušší metodou určení prahu H je položit jej do středu mezi vrchol pozadí a vrchol bílých stínů, viz rovnice 4.13.

$$\begin{aligned} background = &\quad max_{i=0}^{255} hist(i) \\ max\_white \approx &\quad 255 \\ H = &\quad \frac{max\_white + background}{2} \end{aligned} \tag{4.13}$$

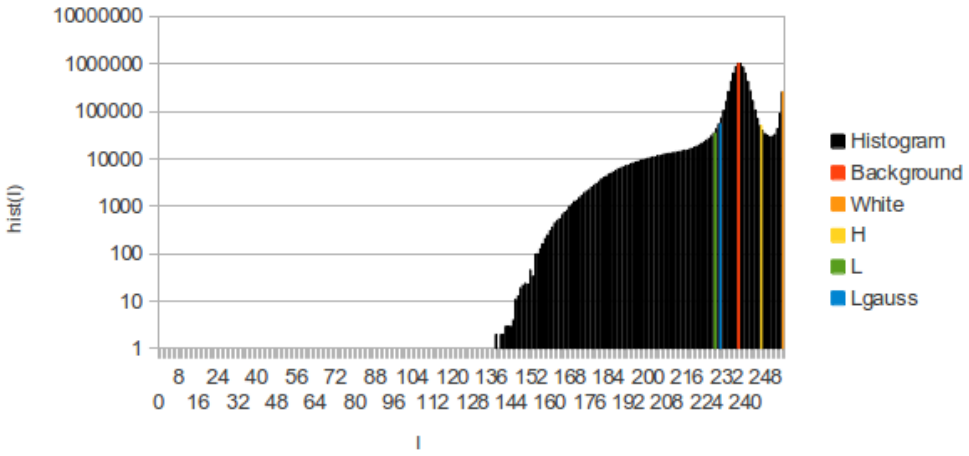


Obrázek 4.5: Histogram běžného dokumentu v Braillově písmu

Druhý práh L analogicky leží na opačné straně vrcholu pozadí. Můžeme předpokládat, že se nachází přibližně ve stejné vzdálenosti, rovnice 4.14.

$$\begin{aligned} diff &= H - background \\ L &= background - diff \end{aligned} \tag{4.14}$$

Lepší výsledky vycházejí při aplikaci Gaussovy křivky, kdy hodnotu L určíme tak, aby měla na druhé straně vrcholu pozadí stejnou intenzitu, jako hodnota H. Rozložení hodnot je vidět na obrázku 4.6.

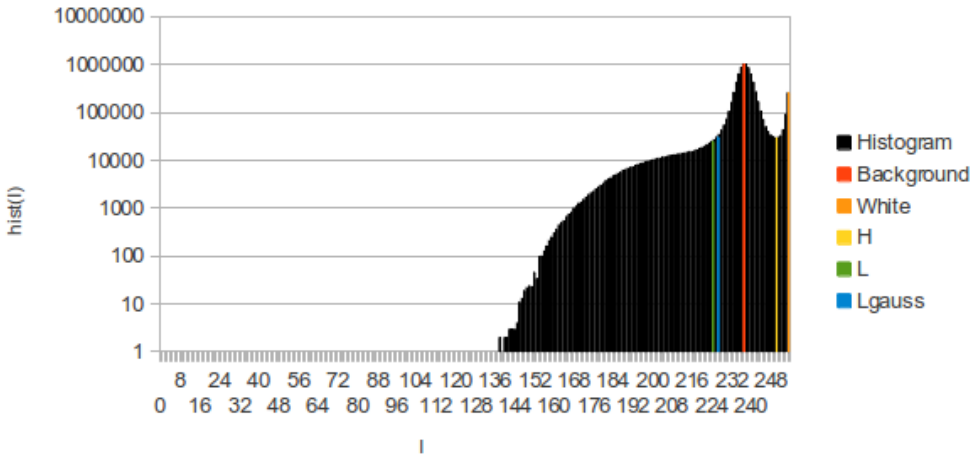


Obrázek 4.6: Rozložení hodnot pro thresholding s maximální hodnotou

4.3.4 Nejfrekventovanější maximální i minimální hodnota

Tato metoda vychází z metody předchozí, liší se v nastavení prahu H, který se nenastaví do poloviny vzdálenosti *background* a *white*, ale do lokálního minima mezi těmito vrcholy.

Práh L se nastavuje stejně, s lepšími výsledky při aplikací Gaussovy křivky. Rozložení hodnot je vidět na obrázku 4.7

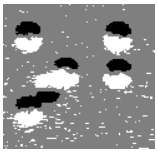


Obrázek 4.7: Rozložení hodnot pro thresholding s maximální i minimální hodnotou

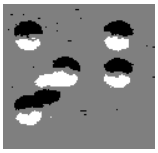
4.3.5 Srovnání thresholdingů

Výsledky předchozích thredhoslindgů jsou na obrázcích 4.8. U výsledků 4.8b a 4.8c je zřejmé, že je potřeba ještě následně odstranit šum obrazu (viz kapitola 4.4). Nejlepší hodnoty má thresholding s maximální a minimální hodnotou 4.8d, na nichž se šum vyskytuje jen u krajů obrazu a je jednodušší jej odstranit.

(a) Originální obrázek



(b) Arabic OBR



(c) S maximální hodnotou

(d) S maximální a minimální hodnotou

Obrázek 4.8: Srovnání thresholdingů

4.4 Reducing

Výstup thresholdingu z předchozího kroku nebude zcela ideální i při použití adaptivních metod. Vlivem znečištění nebo ohybů papíru vznikají černé a bílé shluky bodů, při nekvalitních prazích někdy i celé oblasti; u neadaptivních metod často u krajů. Cílem tohoto kroku je tento šum detekovat a odstranit.

Obecně je nejjednodušší stanovit vlastnosti, které mají bílé a černé stíny bodů, a všechno ostatní označit jako pozadí dokumentu. K tomu je potřeba rozpoznávat tvar stínů, nebo

nějak stanovit obecnou velikost i s tolerancemi.

4.4.1 Velikost bodu

Rozpoznávání dle tvaru bodu není snadné. Jednoduchá forma určení používá výšku a šířku bodu, která se určuje z pozičních extrémů bodu (nejhořejší, nejlevější, atd.). Další možností je použití velikosti plochy bodu. Situaci komplikuje rozdílnost tvaru bílých a černých stínů i vliv otočení bodu (oboustranné dokumenty).

Dále je vhodné znát standardní šířku a výšku bodu i velikost jeho plochy; získáním těchto informací se budeme zabývat v kapitole 4.5.1.

Při redukci dle velikostí bodu existuje několik možností, podle kterých můžeme určovat, zda je daný stín stínem bodu:

- Stanovit poměr, který musí být mezi výškou a šířkou.
- Stanovit toleranci rozměrů bodu: výšky, šířky i obsahu.

Při chybném nastavení se označí stíny bodů za pozadí, proto je potřeba pracovat s dostatečnou tolerancí. Obtížnost tohoto nastavení výrazně snižuje použitelnost metody. Jednoduchou formu můžeme použít při získávání bodu, viz kapitola 4.5.1.

4.4.2 Jednopixelová redukce

Používat tuto metodu současně s Gaussovým rozostřením z kapitoly 4.2.1 nemá smysl. Princip metody je naznačen v algoritmu 1. Metoda zjišťuje, zda pixel je v nějakém směru obklopen pozadím. Pokud ano (návratová hodnota funkce $True$), pixel je označen za šum a nahrazen pozadím. Tato funkce je spuštěná pro každý pixel obrazu.

Algorithm 1 Is the one pixel noise?

```
1:  function isOneNoise(x,y)                    ▷ x a y jsou souřadnice testovaného pixelu
2:      if src(x − 1, y) = bgColor && src(x + 1, y) = bgColor then
3:          return True
4:      end if
5:      if src(x, y − 1) = bgColor && src(x, y − 1) = bgColor then
6:          return True
7:      end if
8:      if src(x − 1, y + 1) = bgColor && src(x + 1, y − 1) = bgColor then
9:          return True
10:     end if
11:     if src(x − 1, y − 1) = bgColor && src(x + 1, y + 1) = bgColor then
12:         return True
13:     end if
14:     return False
15: end function
```

Rozšířením této metody je její opakování, dokud je přítomný šum, který lze redukovat. Metoda zakulacuje body (odstraňuje jednopixelové okraje).

4.4.3 Redukce obklopením

Metoda podobná té předchozí. Pro každý pixel se testuje, zda všechny jeho okolní body (osmiokolí) je stejné barvy, viz algoritmus 2.

Algorithm 2 Is the neighboor noise?

```
1:  function isNbNoise(x,y)                    ▷ x a y jsou souřadnice testovaného pixelu
2:      color ← src(x, y)
3:      if color = bgColor then
4:          return True
5:      end if
6:      if src(x − 1, y) ≠ color || src(x + 1, y) ≠ color then
7:          return True
8:      end if
9:      if src(x, y − 1) ≠ color || src(x, y − 1) ≠ color then
10:         return True
11:     end if
12:     if src(x − 1, y + 1) ≠ color || src(x + 1, y − 1) ≠ color then
13:         return True
14:     end if
15:     if src(x − 1, y − 1) ≠ color || src(x + 1, y + 1) ≠ color then
16:         return True
17:     end if
18:     return False
19: end function
```

Oproti jednopixelové redukci zásadně mění také tvar bodů. Nejenže je zakulacuje, ale i podstatně zmenšuje. To může být žádoucí při použití Gaussova rozostření.

4.5 Získání bodů

U dalších metod lze většinou používat dvě možnosti vstupních dat. První je standardní obraz, kdy se při jakékoliv metodě rozpoznávají body přímo v obrazových datech a výsledky se ukládají zpět do nich. Nebo lze nejdříve body z obrazu rozpoznat a uložit do zvláštních struktur. To má několik výhod, například zrychlení programu, kdy se nemusí znovu procházet všechny pixely obrazu.

Pro názornost budeme zatím uvažovat jen jednostranné nepřesvětlené dokumenty za použití thresholdingu s minimální a maximální hodnotou z kapitoly 4.3.4 a následnou redukcí šumu, příklad je na obrázku 4.9a.

Jeden bod se skládá z černého a bílého stínu (kapitola 3.3), pro jednostrannou detekci postačuje jeden typ stínů označit jako bod. Proto je možné rozpoznávat je jak podle bílých, tak podle černých stínů. Celý proces získání bodů probíhá následovně:

1. Procházíme celý obraz pixel po pixelu.
2. Pokud je pixel zvolené barvy (černý, nebo bílý), zjistíme tvar a rozměry celého stínu.
3. Uložíme si informace o něm do zvláštních struktur.
4. Zabezpečníme, aby nebyl znovu nalezen při dalším procházení.

Na obrázku 4.9b je toto rozpoznávání znázorněno. Bílé křížky značí rozpoznané středy černých stínů a černé křížky bílých stínů.

(a) Originální obrázek (b) Detekce stínů

Obrázek 4.9: Detekce černých a bílých stínů

4.5.1 Rozměry a tvar stínu

Při získání rozměru a tvaru stínu máme k dispozici souřadnice jednoho jeho pixelu. Jsou možné dvě jednoduché metody získání: pomocí obsahu stínu, nebo pomocí jeho obvodu.

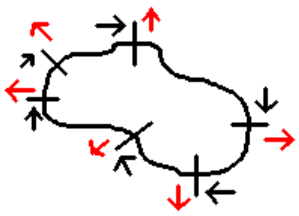
Při získání obsahem se hledají všechny stejně barevné pixely v okolí počátečního pixelu i v jejich okolí se dále vyhledává. Toto procházení může probíhat následujícími způsoby:

- Rekurzivně, kdy se pro každý sousední pixel spouští stejná funkce a počítá se celkový počet pixelů.
- Zásobníkově, kdy se všechny sousední pixely uloží na zásobník a pro vrchní pixel se opět přidají jeho sousední pixely. Zároveň se pixely počítají.

K tomu je potřeba zabezpečit, aby se stejný pixel nepočítal více než jedenkrát.

Rekurzivní metoda má oproti zásobníkové menší nároky na paměť. Nepotřebuje zásobník, ale při rozměrných bodech může být hloubka zanoření i několik set, což výrazně zpomaluje program.

Při získání pomocí obvodu stínu je nutné nejdříve zajistit, aby počáteční pixel byl součástí obvodu stínu, čili na jeho hranici. Pokud není, je třeba v jednom směru tuto hranici najít (případně okraj stránky). Aby funkce postupovala po okraji stínů, je potřeba uchovávat informaci, ze kterého směru jsme k aktuálnímu pixelu přistoupili. Začínáme obvykle u nejlevějšího a nejvyššího pixelu a počáteční směr nastavujeme zleva. Dále nalezneme pixel, který je ještě součástí bodu, a to tak, abychom obvod počítali ve směru hodinových ručiček. První sousední pixel, který je třeba testovat je o 90° posunut od předchozího směru, a dále sousední pixely testujeme ve směru hodinových ručiček, viz červené šipky na obrázku 4.10, černé zobrazují směr, odkud se postupuje. Při dosažení počátečního pixelu je proces ukončen a navrácen počet navštívených pixelů.



Obrázek 4.10: Hledání obvodu bodu

Obsahová metoda je náročnější na paměť a přístupy do obrazových dat, kdy se pixely musí označit jako navštívené. Pokud není vymazání požadované, velmi tím rostou nároky

na paměť, protože se musí vytvářet a mazat kopie obrazových struktur. Obvodová metoda je náročnější na implementaci a kontroly již zpočítaných stínů bodů (pixely v obraze se neoznačují).

Je vhodné určovat i střed bodu. Ten je jednoduché určit z informací o nejvyšší, nejnižší, nejlevější a nejpravější souřadnici stínu.

Pro další metody je třeba zjišťovat také šířku bodu jako rozdíl nejpravější a nejlevější souřadnice.

4.5.2 Slité body

Zejména u oboustranných textů se stíny bodů často slívají dohromady. Při ukládání do zvláštních struktur je nejprve nutné buď tomuto slití nějak zamezit, nebo stíny rozdělit.

K zamezení slití lze použít předchozí krok, thresholding, kdy je dolní práh nastaven na nižší hodnotu a horní na vyšší. Tím se stíny zmenší a slité se od sebe oddělí. Tato metoda má negativní vliv na neslité body, které splynou s pozadím, pokud nejsou dostatečně velké.

Jednodušší a efektivnější metodou je nejdříve zjistit průměrnou šířku všech stínů a delší stíny podle ní rozdělit. To může probíhat tak, že od délky stíny odečítáme průměrnou šířku, dokud je zbytek větší než tento průměr. Tím ale můžou na koncích původních slitných stínů vznikat velmi malé zbytky (šířka je rovna zbytku po dělení šířky průměrem). Řešením je zavést toleranci k průměrné šířce, čili neodečítat celou průměrnou šířku, ale šířku zmenšenou o vlastní procentuální čast.

Nebo určit rozdělení dle rovnic 4.15.

$$\begin{aligned} pocetDilcichStinu &= floor\left(\frac{sirkaStinu}{prumernaSirka}\right) \\ sirkaDilcihoStinu &= \frac{sirkaStinu}{pocetDilcichStinu} \end{aligned} \tag{4.15}$$

Při tvoření dílčího stínu nastavíme jeho nejlevější souřadnici na nejlevější souřadnici slitého stínu, nejpravější vypočteme jako součet nejlevějšího a šířky dílčího stínu. Střed potom určíme jako polovinu mezi těmito souřadnicemi. Je vhodné vypočítat rozměry a tvar pro dílčí stín některou výše uvednou metodou, kde navíc určíme levou a pravou hranici dílčího stínu.

4.5.3 Redukce

Při získání bodů lze uplatnit některé závěry z kapitoly 4.4.1 o redukci stínů dle velikosti. Je vhodné jako normu použít průměrnou nebo nejčastější šířku bodu. Lze odstranit případný šum nebo černé oblasti u okrajů stránek, které by se nepodařilo odfiltrovat předchozími metodami, a to tak, že odstraníme všechny body které:

1. Mají šířku o polovinu menší než je zvolená norma šířky.
2. Mají šířku o čtyřikrát větší než je zvolená norma šířky.
3. Mají výšku stínu dvakrát větší než vlastní šířku.

4.5.4 Rozpoznání oboustranných bodů

U oboustranných dokumentů je vhodné ukládat body z jedné a druhé strany do oddělených struktur, protože jejich dekódování probíhá odděleně. K tomu je nejdříve nutné rozpoznat,

na jakou stranu je bod otočen. To není možné zjistit jen na základě jednoho druhu stínu. Je proto potřeba najít vždy dvojici stínů (černý a bílý) patřící k jednomu bodu Braillova znaku. Je vhodné nejprve zvolit a najít jeden typ stínu a k tomu hledat druhý typ patřící stejnému bodu.

Uvažujme situaci, kdy jsme již získali všechny stíny jednoho typu, slité stíny jsme rozdělili a uložili je do struktur. Hledání druhého stínu (opačné barvy, nad nebo pod prvním stínem) a detekce, na které straně je bod znaku vyražen, bude probíhat následovně:

1. Procházíme všechny stíny uložené ve zvláštní struktuře. Máme o nich následující informace: šířku, výšku, poziční maxima, střed.

2. Pro každý bod vyjdeme z jeho středu nahoru a dolů do vhodně nastavené vzdálenosti, pokud nalezeneme pixel odpovídající hledané barvě stínu, aplikujeme některou metod z kapitoly 4.5.1 a zjistíme informace o stínu. Je možné použít dodatečnou redukci velikosti a tvaru stínu z předchozí kapitoly 4.4.

3. Pokud nenalezeme ani jeden stín, buď je špatně nastavená vzdálenost hledání, nebo jsme narazili na šum. V takovém případě tento šum ignorujeme a pokračujeme na další bod.

4. Pokud nalezeme stín jen na jedné straně, rozpoznáme podle něj, na které straně byl bod vyryt, a pokračujeme na další bod.

5. Pokud nalezneme stín na obou stranách, vypočteme, který stín je blíž, a podle něj se budeme rozhodovat, na které straně byl bod vryt.

Rozpoznání příslušnosti ke straně papíru vyplývá z kapitoly 3.3. Bod znaku na straně přivrácené ke skenovací hlavě bude složen z černého stínu, který bude kousek nad bílým stínem. U bodů na zadní straně dokumentu je situace opačná: bílý stín se nachází nad černým.

Rozpoznání v závislosti na zvolené barvě prvního stínu popisuje tabulka 4.1.

Barva prvního stínu	**Druhý stín**	
	Dole	Nahoře
Černá	Přední strana	Zadní strana
Bílá	Zadní strana	Přední strana

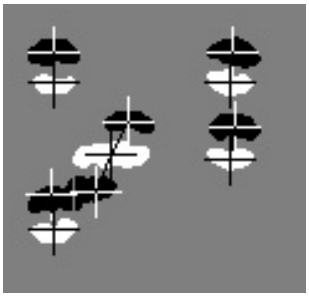
Tabulka 4.1: Rozpoznání strany bodu v závislosti na stínech

Příklad rozpoznávání podle černých stínů je na obrázku 4.11a. Původní pozice bodu je logicky umístěna mezi černý a bílý stín. Je možné ji určit:

- Jako polovinu vzdálenosti mezi středy černého a bílého stínu.

- Jako polovinu vzdálenosti mezi nejspodnější souřadnicí horního stínu a nejvrchnější souřadnicí dolního stínu.

Příklad pozice středů původních bodů je na obrázku 4.11b. Černé křížky jsou pro body z přední strany, bílé pro zadní stranu.



(a) Rozpoznání oboustranných bodů

(b) Středy výsledných bodů

Obrázek 4.11: Detekce strany dokumentu

4.5.5 Překreslení

Další možností získání bodů Braillova písma z obrazu je nejdříve překreslení jeho stínů. Jedná se v podstatě o variantu Gaussova rozostření. Překreslování může být:

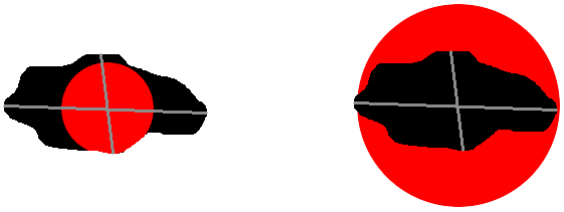
- Kruhové, kdy je potřeba předem rozdělit slité stíny na dílčí. Následně stín překreslíme na kruhový bod, kdy střed bude totožný se středem stínu. Poloměr bodu může být:
  - Nejmenší vzdálenost od středu k nějakému hraničnímu bodu (nejlevější, nejhořejší, atp.), obrázek 4.12a.
  - Šířka původního bodu, obrázek 4.12b.
- Obdélníkové, kdy stín překreslíme na obdélník tak, aby hranice byly maxima stínu, obrázek 4.12c.
- Elipsové, kdy stín překreslíme na elipsu tak, aby hranice byly maxima stínu, obrázek 4.12d. Natočení elipsy může být:
  - Vodorovné s osou x.
  - Odpovídající podélné ose původního stínu.

U obrázků 4.12 šedé čáry určují střed stínu a vzdálenost ke krajním souřadnicím. Černá barva znázorňuje původní stín a červená překreslený stín.

Příklad překreslení je vidět na obrázku 4.13. Pro detekci strany bodu je často kvůli překrývání vhodné používat spíš středy bodů než jejich souřadnicová maxima.

4.6 Rotace

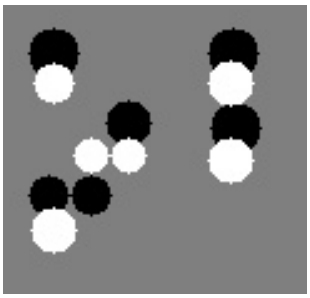
Skenovaný dokument může být při digitalizaci mírně pootočen. Tím se vzájemná poloha páru stínů patřící jednomu bodu vůči souřadnicím neposune, ale samotné znaky a řádky vůči sobě ano. Toto otočení lze při malém úhlu eliminovat při samotném dekódování (kapitola 4.7.4), nebo při větším úhlu ve zvláštním kroku. Existuje několik možností, které se rozdělují podle toho, jestli pracují přímo s obrazovými daty (výpočet pomocí histogramu a pomocí první řady), nebo s již vyexportovanými souřadnicemi bodů (výpočet z bodů). Obecnými metodami nelze správně detekovat obraz otočený o více jak 45° - detekovaný úhel by více inklinoval k správnému otočení dokumentu o 180°, než k vyrovnání na nulové



(a) Kruhové překreslení (b) Kruhové šířkové překreslení

(c) Obdelníkové překreslení (d) Elipsové překreslení

Obrázek 4.12: Překreslení stínu



Obrázek 4.13: Kruhové překreslení dokumentu

otočení. Otočení o takové úhly není potřeba příliš uvažovat, lze předpokládat, že uživatel vloží dokument do skeneru přibližně rovně v rozmezí detekovatelného úhlu. Možnými korekcemi se dále budeme zabývat v kapitole 4.9.

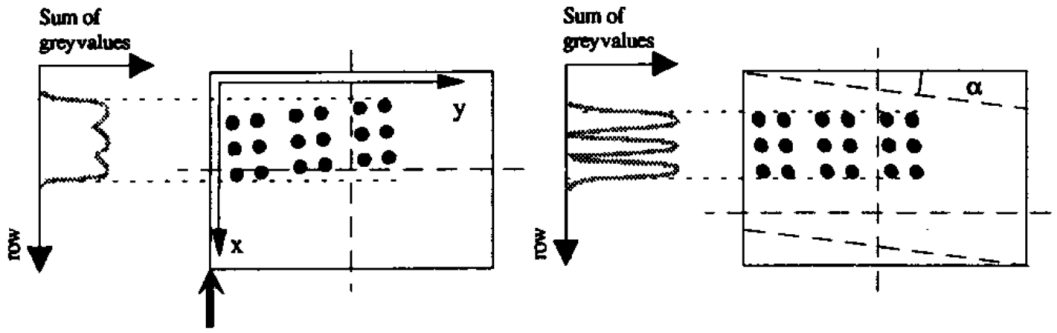
4.6.1 Rotace pomocí histogramu

Tato metoda je založena na řádkovém histogramu, resp. na součtu intenzit pixelů v jednom řádku obrazu, případně na jejich průměrné hodnotě. Lze ji provádět i před thresholdingem, s nevýhodou posunu stínů vůči sobě, což komplikuje jejich spárování a detekci původního bodu.

Pro každý řádek obrazu je spočítán buď součet intenzit, nebo jejich průměr. Uvažujme nejprve situaci, kdy máme bílé (přesvětlené) pozadí jednostranného dokumentu, a body jsou reprezentovány pouze černými stíny. Pro řádky, které neobsahují žádný stín, bude součet i průměr intenzit přibližně nulový; pro řádky obsahující body budou mít určitou hodnotu v závislosti na počtu bodů. Při správném otočení bude část řádků nabývat výrazně nenulové hodnoty (body) a část nulové (pozadí), jak je vidět na obrázku 4.14. Tímto postupelm lze otáčet obrazová data, dokud nedocílíme co nejlepšího oddělení. [7]

Na začátku je potřeba vhodně zvolit krok (úhel), o kolik obraz postupně otáčíme. Pokud se výsledky začnou zhoršovat, je vhodné se o jeden krok vrátit a postupovat stejně s menšími

kroky.



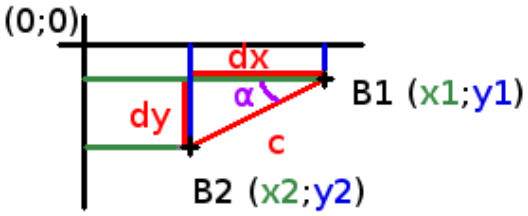
Obrázek 4.14: Příklad rotace pomocí histogramu

Komplikovanější situace nastává při použití této metody před thresholdingem na nepřesvětlený (jednostranný nebo oboustranný) dokument, u nějž je potřeba spočítat limitní hranice hodnot, jedná se ovšem o podobnou metodu, jako je thresholding.

Pro oboustranný nepřesvětlený dokument po thresholdingu je vhodné nerozlišovat bílé a černé stíny (právě kvůli nutnosti výše uvedených limitních hranic hodnot).

4.6.2 Výpočet pomocí načtení bodů

Další možností detekce rotace je načíst několik bodů v jedné řadě a zjistit jejich vzájemnou polohu, neboli úhel, který jejich spojnice svírají s osou x. Bod je vhodné zvolit jako stejnou část (nejlevější souřadnice, střed atp.) stínu - případně rekonstrukovaný bod z páru stínů; nejlepších výsledků je dosahováno při zvolení výpočtu dle středu. Obecně se úhel dvou bodů (B1 a B2 na obrázku 4.15) vypočte dle Pythagorovy věty, rovnice 4.16.



Obrázek 4.15: Výpočet úhlu dvou bodů

$$\alpha = \tan \frac{y_2 - y_1}{x_1 - x_2} \tag{4.16}$$

Je možné načíst jen jednu řadu bodů (například první), nebo všechny body v dokumentu (to koresponduje s kapitolou 4.5). Výpočet úhlu otočení dokumentu může být potom určen jako:

- Úhel mezi dvěma libovolnými body, například mezi krajními. Proto lze teoreticky načíst jen dva body.
- Průměr úhlů všech vedle sebe ležících bodů, nebo všech kombinací bodů.
- Nejčastější úhel mezi dvěma vedle sebe ležícími body, nebo všemi jejich kombinacemi.

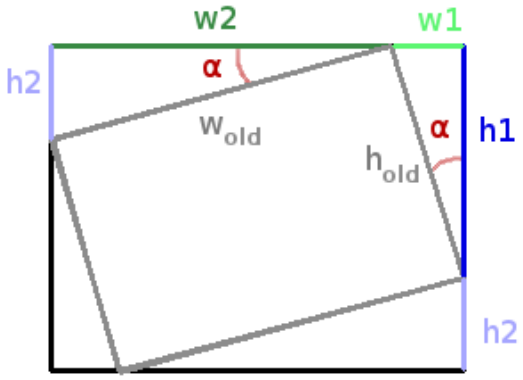
Je potřeba zabezpečit, aby výpočet úhlu probíhal pro stejnou (reálnou) řadou bodů (nikoliv řádku znaků). To lze zajistit několika způsoby:

- Kontrolovat, zda rozdíl souřadnic (dvou bodů) ve směru osy y není větší než ve směru osy x - výsledný úhel by byl větší než 45°.
- Používat výpočet pomocí nejčastější hodnoty úhlu - při otočení přibližně o 45° nemusí fungovat dostatečně přesně.
- U výpočtu nejčastější hodnoty je možné odebrat body, které mají úhel větší než nastavenou toleranci, a výpočet opakovat, případně spočítat průměr úhlů.

Je potřeba dát pozor na znaménka úhlů - odvíjí se od rovnice 4.16, kdy při otočení proti směru hodinových ručiček (obrázek 4.15) bude úhel kladný - v souladu s matematickými zvyklostmi. Dokument se následně otočí o vypočtený úhel v opačném směru, aby se vyrovnal.

4.6.3 Rozměry otočeného obrazu

Implementací otočení obrazových dat se budeme zabývat v kapitole 5.4. Při něm je potřeba i uvažovat zvětšení výsledného obrazu, aby nedošlo ke ztrátě dat. Případně lze ořezat okraje, kde nejsou body, k tomu je ovšem nutná přesná znalost umístění bodů - pokud to víme, je vhodnější otáčet přímo body (následující kapitola 4.6.4). Příklad je vidět na obrázku 4.16. Šedý obdelník znázorňuje původní otočený dokument, černo-barevný obdélník znázorňuje otočený dokument s novými rozměry. Dokument je otočený o úhel α proti směru hodinových ručiček.



Obrázek 4.16: Výpočet rozměrů otočeného obrazu

Výpočet nových rozměrů je znázorněn rovnicí 4.17.

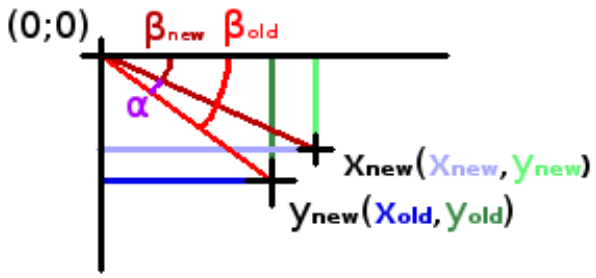
$$\begin{aligned} w_1 &= \cos\alpha * w_{old} \\ w_2 &= \sin\alpha * h_{old} \\ h_1 &= \sin\alpha * h_{old} \\ h_2 &= \cos\alpha * w_{old} \\ w_{new} &= w_1 + w_2 \\ h_{new} &= h_1 + h_2 \end{aligned} \tag{4.17}$$

4.6.4 Výpočet z bodů

Předchozí metody detekce otočení dokumentu pracovaly přímo s obrazovými daty. Tato metoda pracuje s již vyexportovanými souřadnicemi bodů metodami z kapitoly 4.5.

Pro taková data je možné použít některou z metod předchozí kapitoly 4.6.2 pro výpočet úhlu otočení. Protože i dále budeme chtít pracovat s vyexportovanými body je výhodnější přepočítat přímo souřadnice bodu, nikoliv otáčet dokument a znovu načítat body.

Budeme uvažovat situaci z obrázku 4.17, kde již máme vypočtené, že potřebujeme dokument otočit o úhel α, aby byl vyrovnán.



Obrázek 4.17: Rotace souřadnic bodů

Pro takovouto situaci platí rovnice 4.18.

$$\begin{aligned} \beta_{old} &= \arctan \frac{y_{old}}{x_{old}} \\ \beta_{new} &= \beta_{old} - \alpha \\ x_{new} &= \frac{x_{old}}{\cos \beta_{old}} * \cos \beta_{new} \\ y_{new} &= \frac{y_{old}}{\sin \beta_{old}} * \sin \beta_{old} \end{aligned} \tag{4.18}$$

4.7 Dekódování

Nyní máme digitalizovaný dokument rozdělený na černé a bílé stíny a pozadí, šum je odstraněn, dokument správně natočen. Je možné začít se samotnou detekcí a dekódováním textu. Ty probíhají buď přímo s obrazovými daty, nebo s již vyexportovanými souřadnicemi bodů.

Základem obou možností je znalost normy rozměrů Braillova písma, resp. poměrů mezi vzdálenostmi bodů (česká norma byla uvedena na obrázku 2.2) a také minimálně jednoho libovolný známého rozměru. Ten lze určit:

- Pomocí znalosti DPI[1] použitého skeneru - z toho lze vypočítat poměr mezi velikostí reálného bodu a velikostí digitalizovaného bodu (můžeme použít jakýkoliv rozměr normy). Tím získáme počty pixelů pro všechny velikosti a vzdálenosti normy.
- Pomocí průměru bodu - ten lze určit jako průměr nebo nejčastější šířku všech bodů.

[1] *Dots per inch* - počet pixelů na délku jednoho palce (2,54 cm).

- Pomocí vzdálenosti dvou bodů nad sebou v rámci jednoho sloupce znaku. Tu lze určit jako průměrnou, nebo lépe nejčastější nejmenší horizontální vzdálenost všech dvou nad sebou ležících bodů.

Oboustranné dokumenty můžeme za použití některé z dále uvedených metod detekovat dvěma způsoby:

- Rozpoznat obě strany dokumentu najednou, uložit do dvou různých struktur a data z druhé strany po dekódování otočit jednou z následujících možností:
  - Ukládat si při procházení obrazu jen šestici bodů a při otáčení dat z druhé strany postupovat od konce řádku k začátku a sloupce znaku vyměnit, poté dekódovat.
  - Při detekci bodu brát v úvahu otočení a znak dekódovat a ukládat na konec struktury výsledného textu.
- Rozpoznávat nejdříve první stranu (od začátku řádku do konce) a poté druhou stranu (od konce řádku do začátku). Dekódovat znaky lze okamžitě.

4.7.1 Pomocí prvního bodu znaku

S obrazovými daty lze pracovat několika způsoby. Prvním z nich je detekovat postupně znaky pomocí jeho prvního bodu dle následujícího postupu:

- Procházíme celý obraz, při prvním stínu (je lepší se řídit podle jednoho typu stínu) - prvním jeho odpovídajícím pixelu a načtením celého stínu - detekujeme, z které strany byl bod vryt (kapitola 4.5.4), a vypočteme souřadnice původního bodu.
- Pomocí zvolené normy zjistíme, zda v očekávaných místech vzhledem k prvnímu bodu jsou další body. Je potřeba brát v úvahu, že předpokládané souřadnice určují místo reálného bodu, které se často nachází na pozadí mezi dvojicí stínů - proto je vhodné hledat stíny nad a pod daným místem.
- Tím jsme získali souřadnice všech šesti možných bodů a informace, zda se v daném místě bod nachází. Tyto informace je vhodné převést do vnitřní reprezentace znaku Braillova písma a následně převést do latinky.
- Je potřeba zajistit, aby body znaku byly dekódovány jen jednou. To automaticky zajišťuje metoda zjišťování tvaru dle obsahu z kapitoly 4.5.1, nebo můžeme průběžně ukládat souřadnice (například středy) již dekódovaných bodů a s těmi kontrolovat středy nově nalezených bodů.

4.7.2 Pomocí řady znaků

Další možností je načítání dokumentu po řádcích následujícím způsobem:

- Procházíme celý obraz, vždy načteme první tři body z každé řady bodů jednoho Braillova znaku.
- Nalezení znaku provedeme tak, že načteme šest bodů (dva z každé řady), najdeme nejlevější bod a zjistíme (dle normy), zda jsou součástí znaku s nejlevějším bodem a ve kterém sloupci. Tím máme informaci o šestici bodů, kterou můžeme dekódovat do latinky.

- Opět je potřeba zajistit, aby body znaku byly dekódovány jen jednou - viz předchozí kapitola 4.7.1. Body, které nejsou součástí znaku,s je možné uchovat v paměti, aby se nenačítaly znovu, a tím dekódování urychlit.
- Při načítání prvních bodů řad lze provést korekci vzdáleností:
  - Zprůměrováním vzdáleností dvou řad (horní s prostřední a dolní s prostřední) - ve všech normách jsou obě stejně velké.
  - Pokud nalezneme jen dvě řady, je možné třetí dopočítat. Ze vzdálenosti mezi nimi zjistíme, zda chybí prostřední řada. Jinak porovnáváme vzdálenosti k řadám znaků nad a pod právě dekódovanou řadou a podle nich zjistíme, zda chybí řada nahoře, nebo dole. Chybějící řadu dopočítáme.
- Při načítání bodů podle prvního v řadě lze dělat korekci otočení, ale jen po směru hodinových ručiček, a to tak, že hledáme bod v určitém rozsahu hodnot osy y. Při nalezení upravujeme řadu pixelů, na níž budeme hledat dále.

4.7.3 Pomocí vyexportovaných souřadnic

Elegantní řešení detekce a dekódování je pomocí již vyexprotovaných souřadnich z kapitoly 4.5. Tyto souřadnice mohou být uloženy několika způsoby:

- Jako řídké pole, v němž každá buňka odpovídá jednomu pixelu obrazu, dekódování probíhá obdobně jako pomocí prvního bodu znaku (kapitola 4.7.1, nebo pomocí řady znaků 4.7.2).
- Jako dvourozměrné pole, v jehož každé buňce bude uložen nějaký bod - sloupcově pole neodpovídá skutečnosti; při dekódování je potřeba složitě pole procházet, dokud nenarazíme na bod, který je reálně ve stejném sloupci.
- Jako dvourozměrné řídké pole, u nějž vycházíme z předchozí možnosti, ale zarovnáme sloupce pole přidáním prázdných buněk tak, aby ve sloupcích byly body, které jsou i reálně pod sebou. Tím zjistíme všechna místa, kde by se mohly nacházet body znaků.
- Jako jednorozměrné pole (vektor) - ukládání do něj je nejjednodušší, ale vyhledávání znaků je je ještě složitější než u prvního řídkého pole.

První možnost není pro použití příliš vhodná, jedná se v podstatě o složitější variantu dekódování přímo nad obrazovými daty, která není dostatečně efektivní. Zbylé dvě možnosti lze převést na možnost třetí, kterou se dále budeme zabývat nejvíce, protože umožňuje snadnou korekci rotace a chybných bodů.

Jednorozměrné pole (vektor) je potřeba nejdříve převést na dvourozměrné neřídké pole. Toho docílíme tak, že všechny body, které jsou na jedné řádce (jedna hodnota souřadnice y) uložíme jako jednu řádku pole. Na základě znalosti šířky bodu nastavíme toleranci rozmezí souřadnice y, na jejímž základě se rozhodujeme, zda bod do řádky patří.

Při ukládání bodů do řad je vhodné body seřadit podle hodnot souřadnice x kvůli dalším krokům. Toho dosáhneme buď zpětným seřazením, nebo řazením při ukládání.

Dvourozměrné pole je dále potřeba převést na řídké - srovnat body pod sebe - následovně:

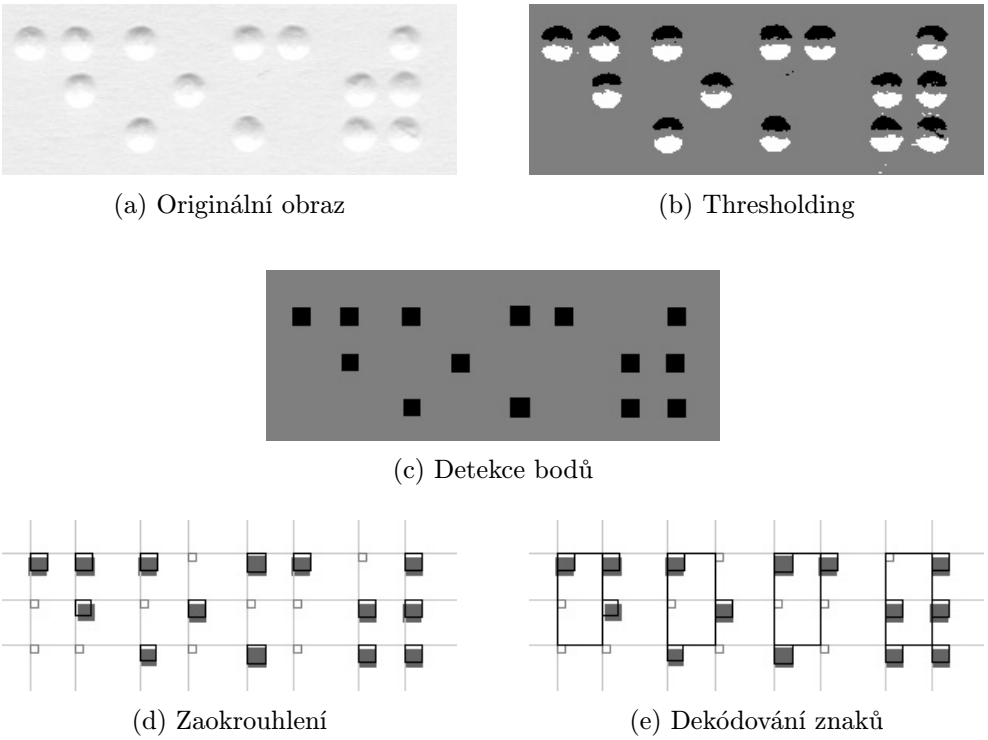
1. Procházíme první body všech řad pole a nalezneme nejlevější bod.
2. Nalezneme všechny body v dalších řadách, které jsou ve stejném sloupci jako vybraný bod. Je opět vhodné stanovit toleranci, v rámci které bod patří do stejného sloupce (korekce rotace).
3. Tyto body uložíme do (nového) dvourozměrného pole; buňky, které neobsahují žádný bod, necháme prázdné - jedná se o místa znaků. Uložené body označíme jako detekované (nebo odstraníme ze struktur).

4.7.4 Zaokrouhlení souřadnic řad a sloupců

Posledním krokem korekce rotace dokumentu je zaokrouhlení souřadnic řad a sloupců dvourozměrného pole. Tato korekce má význam pouze pokud použijeme metodu eliminace chybných bodů z kapitoly 4.7.5.

Zaokrouhlení probíhá tak, že všechny body v jedné řadě nastavíme na jednu hodotu souřadnice y a všechny body v jednom sloupci nastavíme na jednu hodotu souřarnice x. Nové hodnoty můžeme spočítat jako nejčastější souřadnici bodů, nebo vhodněji jako průměr všech daných souřadnic bodů.

Příklad zaokrouhlení lze vidět na obrázku 4.18d, celý postup detekce na 4.18.



(a) Originální obraz

(b) Thresholding

(c) Detekce bodů

(d) Zaokrouhlení

(e) Dekódování znaků

Obrázek 4.18: Příklad dekódování se zaoukrouhlením

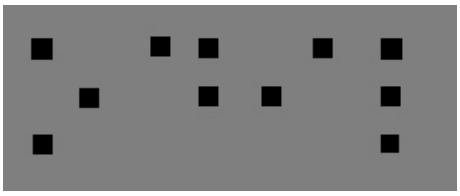
Toto zaokrouhlení má na dostatečně širokém a pootočeném dokumentu nevýhodu, protože limitní hranice řady nastavujeme od prvního bodu, ale první bod řady může mít

s posledním bodem větší výškový rozdíl, než toleranci (velké otočení). Řešením je používat detekce rotace z kapitoly 4.6, nebo při zachování předchozího postupu rozdělení do dvourozměrného pole eliminovat řady a sloupce, které jsou příliš blízko u sebe a sloučit je do jedné řady.

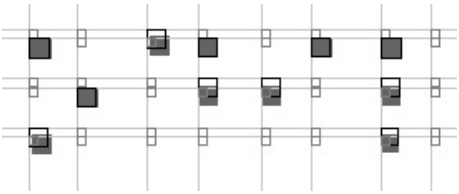
Tato metoda funguje pro korekci řad tak, že porovnáme vzdálenost dvou sousedních řad. Pokud je tato vzdálenost menší než tolerance (zpravidla šířka bodu), jedná se o jednu řadu, a proto je sloučíme do jedné řady. Pro sloupce je situace analogická, příklad lze vidět na obrázcích 4.19.

(a) Originální obraz

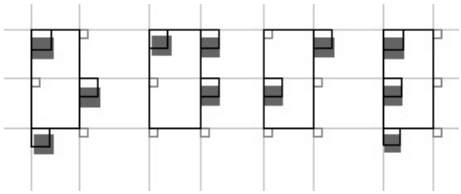
(b) Thresholding



(c) Detekce bodů



(d) Chybné zaokrouhlení



(e) Sloučení řad a dekódování znaků

Obrázek 4.19: Příklad dekódování s eliminací chybných řad

4.7.5 Detekce znaků s eliminací chybných bodů

Ve všech dosud uvedených detekcích pracujeme se správným rozložení bodů do dvourozměrné matice. Přes všechny uvedené redukce a korekce je vhodné detekovat znaky pomocí normy a nevěnovat pozornost prvkům, které jí neodpovídají. Zásadní je vzdálenost dvou bodů v rámci jednoho znaku, vzdálenost znaků mezi sebou a vzdálenost řádků. K tomu je potřeba nastavit vhodně hraniční limity. Tuto toleranci je dobré nastavit jako šířku bodu, resp její procentuální část - například 50 %. Pomocí této tolerance zvolíme horní a dolní limit vzdáleností.

Samotná detekce probíhá následujícím způsobem:

- Pokud vzdálenost sloupců bodů odpovídá nastaveným hodnotám, dekódujeme znak - pokud je větší, ignorujeme ho, nejedná se totiž o znak. Pokud je menší, porovnáme první sloupec s následujícím sloupcem.

- Dekódujeme řadu znaků pokud vzdálenost mezi dílčími řadami bodů v rámci znaku odpovídá nastaveným hodnotám a pokud je za řadou mezera větší než výšková vzdálenost dvou bodů (nebo se jedná o poslední řadu znaků). Pokud je vzdálenost řad bodů v rámci znaku větší, řadu ingorujeme. Pokud je menší, testujeme další řadu.

4.7.6 Jednosloupcové znaky

Při předchozí eliminaci je potřeba brát v úvahu situaci, kdy převádíme jen několikařádkový text v Braillově písmu, který obsahuje jednousloupcový znak. Předchozí metody jsou platné pro situace, kdy je v každém sloupci Braillova znaku v daném dokumentu alespoň jeden bod. Pokud není, předchozí eliminace znak nedetekuje. Tato situace není pro běžné víceřádkové dokumenty příliš častá, přesto je vhodné při každé detekci znaku kontrolovat vzdálenosti k vedlejším sloupcům znaků a podle zvolené normy zjišťovat, zda znak nemá jen jeden sloupec a zda se jedná o sloupec první, nebo druhý.

4.8 Přesvícené dokumenty

Nyní se budeme zabývat tzv. přesvícenými dokumenty, u nichž kvůli způsobu skenování nerozeznáme bílý stín a nemáme tudíž informaci o otočení bodu. Jednostranné dokumenty detekujeme bez problémů na základě předchozích metod, jen je potřeba použít thresholding pro přesvětlené dokumenty (například adaptivní z kapitoly 4.3.1). Pro oboustranné dokumenty máme několik možností.

První z nich je detekovat na základě příslušnosti k řadě - přiřazením bodů do řad a sloupců na základě znalosti oboustranných dokumentů, kdy jsou body z přední a zadní strany vzájemně posunuty (kapitola 3.3.3); přiřadíme všechny body v jedné řadě a sloupci k některé straně. K tomu je potřeba nějakým způsobem zvolit, zda první řada náleží k přední, nebo zadní straně. Toho dosáhneme tak, že budeme při skenování požadovat, aby první řada na začátku dokumentu patřila k přední straně. Druhou možností je rozpoznat tuto skutečnost podle zarovnání textu (viz kapitola 4.9).

Komplikací tohoto řešení je obtížnost určení středu původního bodu, který se nenalézá v černém stínu, který jako jediný máme k dispozici. Řešení brát v úvahu sloupce, nikoli řady; střed původního bodu vertikálně odpovídá středu černého stínu. K tomu je potřeba při detekci nastavovat velice pečlivě tolerance rozměrů.

Třetí možností je detekovat body na základě tvaru stínů; černé stíny jsou vypouklé určitým směrem, to určuje příslušnost ke straně dokumentu. Méně vypouklé stíny vždy patří k zadní straně (kapitola 3.3).

Detekování podle tvaru je možné zjednodušit na detekování dle informací o lokálních maximech a rozměrech bodu, z kterých lze zjistit, na kterou stranu je bod vypouklý - kde leží největší část stínu vzhledem k vypočtenému středu stínu.

Tyto informace můžeme použít k vypočtení původních středů bodů a použít první metodu detekce, nebo lépe detekovat jako nepřesvícený oboustranný dokument, přičemž tyto informace použijeme k rozpoznání příslušnosti ke straně dokumentu.

4.9 Otočené dokumenty

V kapitole 4.6 jsme uváděli možnost otočení dokumentu o 180°, rozpoznat toto otočení totiž není zcela snadné. Jednou z možností u víceřádkových nevycentrovaných textů je rozpoznávat podle zarovnání k levé straně. Spočítáme body dvou nejlevějších a dvou nejpravějších

sloupců, dokument je zarovnán k té straně, na které se nachází více bodů. Pokud je více bodů napravo je dokument otočen o 180°, a je potřeba jej otočit.

To můžeme udělat některou metodou rotace z kapitoly 4.6 a opakovat celou detekci, u exprotovaných bodů lze přepočítat jejich souřadnice, nebo efektivněji detekovat body otočeně: od spodního pravého rohu.

U oboustranných dokumentů je možné provést dvojnásobnou kontrolu pro zarovnání vlevo na přední straně a vpravo na zadní.

Kapitola 5

Implementace

Pro tvorbu vlastního programu jsem zvolil programovací jazyk C++ kombinovaný s knihovnou OpenCV (Open Source Computer Vision) [17], která je svobodným a otevřeným prostředkem pro manipulaci s obrazem. Je zaměřena především na počítačové vidění a zpracování obrazu v reálném čase. Původně ji vyvíjela společnost Intel. Může být zrychlena spoluprací s knihovnou Integrated Performance Primitives (Intel IPP). Knihovnu je možné využít z prostředí jazyků C, C++ a s generátorem rozhraní SWIG také Python a Octave.

Grafická nadstavba je vytvořena pomocí Qt frameworku [10], který je rovněž multiplatformní.

OpenCV je dostupný pod BSD[1] licencí a Qt pod GNU LGPL[2]. Obě jsou volně dostupné a díky multiplatfromitě i běžně použitelné. Instalace je možná dle návodů na jejich internetových stránkách.

5.1 Obrazová data

Digitalizovaný dokument je načten do OpenCV obrazové struktury `IplImage`. Obraz by bylo možné načíst barevně a následně převádět do stupňů šedi, nebo snadněji přímo načítat šedotónově pomocí funkce `cvLoadImage` za použití parametru `CV_LOAD_IMAGE_GRAYSCALE`. Takto je možné zpracovávat obrazová data s příponami: `*.bmp, *.dip, *.jpeg, *.jpg, *.jpe, *.png, *.pbm, *.pgm, *.ppm, *.sr, *.ras, *.tiff, *.tif, *.exr, *.jp2`.

Pro zobrazení OpenCV obrazu ze stuktury `IplImage` v Qt frameworku je potřeba nejdříve stukrutru převést na `QImage`, kterou už není problém v GUI pomocí `QGraphicsView` zobrazit. [11]

Pomocí další OpenCV funkce `cvSmooth` jsem implementoval Gaussovo rozostření za použití parametru `CV_GAUSSIAN`, definice velikosti okolí. Okolí lze zvolit $3 * 3$, $7 * 7$ a $9 * 9$. Standardní odchylku σ funkce dopočítává automaticky (rovnice 4.5 z kapitoly 4.2.1).

Pro globální jednoduchý thresholding je použita funkce `cvThreshold` s parametrem `CV_THRESH_BINARY`, kde další parametr `maxValue` je nastaven na hodnotu 255.

Adaptivní thresholding provádí funkce `cvAdaptiveThreshold` se přidanými parametry pro okolí $9 * 9$, typem thresholdingu `CV_ADAPTIVE_THRESH_GAUSSIAN_C` a konstantu $c = 10$.

[1] *Berkeley Software Distribution*
[2] *GNU Lesser General Public License*

5.2 Histogram

Přerozložení jasu a dvě metody thresholdingů jsou založeny na výpočtu histogramu. Ten jsem implementoval pomocí OpenCV funkce `cvCreateHist` se zjednodušením pro šedotónové obrazy [19]. Histogram je uložen do stuktury `CvHistogram` a pomocí funkce `cvGetMinMaxHistValue` jsou získány minimální a maximální četnosti a jejich odpovídající intenzity.

Histogram je možné v programu zobrazit, pro přehlednost jsem pro osu x použil logaritmické měřítko (funkce `log10`), příklad je na obrázku 5.1.

Obrázek 5.1: Zobrazení histogramu v programu

Určení nejčastější hodnoty intenzity provádím již během výpočtu histogramu - abych zvýšil rychlost programu - porovnáním s dočasnou nejčastější hodnotou; pokud jsou stejné, vybírám tu nižší.

Pro thresholding s minimální a maximální hodnotou (kapitola 4.3.4) potřebujeme určit lokální minimum nejčastější hodnoty s nejvyšší možnou intenzitou. Proto hledám nejméně častou hodnotu na daném intervalu. Při použití Gaussova rozostření tato metoda selhává, viz obrázek 5.2, protože nalezne hodnotu v okolí intenzity 255. Toto kontroluji tak, že porovnávám četnost nalezené intenzity s polovinou četnosti nejčastější hodnoty. Pokud je menší, hledám lokální minimum znovu tak, že postupuji od vrcholu nejčastější hodnoty k maximální intenzity dokud, nezačnou četnosti narůstat - nenarazím na lokální minimum.

Obrázek 5.2: Histogram rozostřeného obrazu

5.3 Redukce

Redukci šumu jsem implementoval podle algoritmů z kapitoly 4.4. Rychlost rozšířené jednopixelové redukce lze optimalizovat pomocí zásobníku tak, že při každé redukci pixelu si uložíme jeho souřadnice. Při další iteraci již neprocházíme celý program, ale hledáme šum pouze v okolí uložených souřadnic.

Pro paměťovou nenáročnost a velké možnosti redukce a korekce jsem se rozhodl implementovat detekci bodů pomocí vyexportovaných souřadnic, viz kapitola 4.5.

Další implementovaná redukce je redukce pomocí tvaru a rozměrů stínu v odvozené formě z kapitoly 4.5.3. Jako normu pro ni používám průměrnou šířku všech nalezených stínů (zabírající plochu větší než deset pixelů). Pro zrychlení programu normu vypočítávám při získání bodů z obrazu. Samotnou redukci provádím při prvním následujícím procházení všech bodů.

Pro rozdělení slitých bodů používám jako normu 90 % z průměrné šířky bodu - kvůli toleranci. Implementoval jsem elipsové překreslení (kapitola 4.5.5) s vodorovným otočení k ose x. Elipsu vykresluji pomocí funkce `cvEllipse`.

5.4 Rotace

Rotaci jsem implementoval jako výpočet z vyexportovaných bodů (kapitola 4.6.4), spolu s možností otočení vlastních obrazových dat.

Při následné rotaci souřadnic nezvětšuji rozměry obrazu pomocí rovnic z kapitoly 4.6.3, ale přepočítávám souřadnice i do záporných hodnot, které jsou platné pro následnou detekci a dekódování znaků.

Detekce rotace je volitelná, při automatickém dekódování se nespouští a používá se korekce rotace při vlastním dekódování.

5.5 Detekce a dekódování

Pro vnitřní reprezentaci Braillova znaku jsem zvolil šest booleovských hodnot dle tabulky 5.1, která se následně převádí na jedno šestimístné číslo. Pro slovník Braillova písma lze proto použít asociativní pole `QMap< int, QChar>`.

1	4
2	5
3	6

Tabulka 5.1: Vnitřní reprezentace Braillova znaku

Pomocí této reprezentace jsou vytvořeny konfigurační slovníkové soubory norem Braillova písma, který obsahuje na každém řádku jeden znak latinky následovaný mezerou a číselným kódem vnitřní reprezetnace. Za znakem `#` následuje řádkový komentář, který je při zpracování souboru ignorován. Prefixy se zapisjí pomocí znaku `@` následovaného číslem prefixu, jejichž význam zobrazuje tabulka 5.2. Českou normu obsahuje soubor *lang.brl*.

Implementoval jsem českou normu vzdáleností s co největší benevolencí. Za rozhodující považuji vzdálenost řad v rámci znaku, která musí být vždy stejná jako vzdálenost sloupců v rámci znaku (kapitola 2.4). Tu určuji jako nejčastější vzdálenosti mezi řadami bodů.

Samotná detekce a dekódování probíhá pomocí vektoru vyexportovaných souřadnic jedné strany (kapitola 4.7.3), je implementována následujícím způsobem:

1. Body rozdělím do řad a seřadím podle hodnot souřadnice x. Body rozdělím do sloupců. Oboje s tolerancí poloviny průměrné šířky bodu.

2. Zaokrouhlím body v každé řadě na její průměrnou hodnotu y a v každém sloupci na jeho průměrnou hodnotu x.

Číslo	Význam	Platnost / Použití
1	Číslo	Ukončené mezerou, dalším prefixem, nebo jinými znaky, než a - j, čárkou a tečkou.
2	Malé písmeno	Slouží pro ukončení platnosti prefixu pro řetězec znaků při zápisu malého písmene bez mezery (např. PhDr., 12a).
3	Velké písmeno	Pouze pro jeden následující znak.
4	Řetězec velkých písmen	Platnost je ukončena mezerou, interpunkčním znaménkem nebo prefixem jiného významu.
5	Malé řecké písmeno	Pouze pro jeden následující znak.
6	Velké řecké písmeno	Pouze pro jeden následující znak.

Tabulka 5.2: Čísla, významy a platnost prefixů

3. Provedu korekci a sloučení příliš blízkých řad (kapitola 4.7.4), jako toleranci jsem zvolil průměrnou šířku bodu.

4. Detekuji znaky, jejichž body jsou od sebe vzdálené dle normy (kapitola 4.7.5) s možností jednosloupcových znaků (kapitola 4.7.6). Limity jsem zvolil pomocí nejčastější vzdálenosti mezi řadami bodů s odečtením nebo přičtením poloviny průměrné šířky bodu.

5. Detekované znaky převádím do vnitřní reprezentace Braillova znaku a její pomocí do latinky. Prefixy ukládám pomocí kódů z tabulky 5.2.

6. Dekódovaný text s prefixy převedu pomocí stavového automatu (pravidla jsou v tabulce 5.2) na výsledný text, který bude zobrazen.

Při detekci bodů z druhé strany se postupuje stejně, jen se na začátku při prvním procházení vektoru bodů (kvůli optimalizaci) přepočtou souřadnice dle rovnice 5.1, aby body nebyly zrcadlově otočeny.

$$x_{new} = (imageWidth - 1) - x_{old} \tag{5.1}$$

5.6 Programové módy

Výsledný program bakalářské práce umožňuje pracovat v několika módech:

- Konzolový testovací mód, v kterém se grafické rozhraní nespouští. V parametrech se předává název souboru a parametry detekce. Je kompatibilní s testovacím programemz kapitoly 6.1.
- Práce s jedním souborem, kdy na začátku nastavíme parametry dekódování pro jeden soubor.
- Práce s více soubory, kdy nastavíme stejné parametry dekódování pro více souborů a dekódujeme je postupně.
- Krokování dekódování jednoho souboru, kdy lze parametry nastavovat v průběhu, zobrazovat dílčí kroky a vracet se k předchozímu.

Konzolový mód se spouští tak, že při spouštění programu se přidá parametr určující soubor pro dekódování. Při spuštění bez parametrů se spustí grafické rozhraní programu (kapitola 5.7).

Vícesouborové dekódování je implementováno jako opakované spouštění jednosouborového dekódování pro každý ze zadaných souborů.

Pro krokové dekódování jsou potřeba zálohovaná obrazová data z výsledku předchozího kroku, aby při změně metody kroku nebyly právě použitou metodou ovlivněny obrazová data.

5.7 Grafické rozhraní

Grafické rozhraní se skládá z několika oblastí:

- Výběr adresáře.
- Náhled obrazu.
- Parametry dekódování.
- Dekódovaný text.

Tyto oblasti se zobrazují v závislosti na následujících stavech programu:

- *Selecting*: vybírají se soubory a zobrazují náhledy dokumentů.
- *Decode* (dekódování jednoho souboru): je zobrazen náhled dokumentu a parametry dekódování.
- *Decode_all* (dekódování více souborů): je zobrazen seznam vybraných dokumentů a parametry dekódování.
- *Decode_step* (krokové dekódování): je zobrazen náhled dokumentu a parametry dekódování, které se mění v závislosti na aktuálním kroku dekódování.
- *Output*: je zobrazen náhled dokumentu a vypsán dekódovaný text.
- *Output_all*: je zobrazen seznam souborů a podle vybraného souboru je vypsán jeho dekódovaný text.

Náhledy programu jsou připojeny v příloze C.

Kapitola 6

Zhodnocení a testování

Program jsem testoval na jednostranném dokumentu s použitím různých parametrů metod, výsledky zobrazuje tabulka 6.1.

Preparing	Reducing	Thresholding	Get points	Errors	Success
no	one pixel	max & min	black	0	100 %,
no	all one pixel	max & min	black	0	100 %,
Gauss 3x3	one pixel	max & min	black	1	99.87 %,
Gauss 7x7	one pixel	max & min	black	1	99.87 %,
Gauss 9x9	one pixel	max & min	black	2	99.74 %,
no	one pixel	max	black	4	99.49 %,
no	all one pixel	max	black	3	99.62 %,
no	one pixel	max & min	white	6	99.24 %,
no	all one pixel	max & min	white	5	99.37 %,
no	one pixel	max & min	repoint black	4	99.49 %,
no	one pixel	max & min	repoint white	5	99.37 %,

Tabulka 6.1: Testování parametrů metod

Při automatických parametrech (první test) dekódování probíhá nejlépe. U běžných jednostránkových dokumentů program detekuje špatně jeden až dva znaky.

U oboustranných dokumentů je situace komplikovanější kvůli slévání bodů, text je přes chybné dekódování některých znaků srozumitelný.

Pro automatické konzolové testování jsem napsal krátký skript v Bashi.

6.1 Testování - creating points

Tento program jsem implementoval kvůli počátečnímu testování jednostranných dokumentů. Zadaný nebo náhodně generovaný text v latince překóduje do Braillova písma a uloží do obrazu, který je možné zpětně dekódovat. Umožňuje nastavení velikosti bodu v pixelech, podle které se přepočítává norma rozměrů. Je možné obraz otočit a tím testovat korekci rotace při následném dekódování. Pomocí slovníkového souboru je možné zvolit jazykovou normu znaků. Náhledy programu jsou v příloze E.

Kapitola 7

Závěr

V této bakalářské práci jsem se pokusil zmapovat několik možných postupů pro dekódování digitalizovaných dokumentů v Braillově písmu. Uvedené metody nepracují se stoprocentní úspěšností a je potřeba je vhodně vybírat v závislosti na konkrétní situaci a dokumentu.

Pro srovnání jsem analyzoval postup těchto metod a část jich implementoval ve výsledném programu.

Na základě jejich porovnání a otestování úspěšnosti jsem zvolil nejvhodnější metody a navrhl pro ně vhodné parametry, které jsem v programu nastavil jako výchozí při automatickém dekódování.

Zabýval jsem se zejména jednostrannými dokumenty a výsledky metod na nich použitých jsem aplikoval na oboustranné dokumenty. Zpočátku jsem se zabýval převážně rozpoznáváním bodů přímo v obrazových datech, ale pro velké výhody různých korekcí a eliminací chyb jsem se později rozhodl exportovat obrazová data do lépe zpracovatelných matematických struktur.

Pro thresholding jsem navrhl dvě logické neadaptivní metody vycházející z histogramu obrazového dokumentu. Adaptivnost jsem se rozhodl simulovat následnými redukcemi šumu.

Díky navržené několikanásobné korekci rotace během dekódování znaků není nutné u běžně digitalizovaných dokumentů zvlášť detekovat otočení.

Běžné jednostranné dokumenty navržený postup dekóduje s maximálně jednou nebo dvěma chybami. U oboustranných dokumentů dochází ke slití bodů a úspěšnost není tak vysoká, je ovšem možné postup a jednotlivé metody vylepšit.

Dále jsem navrhl postup pro dekódování tzv. přesvětelných dokumentů, u nichž kvůli absenci světlého stínu bodu nemáme informace o otočení bodu. Je vhodné dále rozvinout detekci bodu na základě tvaru stínu bodu, jejíž závěry je možné použít i pro detekci nepřesvětelných dokumentů.

Snažil jsem se o maximální obecnost vůči normě rozměrů znaku, kterou by bylo možné dále rozvíjet uvedenou detekcí DPI a lepším odvozením jejích rozměrů.

Analyzované postupy a výsledný program by mohl sloužit jako výchozí bod pro vytvoření programu, který by znatelně usnadňoval lidem práci s dokumenty v Braillově písmu, aniž by byli nuceni učit se Braillovu abecedu (případně více norem) a rozvíjet citlivost hmatu. Zvláště ve firmách, v nichž pracují vidomí spolu s nevidomými, by tento program mohl výrazně usnadnit jejich spolupráci.

Literatura

[1] Al-Salman, A.; AlOhali, Y.; AlKanhal, M.; aj.: An Arabic Optical Braille Recognition System. 2007.

[2] Bradski, G.; Kaehler, A.: *Learning OpenCV*. O'Reilly Media, 2008, ISBN 978-0-596-51613-0.

[3] Haiclová, V.: Hmatové obrázky. [online], [cit. 2013-01-23].
URL http://www.dotknisesveta.cz/

[4] Kalová, I.: Segmentace a detkce geometrických primitiv. [online], [cit. 2013-03-09].
URL http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/05%20-%20Segmentace%20a%20detekce%20geometrickych%20primitiv.pdf

[5] Kršek, P.: Základy počítačové grafiky. [online], [cit. 2012-11-06].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IZG-IT/texts/izg_opora.pdf?cid=7445

[6] Linda, G.; George, C.: *Computer Vision*. Prentice Hall, 2002, ISBN 0-13-030796-3.

[7] Menneus, J.; van Tichelen, L.; Francois, G.; aj.: Optical Recognition of Braille Writing Using Standard Equipment. 1994.

[8] Smýkal, J.: *Pohled do dějin slepeckého písma*. Česká unie nevidomých a slabozrakých, 1994.

[9] Smýkal, J.: *Tyflopedický lexikon jmenný*. Technické muzeum v Brně, 2006, ISBN 80-86413-30-6.

[10] WWW stránky: Qt framework. [online], [cit. 2012-01-10].
URL http://qt-project.org/

[11] WWW stránky: IplImage to QImage. [online], [cit. 2012-03-21].
URL http://umanga.wordpress.com/2010/04/19/how-to-covert-qt-qimage-into-opencv-iplimage-and-wise-versa/

[12] WWW stránky: Skenery a skenování. [online], [cit. 2012-11-06].
URL http://www.gyasos-prelouc.cz/sos/stranky/polygrafie_grafika_drobek/dokumenty/maturita10/graf_18_sken.pdf

[13] WWW stránky: 3D skenování. [online], [cit. 2012-12-05].
URL http://www.3d-skenovani.cz/o-3d-skenovani

[14] WWW stránky: Sjednocená organizace nevidomých a slabozrakých ČR. [online], [cit. 2013-02-02].
URL http://www.sons.cz/

[15] WWW stránky: World Health Organization. [online], [cit. 2013-02-21].
URL http://www.who.int/mediacentre/factsheets/fs282/en/

[16] WWW stránky: OBR Software. [online], [cit. 2013-03-13].
URL http://www.neovision.cz/prods/obr/

[17] WWW stránky: OpenCV. [online], [cit. 2013-04-07].
URL http://opencv.org/

[18] WWW stránky: Tyflokabinet České Budějovice, o.p.s. [online], [cit. 2013-04-20].
URL http://www.tyflokabinet-cb.cz/braill.htm

[19] WWW stránky: Histogram of grayscale image. [online], [cit. 2013-05-01].
URL http://karthicknopencv.blogspot.cz/2010/04/getting-histogram-for-given-grayscale.html

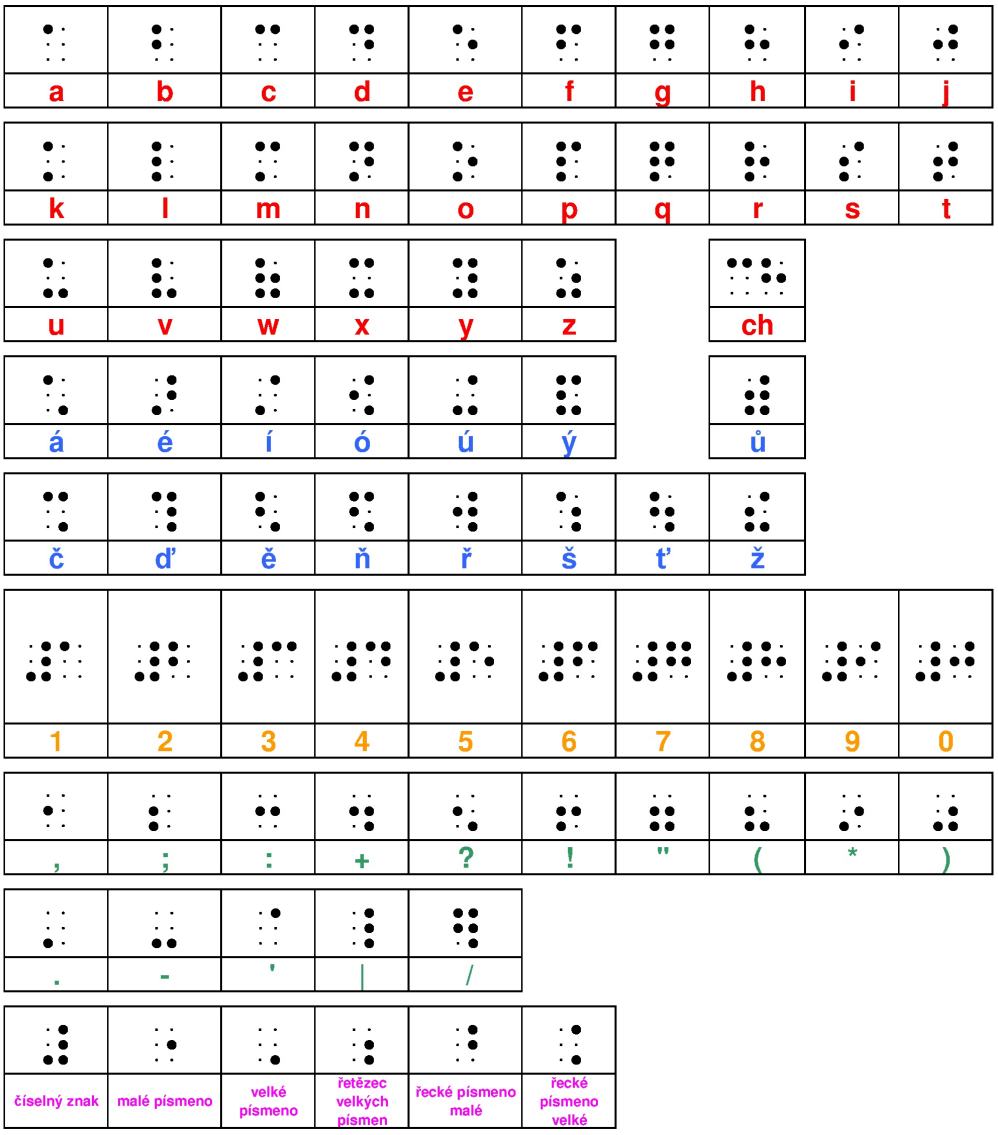
Příloha A

Obsah CD

- `creator` - adresář s pomocným programem pro vytváření obrazů s texty v Braillově písmu
- `documents` - adresář s digitalizovanými testovacími dokumenty (suffix `_g` znamená grayscale - šedotónové skenování a `_c` značí color - barevné skenování)
- `lang` - adresář se slovníkovým souborem české normy Braillovy abecedy
- `latex` - adresář se zdrojovými texty této technické zprávy
- `src` - adresář se zdrojovými soubory programu
- `ibp.pdf` - technická zpráva
- `install.txt` - návod na instalaci

Příloha B

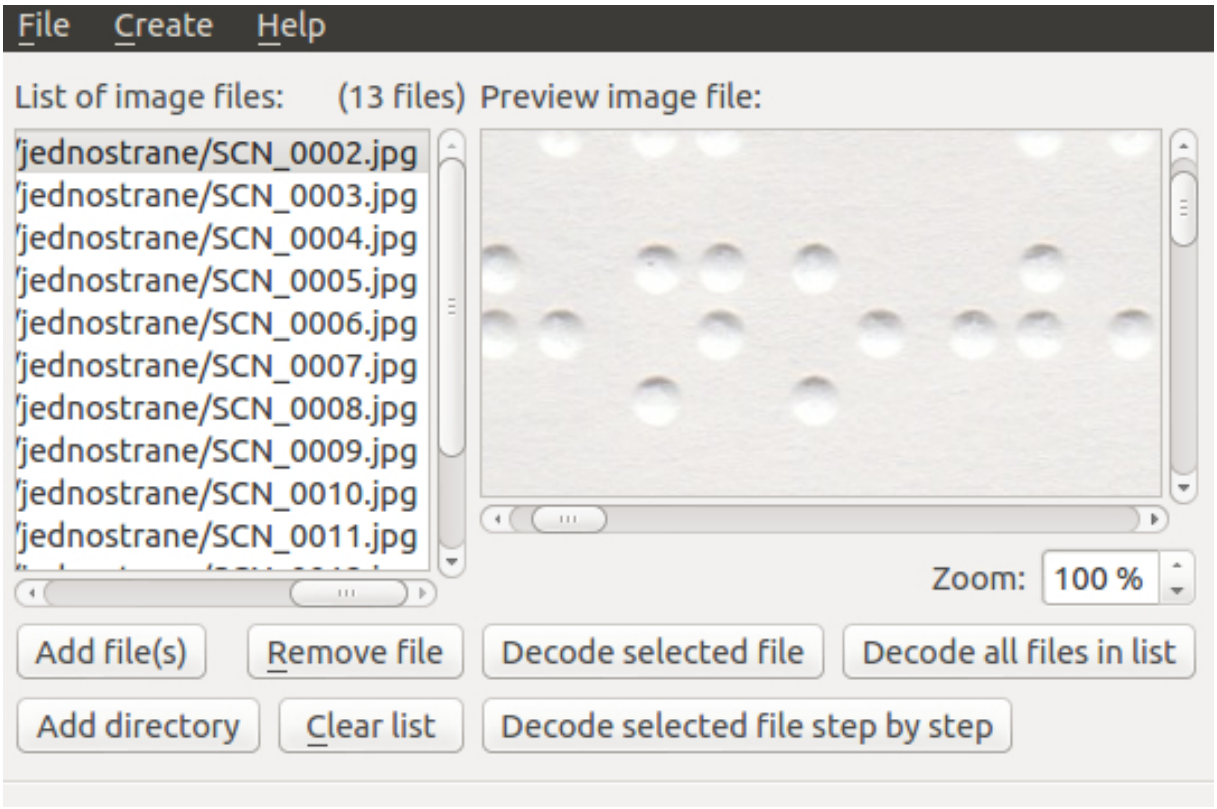
Česká norma Braillova písma



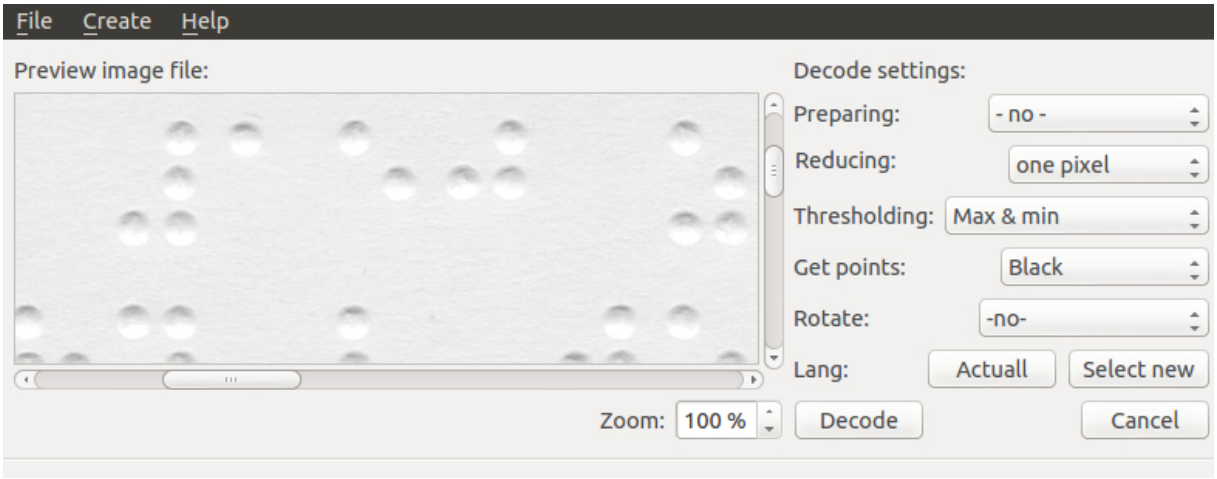
Obrázek B.1: Česká norma Braillova písma

Příloha C

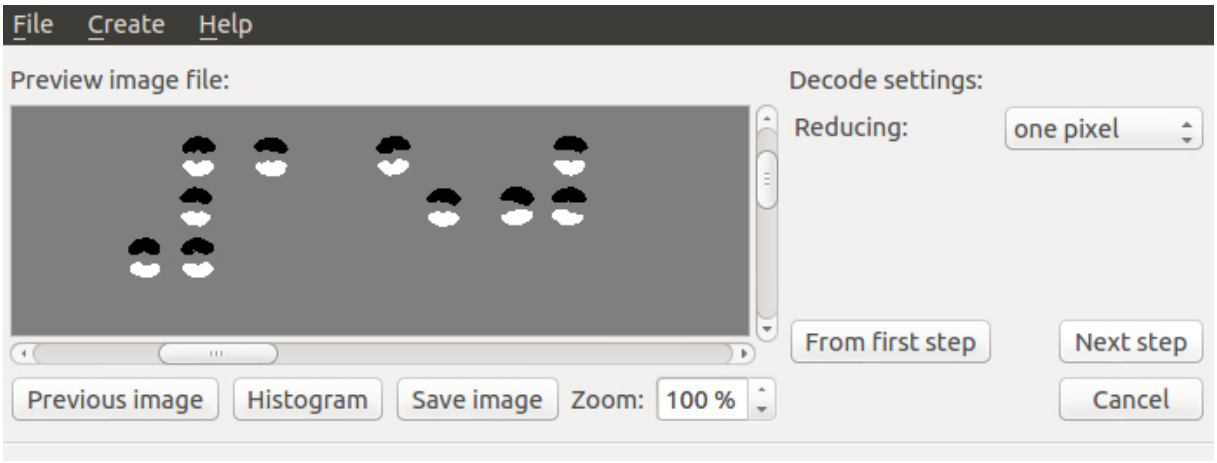
Náhledy programu



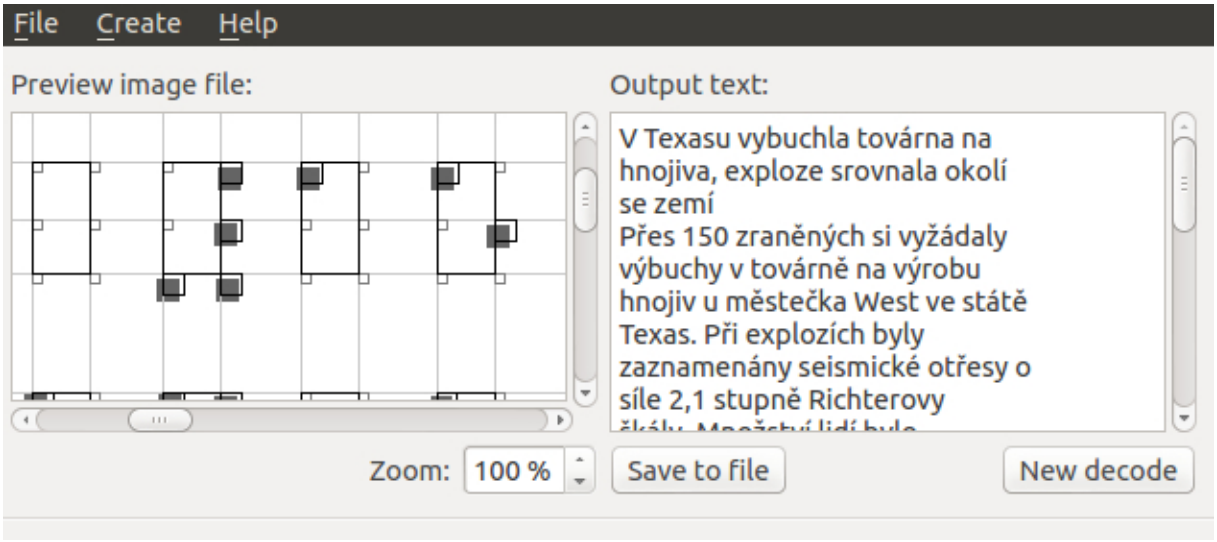
Obrázek C.1: Grafické rozhraní programu



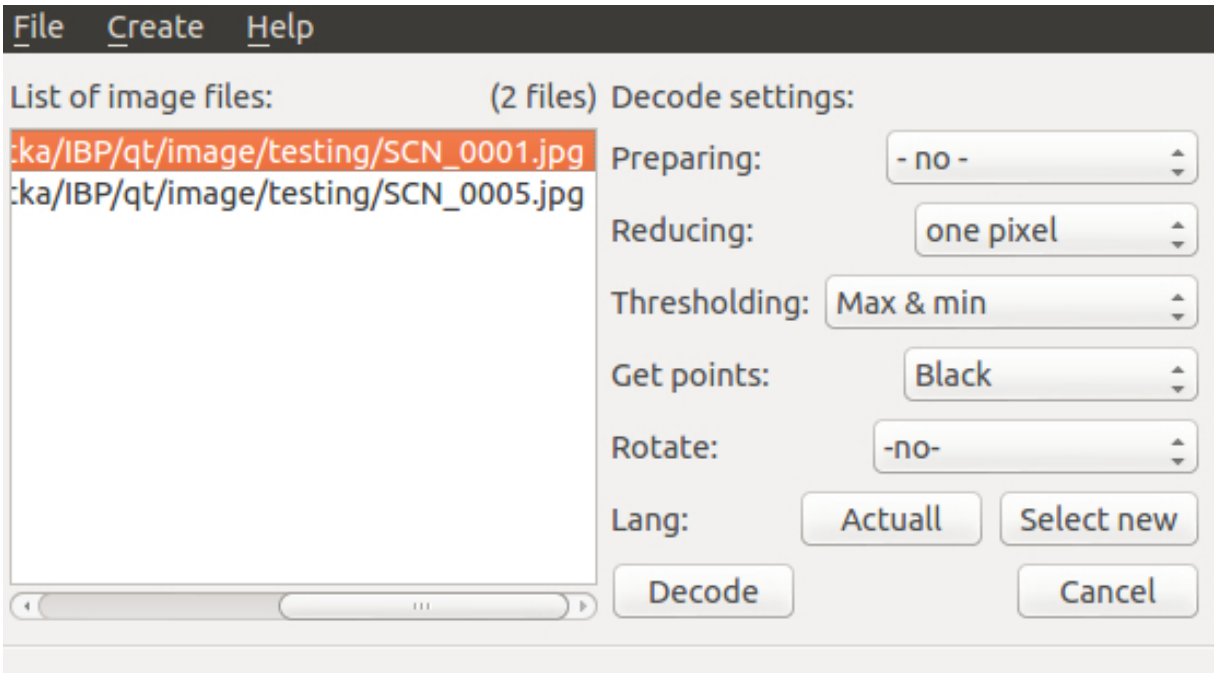
Obrázek C.2: Nastavení parametrů jednosouborového dekódování



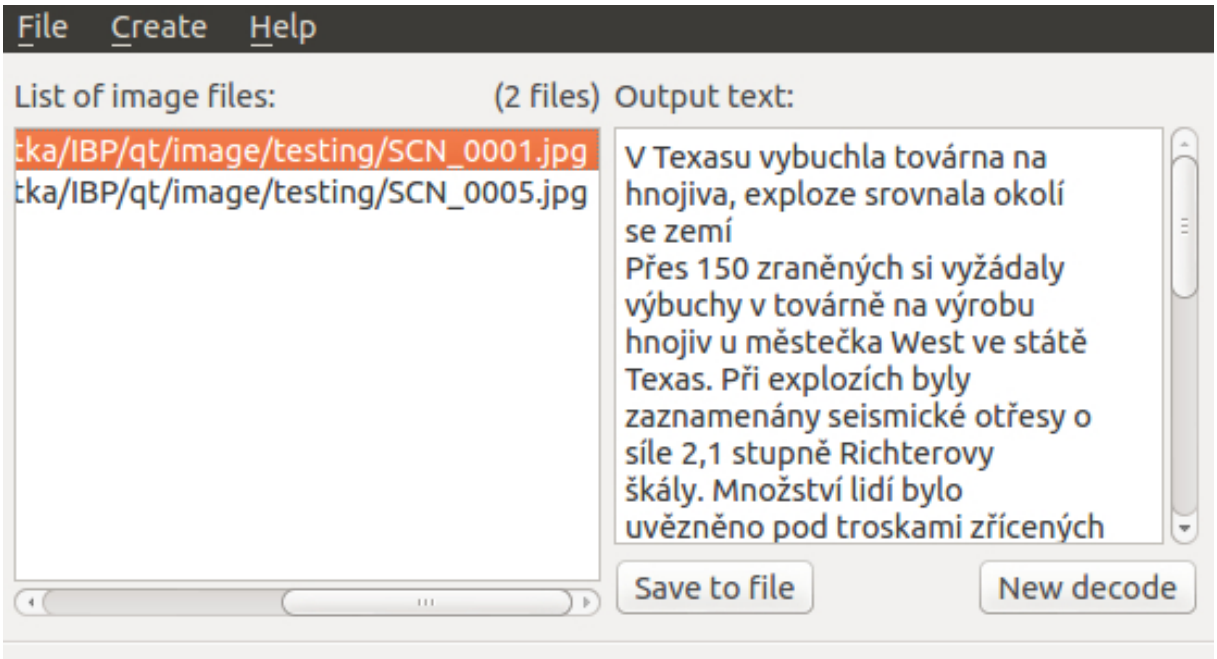
Obrázek C.3: Krokované dekódování



Obrázek C.4: Výstup jednosouborového a krokovaného dekódování



Obrázek C.5: Vícesouborové dekódování

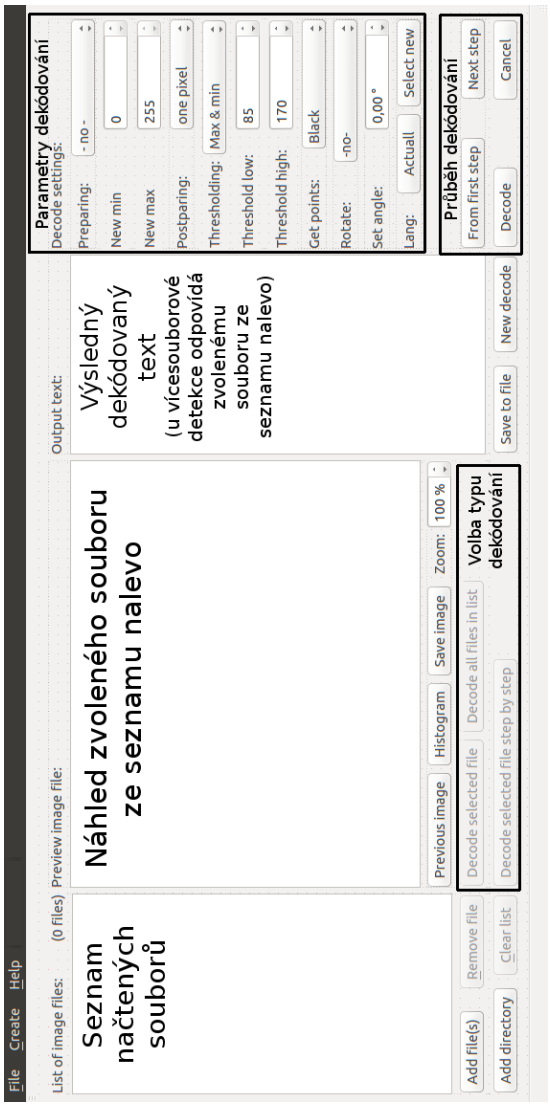


Obrázek C.6: Výstup vícesouborového dekódování

Příloha D

Ovládání programu

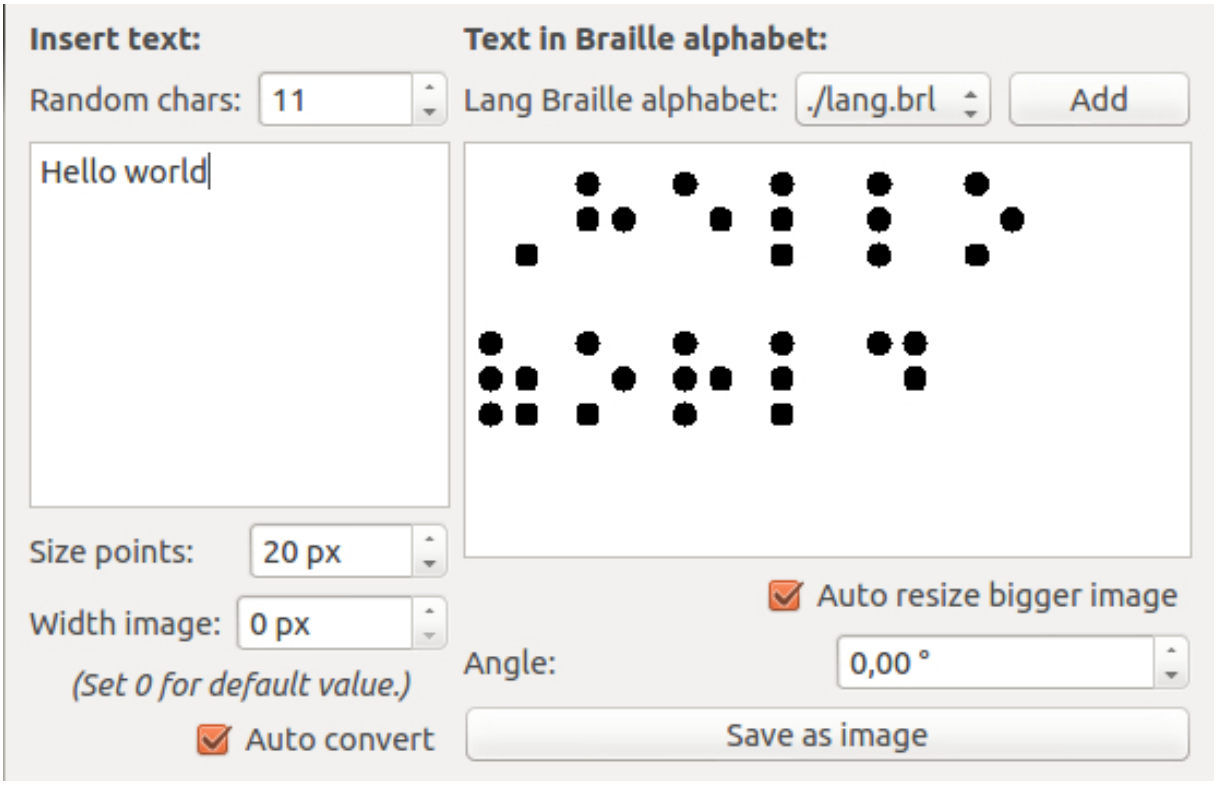
Na obrázku D.1 jsou zobrazeny ovládací prvky grafického prostředí s popisem funkce. Jejich zobrazení v závislosti na stavu programu popisuje kapitola 5.7.



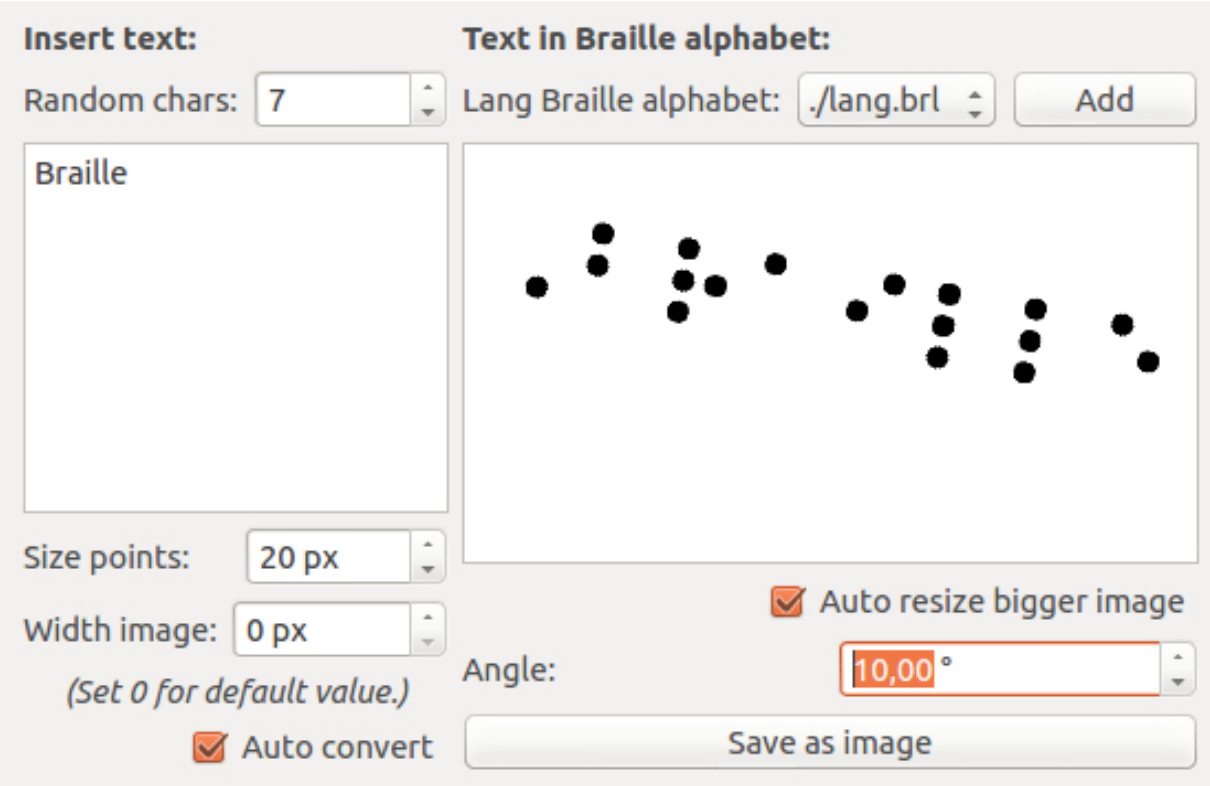
Obrázek D.1: Ovládací prvky programu

Příloha E

Testovací program



Obrázek E.1: Náhled testovacího programu



Obrázek E.2: Možnost rotace