VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

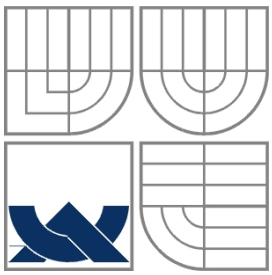
NÁSTROJ NA ZPRACOVÁNÍ FOTOGRAFOVANÉHO TEXTU

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

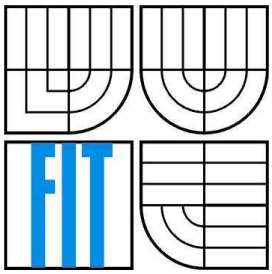
AUTOR PRÁCE
AUTHOR

MARTIN GENČÚR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NÁSTROJ NA ZPRACOVÁNÍ FOTOGRAFOVANÉHO TEXTU

APPLICATION FOR PROCESSING OF A PHOTOGRAPHED TEXT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN GENČÚR

VEDOUCÍ PRÁCE
SUPERVISOR

ING. LUKÁŠ GRULICH

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Genčúr Martin**
Obor: Informační technologie
Téma: **Nástroj na zpracování fotografovaného textu**
Kategorie: Počítačová grafika

Pokyny:

1. Analyzujte požadavky kladené na požadovaný nástroj. Cílem je vytvořit aplikaci, která upraví daný fotografovaný text (barevný, různé stíny, atd.) do černobílé (tj. dvoubarevné) podoby (popř. s využitím OCR do textového formátu).
2. Prostudujte algoritmy používané pro tyto operace, vyberte nejvhodnější, navrhněte strukturu aplikace.
3. Aplikaci implementujte ve vhodném prostředí, testujte, demonstrujte na vhodném příkladě.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Grulich Lukáš, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato práce poskytuje úvod do problematiky převodu fotografovaného textu do černobílé podoby. Rozebírá jednotlivé metody používané pro tuto úlohu. Následuje popis implementace aplikace, která provádí daný převod. Testuje program na vhodných datech (barevný obraz, obraz s různými odstíny apod.) a demonstruje použitelnost aplikace. Obsahuje také úvod do problematiky optického rozpoznávání znaků (OCR) a nastiňuje možné pokračování projektu tímto směrem.

Klíčová slova

Globální prahování, adaptivní prahování, segmentace, optické rozpoznávání znaků, histogram.

Abstract

This work introduces main approches for converting a photographed text into black and white form. It analyses particular methods being used for this task. Following part describes implementation of the application performing a conversion. The program is tested with suitable data (coloured picture, a picture with various shades etc.) and illustrates the usability of the application. The thesis also contains an introduction to optical character recognition (OCR) and suggests potential ways of development of this application.

Keywords

Global thresholding, adaptive thresholding, segmentation, optical character recognition, histogram.

Citace

Martin Genčúr: Nástroj na zpracování fotografovaného textu, bakalářská práce, Brno, FIT VUT v Brně, 2007

Nástroj na zpracování fotografovaného textu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Lukáše Grulicha
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Genčúr
4.5.2007

Poděkování

Tímto bych chtěl poděkovat Ing. Lukáši Grulichovi, který mi poskytl informace důležité pro dokončení práce a konzultoval se mnou možnosti řešení stěžejních bodů projektu.

© Martin Genčúr, 2007.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné,s výjimkou zákonem definovaných případů.

Obsah

Obsah 1
1 Úvod 3
2 Cíle práce 4
3 Problematika zpracování obrazu 5
4 Metody převodu 5
4.1 Globální prahování 5
4.2 Adaptivní prahování 7
4.2.1 Metoda Chow a Kanenko 8
4.2.2 Algoritmus Otsu (metoda optimálního prahu) 8
4.2.3 Lokální prahování 11
5 Programová realizace 14
5.1 Vývojové prostředí a překlad programu 14
5.2 Řešení problému převodu 14
5.2.1 Rozbor zvolené metody převodu 14
5.2.2 Použité třídy a komponenty 16
6 Popis programu 17
6.1 Hlavní menu programu 18
6.1.1 Menu Soubor 18
6.1.2 Menu převod 18
6.1.3 Menu Nápověda 19
6.2 Další součásti a funkce aplikace 19
6.3 Návod k použití programu 20
7 Testování programu 20
7.1 Test 1 20
7.2 Test 2 21
7.3 Test 3 22
7.4 Test 4 23
8 Optické rozpoznávání znaků (OCR) 24
8.1 Úvod do OCR 24
8.2 Implementace 24
9 Závěr 26

Literatura 27
Seznam příloh 28

1 Úvod

S rozvojem moderních technologií v posledních desetiletích došlo i k modernizaci přístupu k psanému a tištěnému textu. Psaný text byl postupně nahrazován textem elektronickým, který má řadu výhod oproti textu na běžném papíře. Mezi jeho zásadní přednosti patří jeho velmi jednoduchá upravitelnost, možnost opakovaně text vytisknout, uchovat pro pozdější použití apod. V současnosti si již lze jen těžko představit moderní komunikaci bez elektronického textu, ať již jde o posílání e-mailů, SMS, tvorbu elektronických knih, nebo vývoj software a hardware.

Stále častěji vzniká potřeba převádět již vytvořené texty do elektronické podoby. Dříve se převáděly tištěné texty do elektronické podoby tak, že byly ručně přepisovány. Dnes již ale existují nástroje pro automatický převod textu, které převedou text z papíru do elektronické podoby jen s malým přičiněním člověka. Tyto nástroje navíc mohou dále zpracovávat text například tak, že analyzují význam jednotlivých slov a jejich kombinací.

Převod textu do elektronické podoby obecně zahrnuje několik fází. Nejprve je nutné naskenovat (popřípadě nafotit) papírovou předlohu, která obsahuje samotný text. Takto získaný obraz ovšem není příliš vhodný pro další zpracování. Může být různě barevný (barevné písmo, pozadí std.) a může obsahovat různé stíny vzniklé např. nekvalitním snímáním předlohy. Proto je nutné obraz tzv. vyčistit a připravit pro další zpracování. Následně jsou znaky zpracovány technologií optického rozpoznávání znaků (OCR). Jedná se v podstatě o automatickou identifikaci grafických znaků snímaných opticky, případně o metodu elektronického čtení tištěných znaků a jejich převádění do elektronické podoby.

Právě vyčištění a připravení obrazu pro zpracování pomocí OCR je hlavní náplní této práce. Obrazy nasnímané povětšinou pomocí scanneru obsahují kromě užitečných informací (samotného textu) také zbytečné informace. Úkolem Nástroje na zpracování fotografovaného textu je především odstranění těchto přebytečných informací a to převedením obrazu do černobílé, tj. dvoubarevné podoby. Předloha s textem musí ovšem splňovat určité předpoklady. Je charakteristická tím, že obsahuje text, který je zobrazen jako tmavý na světlém pozadí. V takovém případě bude převod proveden správně a v obraze zůstanou jen písmena. V případě světlého textu na tmavém pozadí nebude výsledek převodu přijatelný. Vzniklý obraz je méně náročný na velikost diskového prostoru při ukládání a je vhodný pro OCR.

Pro převod obrazu s textem do dvoubarevné podoby se používá několik postupů, které se liší kvalitou zpracování, dobou převodu, paměťovou náročností a samotnou složitostí. V následujícím textu budou tyto postupy rozebrány, bude poukázáno na jejich výhody a nevýhody.

Dále bude popsána programová realizace jedné ze zvolených metod. Představíme si grafické uživatelské rozhranní vytvořené aplikace a její možnosti nastavování parametrů převodu. Následuje testování na několika vzorcích dat různého typu. V závěru se podíváme na jedno z možných řešení optického rozpoznávání znaků.

2 Cíle práce

Cílem této práce je vytvořit aplikaci, která upraví daný fotografovaný text do dvoubarevné (černobílé) podoby. Obraz může být různě barevný a obsahovat stíny. Cílem je také pokusit se o vytvoření programové nadstavby, která bude umožňovat dále zpracovat černobílý obraz pomocí optického rozpoznávání znaků a umožnit tak jeho převod do podoby textového souboru.
K splnění těchto cílů bylo stanoveno několik dílčích kroků:

- Prostudovat algoritmy používané pro tyto operace a vybrat z nich nejvhodnější s ohledem na složitost a výpočetní náročnost
- Vytvořit počítačový program, který by umožňoval na základě parametrů předaných uživatelem provádět převod obrazu do dvoubarevné podoby. Aplikace musí nabídnout příjemné grafické uživatelské rozhranní a být schopna v přijatelném čase zpracovávat jeden nebo více souborů obsahujících fotografovaný text. Požadavkem je také ukládání zpracovaných obrázků v různých formátech pro pozdější zpracování.

3 Problematika zpracování obrazu

Převod barevného obrazu do dvoubarevné podoby sestává ze dvou fází. Nejprve je nutné omezení barevného prostoru obrazu a to tím způsobem, že barva každého bodu v obraze (pixelu) je převedena do odstínu šedi. Poté již může být barva převedena do jedné ze dvou barev, černé nebo bílé.

Barva je subjektivní záležitostí lidského oka. Je tvořena vektorem spektrálních činitelů C = [r, g, b] = r*R + g*G + b*B , kde r,g,b jsou intenzity jasu jedné z barev R (červená), G (zelená) a B (modrá). Lidské oko vnímá barvy různě, nejcitlivější je na zelenou. Aby oko vnímalo barvu jako jeden z odstínů šedi, musí být do vzorce pro výpočet intenzity jasu každá ze složek barvy násobena jiným koeficientem:

I = 0,299 R + 0,587 G + 0,114 B

Tuto intenzitu světla I pak dosadíme za každou z barev R, G a B.

V tomto momentě máme obraz v odstínech šedi a můžeme použít jednu z metod pro převod do dvoubarevné podoby. Předpokládejme, že máme tmavé objekty (písmena) na světlém pozadí, jelikož jde o fotografovaný text. Body, které příslušejí písmenům v obraze, mají nízkou hodnotu odstínu šedi. Body, které představují pozadí obrazu, mají vysokou hodnotu odstínu šedi. Na základě těchto předpokladů můžeme rozdělit body do dvou hlavních skupin, v závislosti na hodnotě jejich odstínu šedi. Objekty (písmena) a pozadí. Musíme si zvolit úroveň šedi T mezi těmito dvěma skupinami, která bude sloužit jako prahová hodnota. Použitím této prahové hodnoty je možné vytvořit nový binární obraz, ve kterém jsou objekty vykresleny kompletně černou barvou a zbylé body bílou barvou. Pokud označíme hodnotu stupně šedi v originálním obraze f(x,y), pak výsledný obraz bude vytvořen tak, že budou skenovány jednotlivé body z originálního obrazu a porovnávány s prahovou hodnotou. Pokud f(x,y) > T, bude bod klasifikován jako pozadí, v opačném případě bude klasifikován jako bod objektu v popředí obrazu. Tento vztah můžeme vyjádřit následující definicí, kde b(x,y) značí výsledný obraz:

$$b(x,y) = \begin{cases} 255, \text{if } f(x,y) > T \\ 0, \quad \text{if } f(x,y) \le T \end{cases}$$

Tento postup se nazývá "prahování" (Thresholding). Existují různé varianty této metody. Dva hlavní přístupy se nazývají globální prahování a adaptivní prahování.

4 Metody převodu

4.1 Globální prahování

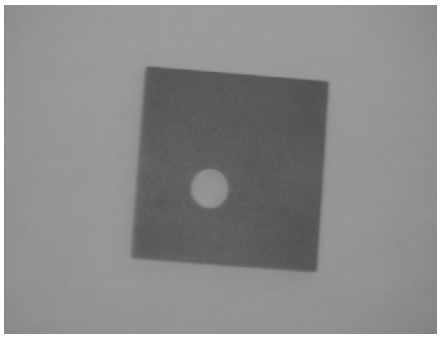
Český název byl převzat z anglického Global Thresholding. Vstupem operace prahování je typicky obraz v odstínech šedi. Metoda je charakteristická tím, že do ní vstupuje jediný parametr známý jako globální prahová hodnota. V tomto případě je každý pixel obrazu porovnáván s touto prahovou hodnotou. Pokud je intenzita bodu vyšší než prahová hodnota, je barva bodu nastavena na bílou, v opačném případě je bodu přiřazena černá barva.

Ovšem ne všechny obrázky mohou být takto segmentovány na objekty a pozadí. Zda může být obraz správně segmentován tímto způsobem nebo ne, může být zjištěno vyšetřením jeho histogramu intenzity světla (jasu).

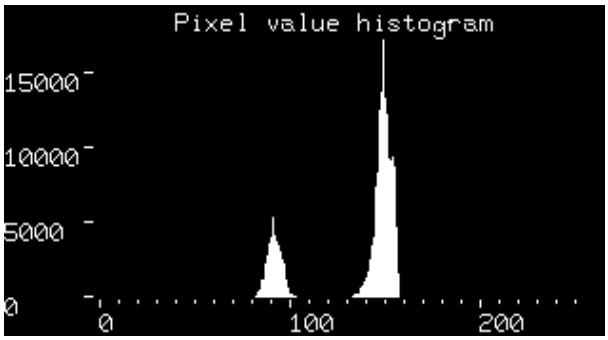
Histogram je graf zobrazující počet bodů (pixelů) v obraze pro všechny jednotlivé hodnoty intenzity nalezené v tomto obraze. Například pro 8-bitový obrázek v odstínech šedi je zde 256 možných stupňů intenzity, a tak histogram graficky zobrazí 256 hodnot, které představují rozdělení obrazovkových bodů mezi tyto stupně intenzity. Histogram také může být pořízen z barevného obrazu. Buď jsou vytvořeny 3 individuální histogramy pro červený, zelený a modrý barevný kanál nebo je vytvořen 3-D histogram, kde každý ze tří rozměrů reprezentuje jeden barevný kanál. Přesný výstup operace závisí na implementaci. Může to být jednoduše obrázek histogramu v jednom z formátů používaných pro uložení obrazu, nebo může vzniknout datový soubor jiného formátu reprezentujícího histogram.

Jednou z hlavních funkcí histogramu je určení prahové hodnoty, kterou je vhodné použít pro převádění obrazu v odstínech šedi do dvoubarevné podoby pomocí prahování. Pokud je obrázek vhodný pro globální prahování, bude histogram bi-modální, tj. intenzity jednotlivých bodů budou seskupeny okolo dvou dobře separovaných hodnot. Vhodná prahová hodnota pro oddělení těchto dvou skupin je někde mezi těmi dvěma vrcholy v histogramu. Pokud ovšem rozdělení intenzit nemá tuto charakteristiku, pak je nepravděpodobné, že globální prahování bude dávat pro zpracovávaný obraz přijatelný výsledek.

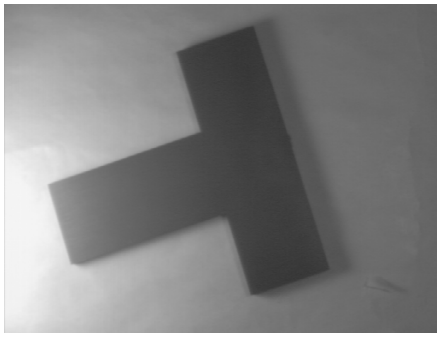
Následují dva typy obrázků s jejich histogramy. Je možné z nich usoudit, které typy obrázků jsou vhodné pro použití metody globálního prahování a které ne.



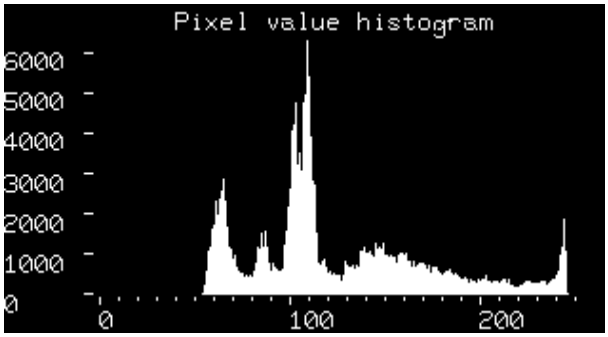
Obr. 4.1



Obr. 4.2 - Histogram k obrazu 4.1

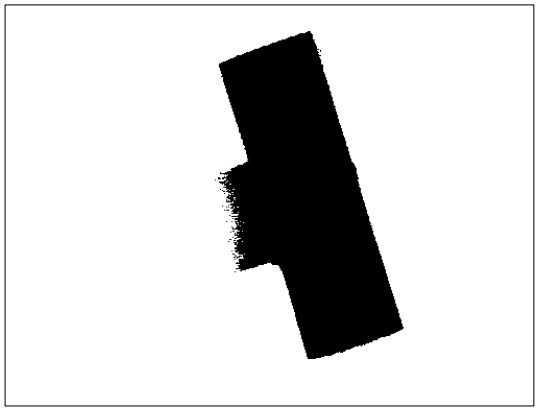


Obr. 4.3



Obr. 4.4 - Histogram k obrazu 4.3

Pokud bychom chtěli převést obrázek pomocí globálního metody, která pro jeho zpracování není vhodná, výsledek bude nepoužitelný:



Obr. 4.5 - Výsledek použití nevhodné metody prahování

V případě, že nelze použít metodu globálního prahování, musí být použito adaptivní prahování (Adaptive Thresholding)

4.2 Adaptivní prahování

Pokud se v obraze vyskytují různé stíny a šum, je potřeba zamyslet se nad určováním prahové hodnoty. Prahovou hodnotu není možné určit globálně, protože jedna hodnota není vhodná pro celý obraz jako celek. Z toho vyplývá, že pro jeden obraz musí být určeno více prahových hodnot.

Obecně je práh počítán pro každý pixel originálního obrazu, tato hodnota je poté použita pro porovnávání s hodnotou daného pixelu a vzniká nový obraz.

Definice prahové hodnoty může být zapsána následovně:

$$T = T[x,y,p(x,y),f(x,y)] ,$$

kde f(x,y) je stupeň šedi bodu (x,y) originálního obrazu a p(x,y) je určitá lokální vlastnost tohoto bodu. Pokud hodnota T závisí jen na odstínu šedi v daném bodě, potom z něj vzniká obyčejná globální prahová hodnota. Zvláštní pozornost by měla být věnována faktoru p(x,y), který byl popsán jako vlastnost určitého bodu. Ve skutečnosti je tato vlastnost jednou z nejdůležitějších prvků při počítání prahové hodnoty pro bod v obraze. Výpočet této vlastnosti je obvykle založen na okolí bodu, aby bylo možné vzít v úvahu vliv osvětlení a šumu. Vlastnost může být například průměrná hodnota v předdefinovaném okolí, jehož je daný bod středem.

Metoda adaptivního prahování je tedy typická tím, že mění velikost prahové hodnoty dynamicky nad obrázkem, který zpracovává. Tato lépe propracovaná verze prahování umožňuje přizpůsobit se stavu osvětlení v obraze, tj. velkým změnám intenzity osvětlení nebo stínům.

Adaptivní prahování typicky používá obrázek ve stupních šedi jako vstup a výstupem je dvoubarevný (černobílý) obraz. Pro každý pixel v obraze je počítána nová prahová hodnota. Převod poté probíhá stejným způsobem jako u prahování s globální hodnotou prahu: pokud je hodnota pixelu

nižší než prahová, je jeho barva nastavena na barvu popředí (barvu objektu), v opačném případě je bodu přiřazena hodnota pozadí.

Pro hledání prahové hodnoty se používá několik postupů. Mezi hlavní patří metoda Chow a Kanenka, algoritmus Otsu a "lokální prahování". Všechny metody využívají společného předpokladu a sice, že u menších oblastí obrazu je pravděpodobnější, že budou mít uniformní osvětlení. Proto se menší oblasti jeví jako vhodnější pro určování prahové hodnoty

4.2.1 Metoda Chow a Kanenko

Metoda od Chow a Kanenka (Chow and Kanenko approach) funguje na principu rozdělení obrazu do pole překrývajících se pod-obrazů a následném nalezení vhodné prahové hodnoty pro každý tento pod-obraz vyšetřením jeho histogramu. Prahová hodnota pro každý pixel je vypočtena interpolací hodnot jednotlivých pod-obrazů. Nevýhoda tohoto přístupu je výpočetní náročnost a tak není vhodná pro aplikace pracující v reálném čase.

4.2.2 Algoritmus Otsu (metoda optimálního prahu)

Tato metoda je pojmenována podle Nobuyuki Otsu, který ji vymyslel. Vychází z relativního histogramu a je založena na statistických výpočtech. Předpokládá existenci dvou tříd C_0 a C_1 (body objektů a body pozadí) rozdělené prahem v hodnotě k, což znamená, že body patřící do třídy C_0 mají hodnotu jasu v intervalu $\langle 0, k-1 \rangle$ a body ze třídy C_1 v intervalu $\langle k, J-1 \rangle$. Hodnota J odpovídá počtu jasových úrovní obrazu. Potom tedy pravděpodobnost výskytu bodu ze třídy C_0 (označíme ji jako ω_0) je rovna

$$\omega_0 = \sum_{i=0}^{k-1} p_i = \omega(k) \tag{4.1}$$

a pravděpodobnosti výskytu ve třídě C_1 (označíme jako ω_1) odpovídá vztah

$$\omega_1 = \sum_{i=k}^{J-1} p_i = 1 - \omega(k), \tag{4.2}$$

kde p_i je i-tá hodnota z relativního histogramu. Jelikož máme pouze dvě třídy, do nichž jsou všechny obrazové body rozděleny, platí vztah

$$\omega_0 + \omega_1 = 1. \tag{4.3}$$

Nyní určíme střední hodnotu obou tříd. Pro třídu C_0 platí

$$\mu_0 = \sum_{i=1}^{k-1} \frac{i p_i}{\omega_0} = \frac{\mu(k)}{\omega(k)}, \tag{4.4}$$

pro třídu C_1 platí

$$\mu_1 = \sum_{i=k}^{J-1} \frac{ip_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1-\omega(k)}. \tag{4.5}$$

Dále zavedeme hodnoty

$$\omega(k) = \sum_{i=0}^{k-1} p_i \quad \text{a} \quad \mu(k) = \sum_{i=0}^{k-1} ip_i \tag{4.6}$$

jako nultý a první centrální moment histogramu do jasu k-1 a hodnotu

$$\mu_T = \sum_{i=0}^{J-1} ip_i \tag{4.7}$$

jako celkovou střední hodnotu celého originálního obrazu. Z těchto vztahů nyní můžeme snadno odvodit platnost dalšího vztahu

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T . \tag{4.8}$$

Dále se zde zavádí pojem variance (různost, kolísání) třídy. Variance je definována pro třídu C_0 a C_1 takto:

$$\sigma_0^2 = \sum_{i=0}^{k-1} \frac{(i-\mu_0)^2 p_i}{\omega_0} \;,\; \sigma_1^2 = \sum_{i=k}^{J-1} \frac{(i-\mu_1)^2 p_i}{\omega_1} . \tag{4.9}$$

Nyní se dostáváme k jádru problému, definujeme totiž takzvanou mezitřídovou varianci σ_B^2, jejíž maximalizací získáme hodnotu prahu v histogramu. Platí pro ni následující vztah

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2, \tag{4.10}$$

který si ještě dále upravíme a zjednodušíme aplikací výše uvedených vzorců. Následující úpravy si podrobně rozepíšeme. Nejprve tedy dosadíme za μ_T ze vztahu (4.8), umocníme a roznásobíme výrazy v závorkách, čímž obdržíme vztah

$$\begin{aligned}\sigma_B^2 = {} & \omega_0\mu_0^2 - 2\mu_0^2\omega_0^2 - 2\mu_0\omega_0\mu_1\omega_1 + \omega_0^3\mu_0^2 + 2\mu_0\omega_0^2\mu_1\omega_1 + \omega_0\omega_1^2\mu_1^2 + \\ & + \omega_1\mu_1^2 - 2\mu_0\omega_0\mu_1\omega_1 - 2\mu_1^2\omega_1^2 + \omega_0^2\mu_0^2\omega_1 + 2\mu_0\omega_0\mu_1\omega_1^2 + \omega_1^3\mu_1^2\end{aligned} . \tag{4.11}$$

Zaměříme se na šestý a poslední výraz a vytkneme z nich hodnotu $\omega_1^2\mu_1^2$, ze čehož nám vznikne výraz $\omega_1^2\mu_1^2(\omega_0+\omega_1)$. Podle výše uvedeného vztahu (4.3) víme, že $\omega_0+\omega_1=1$,a tedy získáme výsledný vztah pouze ve tvaru $\omega_1^2\mu_1^2$. Podobným způsobem zjednodušíme i čtvrtý a desátý výraz na tvar $\omega_0^2\mu_0^2$. Dále si vezměme třetí, pátý, osmý a předposlední výraz a vytknutím z nich vytvoříme výraz $2\mu_0\omega_0\mu_1\omega_1(-1+\omega_0-1+\omega_1)$. Výraz v závorce se nám opět zredukuje a to na hodnotu -1.

Napíšeme-li si nyní celý původní výraz znovu, zjistíme, že již došlo ke značnému zjednodušení. Jeho tvar bude

$$\sigma_B^2 = \omega_0\mu_0^2 - 2\mu_0^2\omega_0^2 + 2\mu_0\omega_0\mu_1\omega_1 + \omega_0^2\mu_0^2 + \omega_1^2\mu_1^2 + \omega_1\mu_1^2 - 2\mu_1^2\omega_1^2 . \tag{4.12}$$

Sečteme druhý výraz se čtvrtým a pátý s posledním, vyjde nám rovnice

$$\sigma_B^2 = \omega_0\mu_0^2 - \mu_0^2\omega_0^2 + 2\mu_0\omega_0\mu_1\omega_1 + \omega_1\mu_1^2 - \mu_1^2\omega_1^2 . \tag{4.13}$$

Vytknutím z jejích prvních dvou členů získáme výraz $\omega_0\mu_0^2(1-\omega_0)$, což je vlastně $\omega_0\omega_1\mu_0^2$.

Aplikací stejného postupu na poslední dva členy uvedené rovnice obdržíme výraz $\omega_0\omega_1\mu_1^2$. Opět si napíšeme celou rovnici, jak by vypadala po této úpravě

$$\sigma_B^2 = \omega_0\omega_1\mu_0^2 + \omega_0\omega_1\mu_1^2 - 2\mu_0\omega_0\mu_1\omega_1 . \tag{4.14}$$

Nyní již jen provedeme vytknutí $\omega_0\omega_1$ ze všech členů a ještě drobnou úpravu, kde zbylé členy přepíšeme na čtvercový tvar a tím se dopracujeme k výslednému tvaru celé rovnice

$$\sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 . \tag{4.15}$$

Tento vztah můžeme ještě dále přepsat pomocí vztahů (4.1), (4.2), (4.4) a (4.5) na tvar

$$\sigma_B^2(k) = \omega(k)\big(1-\omega(k)\big)\left(\frac{\mu_T - \mu(k)}{1-\omega(k)} - \frac{\mu(k)}{\omega(k)}\right)^2 . \tag{4.16}$$

Zlomky rozšíříme a převedeme na společného jmenovatele, z čehož obdržíme vztah

$$\sigma_B^2(k) = \omega(k)\big(1-\omega(k)\big)\left(\frac{\mu_T\omega(k) - \mu(k)\omega(k) - \mu(k) + \mu(k)\omega(k)}{\omega(k)\big(1-\omega(k)\big)}\right)^2 .$$

Vidíme, že nyní můžeme vykrátit jmenovatele s výrazem před zlomkem a rovněž můžeme dopočítat čitatele ve zlomku. Výsledkem je vzorec ve tvaru

$$\sigma_B^2(k) = \frac{\big[\mu_T\omega(k) - \mu(k)\big]^2}{\omega(k)\big[1-\omega(k)\big]} . \tag{4.17}$$

Hledanou prahovou hodnotu pak dostaneme vyhodnocením tohoto výrazu pro všechna $k \in \langle 0, J-1\rangle$ a nalezením takového k, pro které výraz nabude maximální hodnoty.
Výsledké k^* tedy bude, zapsáno matematicky

$$\sigma_B^2(k^*) = \max_{0\le k<J-1} \sigma_B^2(k) . \tag{4.18}$$

Jedná se tedy o nalezení prahu v takovém místě histogramu, kde vzdálenost středních hodnot obou dvou tříd je maximální a dochází tak k optimální separaci objektů a pozadí.

Ve výpočtu mezitřídové variance (4.17) se nám vyskytuje zlomek a jeho jmenovatel představuje jisté nebezpečí při provádění výpočtu, pokud by nabýval nulové hodnoty. K tomu může dojít ve dvou případech. Pokud je výraz $\omega(k)$ roven nule nebo pokud je výraz $1-\omega(k)$ roven nule.

První případ nastane, pokud všechny hodnoty relativního histogramu až do k-tého budou nulové (vyplývá to ze vztahu uvedeného výše), což ovšem může nastat pouze v situaci, kdy se takovéto jasy v obrázku vůbec nevyskytují. Problému tedy můžeme předejít, pokud začneme počítat mezitřídovou varianci až od prvního (nejnižšího) existujícího jasu v obraze.

Druhý případ nastane, pokud $\omega(k)$ bude maximální, tedy rovno jedné. To se projeví v okamžiku, kdy k bude nabývat hodnoty jasu, po nemž už se žádný jiný jas s vyšší hodnotou v obraze nevyskytuje. Tomuto případu můžeme zamezit tak, že mezitřídovou varianci budeme počítat pouze do předposledního jasu vyskytujícího se v obraze.

Při předcházení těmto problémům je tedy nutné nejdříve zjistit první a poslední existující jas v obraze. Pak je tedy k ze vztahu (4.18) omezeno následující podmínkou:

první hodnota jasu $\leq$ k < poslední hodnota jasu

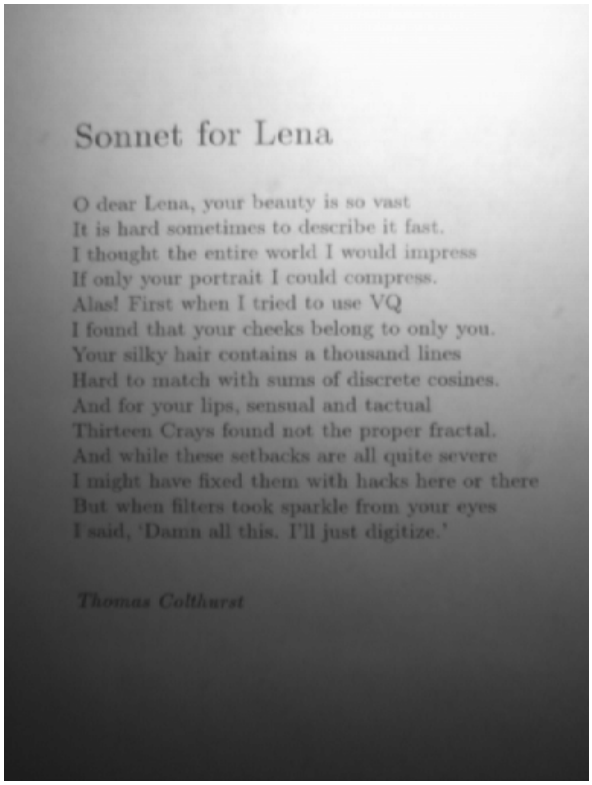
4.2.3 Lokální prahování

Metoda lokálního prahování hledá prahovou hodnotu pro každý pixel statistickým vyšetřením intenzit jasu v lokálním okolí bodu. Druh statistiky, který je nejvhodnější, je do značné míry závislý na vstupním obrazu. Jednoduché a rychlé funkce pro tuto operaci počítají s hodnotou T, kterou je možno chápat třemi způsoby. Prvním přístupem je určení střední hodnoty z intenzit jasu v lokálním okolí bodu (**T = střední hodnota**). Dalším je určení hodnoty zvané medián, což je prostřední hodnota z intenzit jasu v okolí (**T = medián**). Posledním a nejjednodušším způsobem je vypočtení T jako
T = (max - min) / 2, kde max je rovno maximální hodnotě jasu v okolí bodu a min je rovno minimální hodnotě v daném okolí.

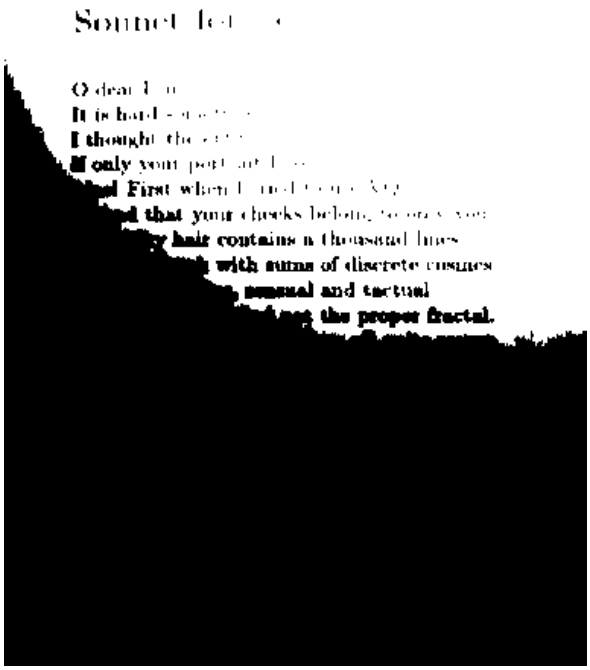
Velikost okolí musí být tak rozlehlá, aby pokryla dostatečně mnoho pixelů, které jsou součástí objektů (písmen) i pixelů pozadí. V opačném případě je zvolena špatná prahová hodnota. Na druhou stranu, pokud vybereme příliš velké okolí bodu, může to vést k porušení předpokladu, že osvětlení v lokálním okolí bodu se příliš nemění. Tato metoda je méně výpočetně náročná než algoritmus Otsu a produkuje dobré výsledky pro mnohé aplikace.

Jako globální prahování, je lokální adaptivní prahování používáno k oddělení objektů obrazu (popředí) od pozadí. Globální prahování používá jednu prahovou hodnotu pro celý obraz, ovšem je použitelné jen pokud histogram intenzity osvětlení v obraze splňuje určité požadavky. Nemůže být například použito pro obraz obsahující velké změny osvětlení. Lokální adaptivní prahování volí pro každý bod jinou hodnotu prahu a tak může být použito pro obrazy, jejichž histogram intenzity osvětlení neobsahuje dva oddělené vrcholy.

Vhodným objektem pro testování adaptivního lokálního prahování je obrázek 4.6



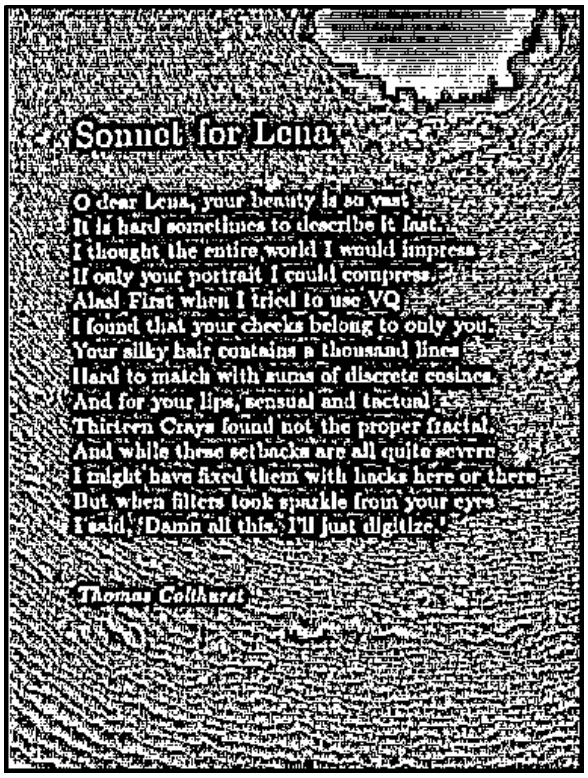
Obr. 4.6 - Originální obraz

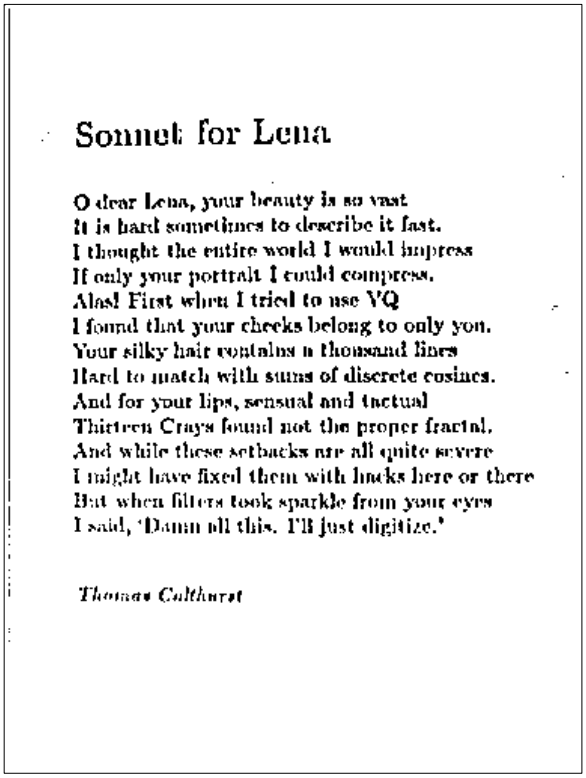


Obr. 4.7 - Výsledek globálního prahování

Protože obraz obsahuje velký pokles osvětlení, je výsledek globálního prahování na obrázku 4.7 nevhodný pro další použití.

Použití střední hodnoty "mean" a okolí 7x7 pixelů pro lokální prahování dává přijatelnější výsledek. Jak je vidět na obrázku 4.8, metoda je úspěšná v oblasti obklopující samotný text, protože je zde dostatek bodů popředí (objektů) a zároveň bodů pozadí v lokálním okolí každého pixelu. V takovém případě střední hodnota leží mezi samotnými hodnotami jasu popředí a pozadí, a tak mohou být objekty snadno odděleny od pozadí. Na okrajích obrazu není ovšem střední hodnota vhodná pro prahování, protože rozsah intenzit jasu v lokálním okolí bodu je velmi malý a jejich střední hodnota se blíží hodnotě samotného pixelu, který je zpracováván.





Obr. 4.8 - Lok. prahování bez optimalizace

Obr. 4.9 - Lok. prahování s optimalizací

Lepších výsledků může být dosaženo použitím metody zvané (mean-C). Tato metoda pracuje takovým způsobem, že barva všech pixelů, které se nalézají v uniformním okolí (tj. podél okrajů), je nastavena na barvu pozadí. Pro zmíněné uniformní okolí se používá zvláštní prahová hodnota C. Parametr C může mít například hodnotu C=10. Správná volba hodnoty závisí na charakteristice obrazu, který je zpracováván. Výsledek takovéto metody s okolím o velikosti 7x7 pixelů a hodnotou C = 7 je na obrázku 4.9.

Metody lokálního prahování jsou určeny pro zpracování obrazů obsahujících text. Pokud však obraz obsahuje velký objekt, dochází k chybnému převodu. Barva všech pixelů uvnitř objektu, které nemají ve svém okolí žádný pixel náležící do pozadí, jsou nastaveny na barvu pozadí, i když by tomu tak být nemělo. Takové chování lze ovlivnit změnou velikosti okolí bodu na vysokou hodnotu, ovšem výpočty pro okolí velké například 140x140 pixelů mohou být časově náročné. Potom je lepší použít jinou metodu než adaptivní lokální prahování.

5 Programová realizace

V této kapitole si rozebereme postup použitý pro převod fotografovaného textu do dvoubarevné podoby. Popíšeme kroky, které je nutné učinit k dosažení uspokojivých výsledků. Dále se seznámíme s vývojovým prostředím, jež bylo použito pro implementaci aplikace a zmíníme hlavní komponenty pro vizualizaci výsledku převodu.

5.1 Vývojové prostředí a překlad programu

Program je napsán v programovacím jazyce C++. Je použito několik druhů datových kontejnerů ze standardní knihovny tříd (STL). Pro implementaci uživatelského rozhranní bylo použito vývojové prostředí Borland C++ Builder Enterprise Suite ve verzi 6.0. Toto prostředí je určeno pro vývoj aplikací pro Windows a obsahuje podporu webových služeb, grafického uživatelského rozhranní a vývoj databázových aplikací. Prostředí vyvinula firma Borland (www.borland.com). Obsahuje překladač jazyka C++. Program je možné přeložit dvěma způsoby. Prní možnost je z příkazové řádky programem bcc.exe, který lze nalézt v adresářové struktuře produktu C++ Builder po nainstalování do počítače. Typicky je umístění tohoto souboru v adresáři C:\Program Files\Builder\Bin. Příklad zkompilování souboru tímto překladačem může vypadat takto:

"bcc32.exe Converter.cpp"

Druhou možností je ve vývojovém prostředí otevřít soubor projektu s příponou .bpr a překlad tak provést bez nutnosti cokoliv zadávat z příkazové řádky. Pro kompilaci projektu je ovšem nutné provést určitá nastavení, aby bylo později možné výslednou aplikaci spustit na jakémkoliv počítači s příslušnými Windows. V nastavení projektu je nutné odškrtnout volbu "Build with runtime packages" a "Use dynamic RTL". Tyto možnosti slouží pro kompilátor a pokud bychom nastavení neprovedli, překladač nezahrne do kompilace všechny soubory nutné pro správné spuštění a běh programu.

Z použití vývojové prostředí C++ Builder vyplývají jistá omezení týkající se komponent použitelných pro stavbu uživatelského rozhranní. Komponenty mají specifické vlastnosti a metody, kterým je nutno se přizpůsobit.

5.2 Řešení problému převodu

Pro realizaci postupu převodu obrazu do dvoubarevné podoby byl zvolen přístup adaptivního lokálního prahování (popsán v sekci 4.2.3). Vzhledem k použití aplikace, kdy hlavním účelem je převést obrázky a uložit je pro další použití, realizace pomocí této metody je dostačující. Jednotlivé obrázky jsou zpracovány v řádu sekund, což je přijatelná hodnota. Výsledek metody je při použití jistých optimalizací vhodný k dalšímu zpracování, například pomocí optického rozpoznávání znaků.

5.2.1 Rozbor zvolené metody převodu

Jak bylo uvedeno v kapitole o problematice převodu obrazu do dvoubarevné podoby, celý tento postup sestává ze dvou fází. Nejprve je nutné omezení barevného prostoru obrazu převedením barvy každého bodu (pixelu) do odstínu šedi.

Programová konstrukce specifická pro prostředí C++ Builder může vypadat následovně:

```
for(int i=0;i<vyska;i++) {
    for(int j=0;j<sirka;j++) {
      color = bit->Canvas->Pixels[j][i];  //bitmapa
      red = GetRValue(color);
      green = GetGValue(color);
      blue = GetBValue(color);
      barva=0.299*red + 0.587*green + 0.114*blue;
      color = static_cast<TColor>(RGB(barva, barva, barva));
      zaloha[i][j] = GetRValue(color);
    }
}
```

Nejprve si musíme zjistit rozměry obrázku. Procházíme všechny body obrazu a pro každý bod zjišťujeme jeho barevné složky (červenou, zelenou a modrou). Když známe složky, můžeme vypočíst hodnotu jasu a poté ji dosadit za každou ze složek R, G, B. Zkomponujeme barvu a jednu její barevnou složku uložíme do nového pole. Položky tohoto typu jsou unsigned char, aby byla efektivně využita paměť.

Po převodu obrazu do odstínů šedi již můžeme aplikovat další postupy pro dokončení převodu. Stěžejním bodem výpočtu je zjišťování prahové hodnoty pro každý pixel, podle níž bude následně rozhodnuto, jestli se má jeho barva změnit na černou nebo bílou.

```
for(int i=okoli;i<vyska-okoli;i++){ //cyklus přes všechny řádky
    for(int j=okoli;j<sirka-okoli;j++) { //cyklus přes všechny sloupce
            //cyklus pro zpracování okolí bodu
            for(int y=i-okoli;y <= i+okoli;y++)
                  for(int x=j-okoli;x <= j+okoli;x++) {
                        policko = zaloha[y][x];
                        soucet += policko;
                        if (policko < min){
                              min = policko;
                        }
                        if (policko > max){
                              max = policko;
                        }
                  }
            //počet hodnot v okolí bodu
            ctverec = (okoli+okoli+1)*(okoli+okoli+1);
            //nalezení průměrné hodnoty okolí
            prumer = (int) soucet / ctverec;
            //porovnání s parametrem funkce
            if ((max-min) < max_min)
                  prumer = 7;
            //zjištění hledané prahové hodnoty a její uložení
            prah[i][j] = prumer;
            soucet = 0;
            max = 0;
            min = 255;
    }
 }
```

Z programového kódu vyplývá, že pro každý pixel obrazu je procházeno jeho okolí a je počítána průměrná hodnota T, která představuje práh. Zároveň je vidět ošetření situace, kdy okolí bodu je

homogenní, tj. neobsahuje dostatek bodů představujících objekty a bodů představujících pozadí. Taková situace nastává zejména v okrajích obrazu. Z toho důvodu je počítána odchylka maximální a minimální hodnoty jasu v okolí bodu a je porovnávána s hodnotou max_min, která vstupuje do celé funkce jako parametr. Uživatel si ji může nastavit přes grafické uživatelské rozhranní. Nastavuje se v závislosti na tom, jaký typ obrazu se zpracovává. Klasický obraz, který je získán snímáním papírové předlohy pomocí scanneru, obsahuje text, jehož barva je ve velkém kontrastu k pozadí. Typicky je pozadí bílé nebo téměř bílé. Při těchto předpokladech je vhodné nastavit hodnotu **max_min** na co největší. Ovšem v případě obrazu s velkými změnami osvětlení je vhodné parametr co nejvíce snížit. Pokud je odchylka maximální a minimální hodnoty v okolí bodu příliš malá (menší než nastavený limit), je prahová hodnota pro zpracovávaný pixel nastavena na velmi nízkou hodnotu, což zapříčiní, že při prahování se výsledná barva s největší pravděpodobností změní na bílou (barva pozadí).

Druhým parametrem, který si může uživatel nastavit, je velikost okolí. Správná volba této hodnoty je taktéž závislá na parametrech obrazu. Pro obraz s písmem (stěžejní pro tuto aplikaci) se běžná velikost okolí pohybuje od 7x7 do 25x25 bodů. Obecně platí, že čím větší objekty jsou v obraze, tím větší velikost okolí je možné nastavit.

Následuje jednoduchá metoda prahování:

```
for(int i=0;i<vyska;i++)
      for(int j=0;j<sirka;j++)
            if (zaloha[i][j] < prah[i][j])
                  cernobily[i][j] = 0;
            else
                  cernobily[i][j] = 255;
```

V obraze někdy zůstávají drobné vady jako osamocené černé pixely. Proto je provedeno ještě tzv. vyčištění obrazu, kdy je znovu procházen celý obraz. Černé body, jenž sousedí s jediným dalším bodem s černou barvou nebo jsou zcela osamoceny, změní svou barvu na barvu pozadí.

Doba převodu není zanedbatelná. Aby nezatuhlo grafické uživatelské rozhranní po dobu výpočtu, jsou průběžně odebírány zprávy ze systému. K tomu slouží funkce osetritZpravy(MSG Message). Tato funkce nemá návratový typ a je mnohokrát volána v průběhu výpočtu. Program se tak stává ovladatelným i při zpracovávání větších obrázků.

5.2.2 Použité třídy a komponenty

Prostředí C++ Builder dává k dipozici třídu typu Graphics::TBitmap. Tato třída představuje bitmapu, do které je možné kreslit pomocí její vlastnosti Canvas. Po vytvoření instance této třídy a přiřazení její adresy proměnné daného typu je možno do této proměnné vykreslit celý obrázek a přistupovat k jednotlivým pixelům.

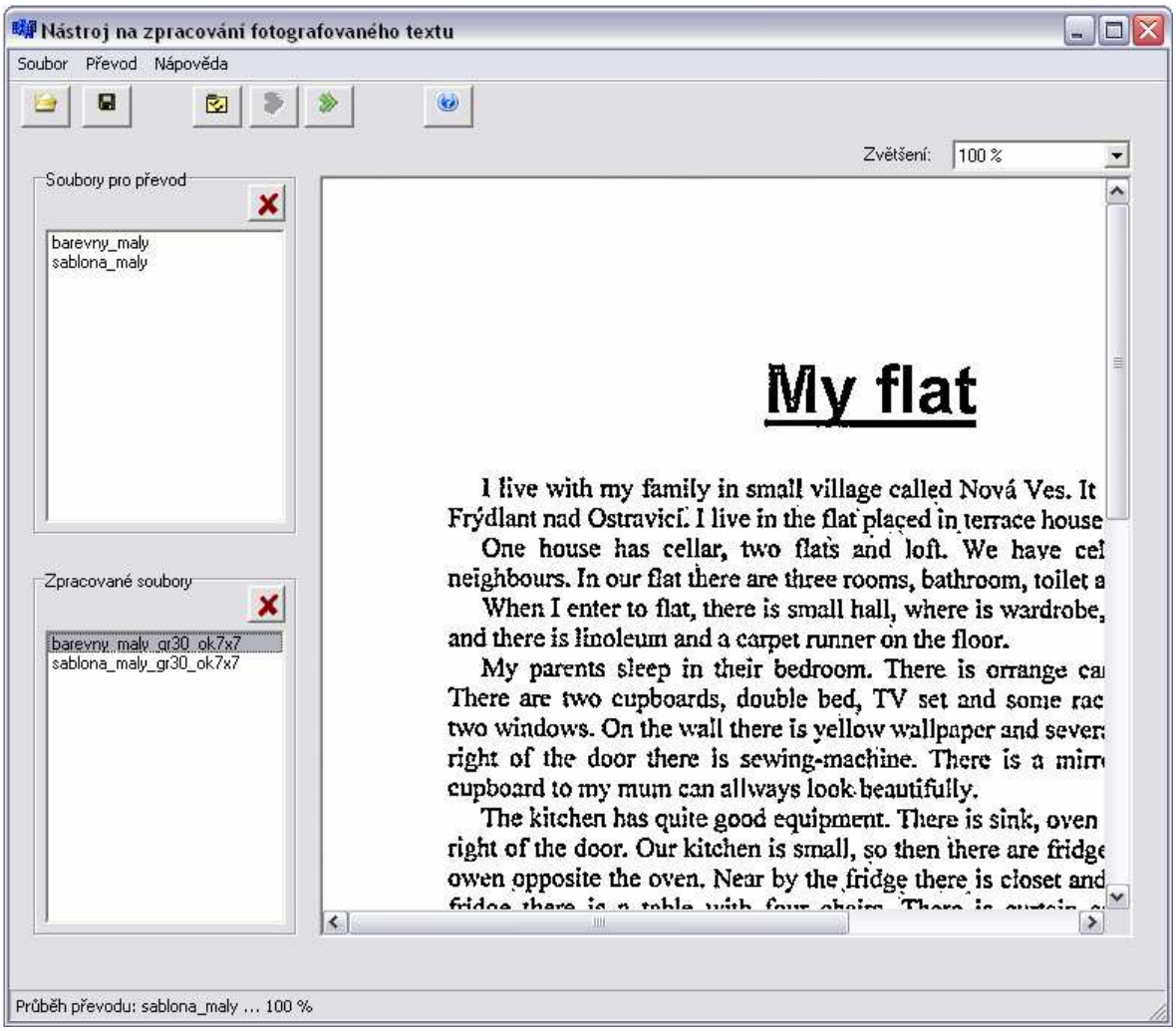
Pro samotné zobrazení obrázku slouží komponenta Image. Ta obsahuje obraz ve vlastnosti Picture. Komponenta umožňuje načítat obraz ze souboru pomocí funkce LoadFromFile(). Parametrem je jméno souboru s obrazem. Podporuje formáty bmp, jpg , ico, emf a wmf. Po načtení obrázku do komponenty již ovšem není možné kreslení. Aby mohla zobrazit další obrázek, musí jí být přiřazena kompletní nová bitmapa.

Ukládání obrázků do formátu je řešeno pomocí třídy TJPEGImage. Třída obsahuje instanční vlastnost Compression Quality, která udává kompresi obrázku při ukládání. Po určení této hodnoty v rozmezí 1-100% je nutno instanci třídy přiřadit obraz funkcí Assign(), která jako parametr požaduje bitmapu. Pro uložení obrázku z instance slouží metoda SaveToFile(), kde se opět předá jako parametr

jméno souboru. Pro tvorbu dalších prvků grafického rozhranní byly použity komponenty jako MainMenu, ScrollBox, ToolBar, ComboBox a další.

6 Popis programu

Aplikace pro svůj běh vyžaduje operační systém Windows 95 nebo vyšší. Ovládání je možné pomocí hlavního menu nebo nástrojové lišty. Grafické uživatelské rozhranní je vidět na následujícím obrázku:



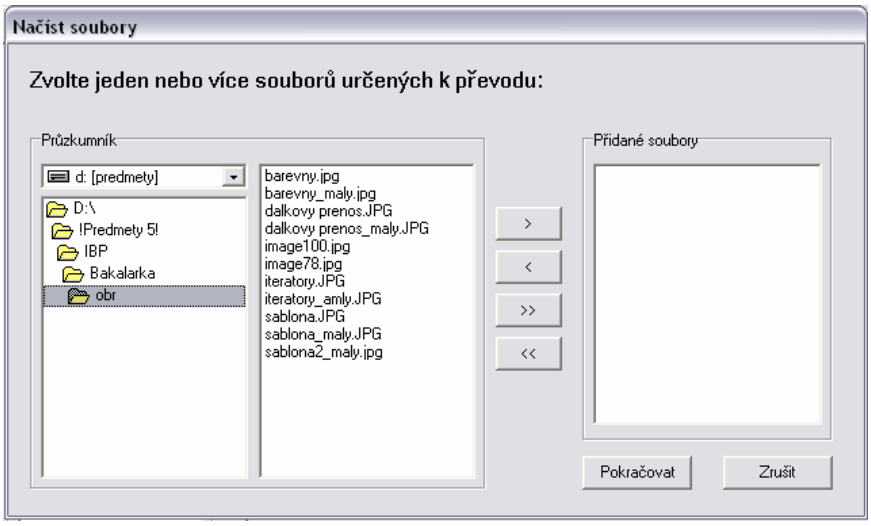
Obr. 6.1 - Hlavní okno aplikace

V následujícím textu si popíšeme strukturu hlavního menu.

6.1 Hlavní menu programu

6.1.1 Menu Soubor

Toto menu obsahuje nejzákladnější funkce programu, celkem pět položek. První volba tohoto menu má jméno **Otevřít.** Na následujícím obrázku je vidět okno, které se objeví po kliknutí na tuto volbu. Grafické rozhranní umožňuje přidat nebo odebrat jeden nebo více souborů pro zpracování. Adresářová struktura je po spuštění programu nastavena na aktuální složku aplikace. Po stisknutí tlačítka Pokračovat se zvolené soubory načtou do hlavního okna programu.

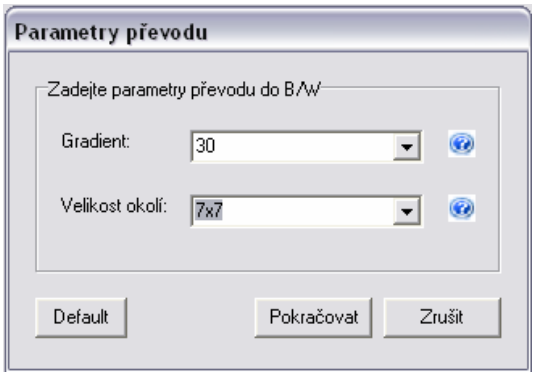


Obr. 6.2 - Dialog pro načtení souborů do programu

Další tři volby menu slouží pro uložení jednoho nebo více obrázků, které byly převedeny do černobílé podoby. Volba **Uložit vybraný** otevře dialogové okno, kde je možné zadat jméno souboru, pod nímž bude obraz uložen. Aby bylo možné použít tuto možnost, musí být v hlavním okně aplikace označen výsledný obraz uložení. Je také možno zvolit jeden z formátů pro uložení. Podporované formáty jsou BMP a JPEG. Pokud ukládáme soubory ve formátu JPEG, je použita kvalita komprese 99% a tudíž nepřijdeme o žádné informace obsažené v obrázku. Volba **Uložit vše** postupně otevírá dialogové okno pro uložení všech zpracovaných obrázků, přičemž pro každý obr. můžeme zadat vlastní jméno. Takovou možnost neposkytuje následující volba s názvem **Uložit vše auto.** Ta prochází postupně všechny zpracované obrázky a každý uloží s tím, že na konec jména souboru přidá koncovku "_bw". Ukládá se ve formátu jpeg. Poslední volba **Konec** slouží pro ukončení běhu programu.

6.1.2 Menu převod

V tomto menu se nalézají volby pro převod obrazu do dvoubarevné podoby. První možností jsou **Parametry převodu.** Po kliknutí se objeví okno, ve kterém můžeme zadat parametry převodu.



Obr. 6.3 - Okno pro nastavení parametrů

Parametr Gradient se předá proměnné max_min popsané v kapitole 5.2.1. Je možné zvolit jednu z přednastavených hodnot 20, 30, 40, 50, 75 a 100. Čím vyšší je kontrast mezi písmem a pozadím obrazu, tím vyšší hodnota by měla být zvolena. Po kliknutí na obrázek otazníku se zobrazí nápověda pro zadávání tohoto parametru. Parametr "Velikost okolí" byl rovněž popsán v kapitole 5.2.1. Přednastavené hodnoty jsou 5x5,7x7,9x9, ..., 21x21. Rovněž existuje nápověda pro volbu hodnoty.
Pro vrácení parametrů do původního nastavení slouží tlačítko Default. Druhou volbou tohoto menu (**Převést vše**) převedeme všechny načtené obrázky do dvoubarevné podoby. Poslední volba je **Převést vybraný**. Aby byla tato možnost přístupná, musí být v horním poli s načtenými soubory jeden zvolen. Po provedení převodu se objeví zpracované soubory v dolním okénku, přičemž každá položka obsahuje informaci o tom, jaká hodnota gradientu a velikosti okolí byla zvolena (např. "barevny_maly_gr30_ok7x7"). Tento doplněk slouží pro odlišení jednotlivých převodů.

6.1.3 Menu Nápověda

V tomto menu můžeme nalézt dvě položky. První je Nápověda, obsahuje jednoduchou nápovědu pro postup převodu obrázku do dvoubarevné podoby. Druhá volba, "O programu", obsahuje informace o programu.

6.2 Další součásti a funkce aplikace

Program obsahuje také nástrojovou lištu, kde jsou tlačítka s funkcemi menu. Jejich funkcionalita je (zleva): Otevřít, Uložit vybraný, Parametry převodu, Převést vybraný, Převést vše a Nápověda. V pravém horním rohu aplikace je rozbalovací nabídka pro zvětšení nebo zmenšení rozlišení při zobrazování výsledku. Na stavovém řádku se zobrazuje průběh převodu v procentech a jaký soubor je v danou chvíli zpracováván. Pokud máme načteny nějaké soubory určené pro zpracování, můžeme si zobrazit jeho vlastnosti kliknutím pravým tlačítkem myši na příslušnou položku v horním okénku. V tomto menu zároveň můžeme převést vybraný soubor. Při kliknutí pravým tlačítkem na položku v dolním okénku se zobrazí možnost, kterou můžeme příslušný obraz uložit do souboru.

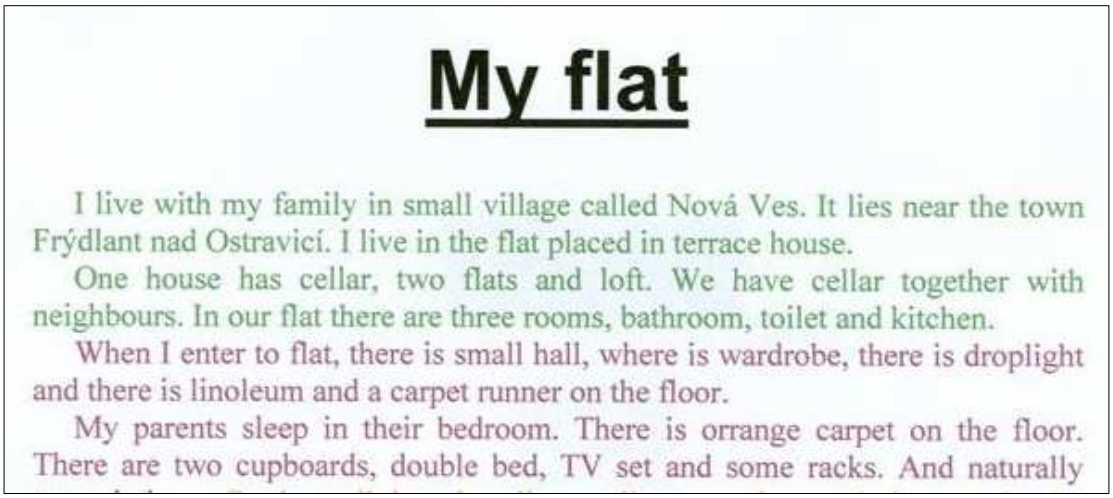
6.3 Návod k použití programu

- Načtěte jeden nebo více souborů do programu tlačítkem "Otevřít" nebo zvolte ekvivalentní volbu z rozbalovacího menu. Zvolené soubory jsou nyní zobrazeny v okénku s názvem "Soubory pro převod"
- Nyní můžete zadat parametry nebo ponechat jejich přednastavené hodnoty (doporučeno).
- Převeďte fotografovaný text do dvoubarevné podoby tak, že vyberete jeden ze souborů v horním okénku a stisknete tlačítko "Převést vybraný" nebo alternativní volbou "Převést vše", kdy jsou převedeny všechny soubory z horního okénka.
- Zobrazte výsledek převodu kliknutím na jeden ze souborů v dolním okénku, kam byly po převodu přidány
- Uložte výsledek do souboru jednou z voleb "Uložit vybraný","Uložit vše" nebo "Uložit vše auto"

7 Testování programu

Funkčnost aplikace byla testována na několika vzorcích dat. Tyto obrazy byly voleny tak, aby odpovídaly nejčastěji zpracovávaným typům. Data jsou nejčastěji získávána snímáním pomocí scanneru nebo kopírky. Zahrnuty byly obrázky s barevným písmem, s nehomogenním pozadím v odstínech šedi, s prudce se svažujícím odstínem pozadí a s velkými písmeny.
Následují náhledy na vzorky dat a komentář týkající se výsledku a možností nastavení parametrů převodu. Jsou zde zobrazeny pouze výřezy dat, celé obrazy jsou k nalezení na přiloženém CD.

7.1 Test 1


My flat

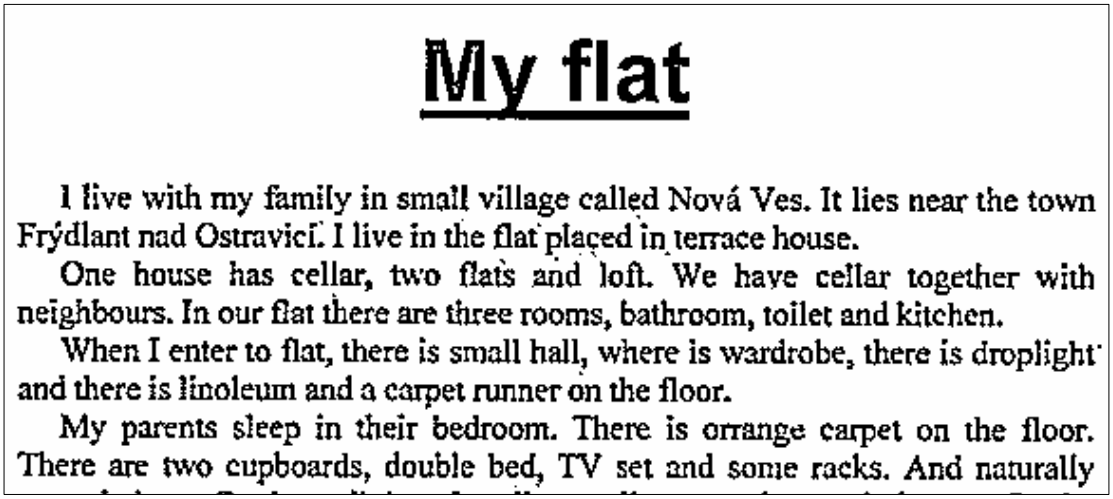
I live with my family in small village called Nová Ves. It lies near the town Frýdlant nad Ostravicí. I live in the flat placed in terrace house.

One house has cellar, two flats and loft. We have cellar together with neighbours. In our flat there are three rooms, bathroom, toilet and kitchen.

When I enter to flat, there is small hall, where is wardrobe, there is droplight and there is linoleum and a carpet runner on the floor.

My parents sleep in their bedroom. There is orrange carpet on the floor. There are two cupboards, double bed, TV set and some racks. And naturally

Obr. 7.1 - Originální obraz s barevným textem



My flat

I live with my family in small village called Nová Ves. It lies near the town Frýdlant nad Ostravicí. I live in the flat placed in terrace house.

One house has cellar, two flats and loft. We have cellar together with neighbours. In our flat there are three rooms, bathroom, toilet and kitchen.

When I enter to flat, there is small hall, where is wardrobe, there is droplight and there is linoleum and a carpet runner on the floor.

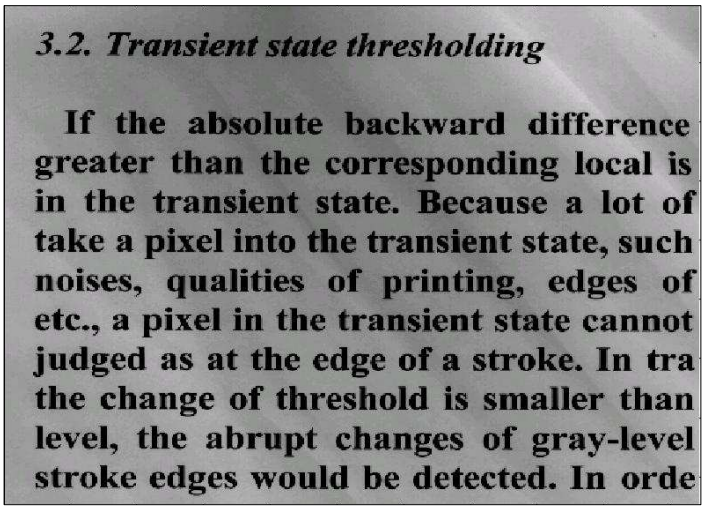
My parents sleep in their bedroom. There is orrange carpet on the floor. There are two cupboards, double bed, TV set and some racks. And naturally

Obr. 7.2 - Výsledek převodu obrazu s barevným textem

Test ukazuje převod obrazu s barevným textem a klasickým bílým pozadím. Tento obraz vznikl naskenováním papírové předlohy. Jeho rozměry v digitální podobě jsou 765x1053 px, ovšem na obrázku je vidět pouze jeho výřez pro lepší posouzení detailů. Převod celého obrazu trval 3,13 sekund v počítači s procesorem o taktu 1600MHz a 256MB operační paměti. Parametry byly nastaveny na implicitní hodnoty programu, tj. hodnota parametru Gradient byla 30 a velikost okolí 7x7 pixelů. V obraze je vidět jisté vady ve středové oblasti. Tyto nežádoucí objekty, které nejsou písmeny je možné z velké části eliminovat zvýšením hodnoty gradientu na 40. Originální obraz je uložen na přiloženém CD pod označením "7_1 barevne pismo.jpg", výsledek převodu je označen "7_2 barevne pismo.jpg".

7.2 Test 2

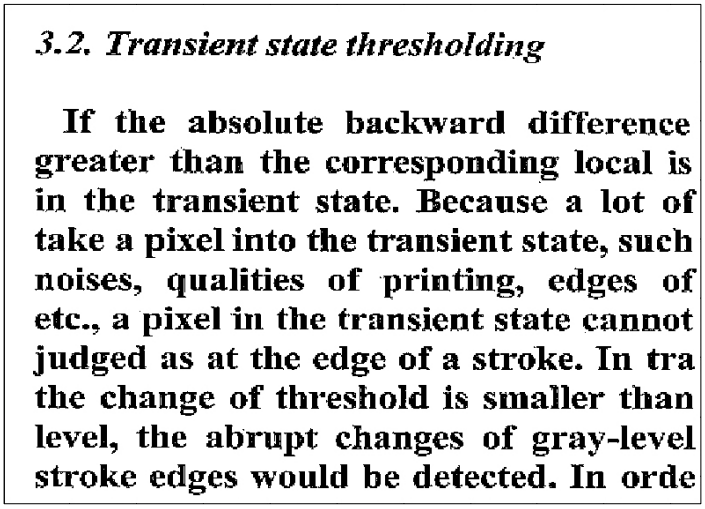
Dalším testovaným je obraz s textem na pozadí, které obsahuje nerovnoměrné změny osvětlení.



3.2. Transient state thresholding

If the absolute backward difference greater than the corresponding local is in the transient state. Because a lot of take a pixel into the transient state, such noises, qualities of printing, edges of etc., a pixel in the transient state cannot judged as at the edge of a stroke. In tra the change of threshold is smaller than level, the abrupt changes of gray-level stroke edges would be detected. In orde

Obr. 7.3 - Obraz s nerovnoměrným osvětlením pozadí



Obr. 7.4 - Výsledek převodu obrazu s nerovnoměrným osvětlením pozadí

Originální rozměry obrázku jsou 792x567 px. Převod trval 1,97 sekund. Pro eliminaci vad v obraze byla hodnota gradientu nastavena tentokrát na 75, což je druhá nejvyšší možnost z nabízených. Originální obraz je uložen na přiloženém CD pod označením "7_3 transient.jpg", výsledek převodu je označen "7_4 transient.jpg".

7.3 Test 3

Následující úloha testuje větší obrázek tak, jak by byl pravděpodobně nasnímán pomocí scanneru - černý text na bílém pozadí.

Obr. 7.5 - Výřez obrazu s větším rozlišením

Na obrázku 7.6 je vidět výsledek převodu s parametry Gradient = 100 a velikostí okolí 7x7. Originální obraz má rozměry 1692x2332 a převod trval 14,17 sekund. Takovýto výsledek však není zcela uspokojívý, protože oblasti uvnitř písmen nemají správnou barvu. Vylepšení můžeme dosáhnout zvětšením hodnoty okolí na 15x15 pixelů. Převod poté trvá 21,52 sekund a výsledek je vidět na obrázku 7.7

Obr. 7.6 - Výsledek převodu pro parametry Gradient=100, Velikost okolí = 7x7 px

Obr. 7.7 - Výsledek převodu pro parametry Gradient = 100, Velikost okolí = 15x15 px

7.4 Test 4

Poslední test naznačuje důležitost parametru Gradient. Obrazy jsou jen na přiloženém CD a to pod názvy "7_8 svazujici orig.jpg", "7_9 svazujici 20.jpg" a "7_10 svazujici 75.jpg". Obrázek 7_8 představuje originální vzorek dat, na kterém je vidět prudké svažování osvětlení pozadí. Tento jev vzniká například při špatném kopírování z papírové předlohy. Pokud při převodu nastavíme parametr Gradient na příliš vysokou hodnotu (např. 75), část písma se vytratí. Je to způsobeno tím, že rozdíly jasových hodnot písma a pozadí v těchto místech se liší jen málo a program tak nerozpozná samotné písmo. V takových případech je vhodné nastavit parametr na co nejnižší hodnotu, kterou je mezi nabízenými možnostmi číslo 20.

8 Optické rozpoznávání znaků (OCR)

8.1 Úvod do OCR

Optické rozpoznávání znaků, obvykle zkracováno jako OCR, zahrnuje počítačový software navržený pro převod obrazu se strojově tištěným textem (obvykle pořízeným pomocí scanneru) do podoby textu editovatelného pomocí standardního textového editoru. OCR je dnes předmětem výzkumu v oboru umělé inteligence a počítačového vidění.

Optické rozpoznávání (používající optické techniky jako zrcadla a čočky) a digitální rozpoznávání znaků bylo původně považováno za dva odlišné směry. Protože již existuje jen velmi málo aplikací používajících ryze optické metody, termín optické rozpoznávání znaků nyní pokrývá i oblast digitálního rozpoznávání znaků. Existují v zásadě dva přístupy pro řešení tohoto problému. Prvním z nich je rozpoznávání znaků v obraze pomocí učení se znaků ze šablon známých fontů. Druhým přístupem je inteligentní rozpoznávání znaků, které funguje na principu zkoumání charakteristických rysů každého alfanumerického a nealfanumerického znaku. Zkoumají se například otevřené oblasti uzavřené tvary nebo průniky linií.

8.2 Implementace

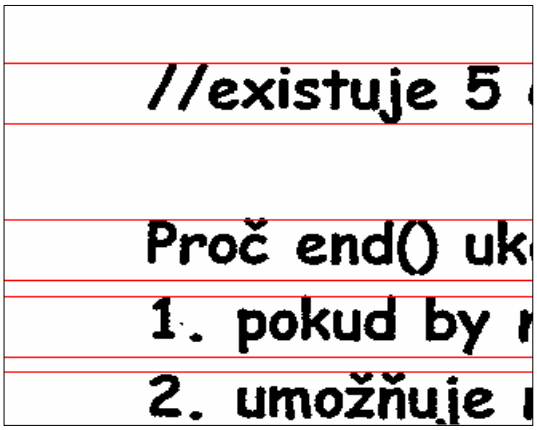
Cílem této práce bylo také pokusit se o vytvoření programové nadstavby provádějící optické rozpoznávání znaků. Je nutno hned na začátku uvést, že se nepodařilo vyvinout kvalitní nástroj, který by splňoval hlavní požadavek pro provádění tohoto převodu, tj. malou chybovost. Proto nebyl výsledek zahrnut do konečné podoby aplikace. Ovšem program provádějící převod do textové podoby byl vytvořen a je k nalezení na přiloženém CD ve složce OCR. V této složce jsou všechny zdrojové texty včetně zkompilovaného programu. Ten je představován souborem OCR.exe. Je možno vyzkoušet funkčnost na přiložených vzorcích dat, která se nacházejí v pod-složce "Vzorky dat". Ač není výsledek převodu obrazu do podoby textu příliš dobrý, je implementováno několik funkcí a datových struktur, jež by mohly být použity při vývoji kvalitního nástroje tohoto druhu. Tyto programové prvky jsou implementovány ve zdrojovém souboru Unit1.cpp v již zmíněné složce s názvem OCR.

První datovou strukturou, která stojí za zmínku je třída RadekPisma. Tato třída představuje jeden řádek písma v obraze a její instance může být později vložena do pole. Třída poskytuje několik instančních metod jako například nastav_y1() a nastav_y2(), které přebírají jeden argument typu celé číslo. Slouží pro nastavení horní a dolní hranice řádku. K nim existují ekvivalentní metody dej_y1() a dej_y2(). Od této třídy je odvozena další třída s názvem Pismeno pro jednotlivá písmena každého řádku.

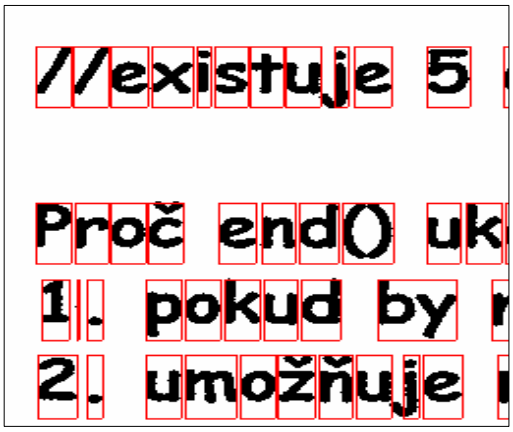
Zpracování obrazu pomocí OCR zahrnuje několik fází. Pokud již máme obraz v černobílé podobě, je nutno nejprve detekovat řádky písma v obraze. K tomu slouží funkce detekceRadku(), která zjistí horní a dolní hranice každého řádku a postupně je ukládá za použití třídy RadekPisma do vektoru řádků. Funkčnost programu v této fázi je vidět na obrázku 8.1. V programu je možno natočit plátno obrazu, tak aby linky co nejlépe kopírovaly dolní a horní hranici písma. To se používá v případě, že obraz je nasnímán pootočený. Pootočení do správné polohy vykonává funkce rotace(). Přebírá argument udávající úhel pootočení.

Ve chvíli, kdy máme detekovány jednotlivé řádky, přichází na řadu detekce jednotlivých písmen. Tuto akci provádí funkce detekcePismen(), která hranice každého písmene uloží do datové

struktury Pismeno a tuto strukturu následně do vektoru písmen. Program dovede detekovat písmena tak, jak je znázorněno na obrázku 8.2



Obr. 8.1 - Detekované řádky



Obr. 8.2 - Detekovaná písmena

Po detekování jednotlivých znaků již následuje samotné rozpoznávání. Pro tento program byla zvolena metoda učení z šablon. Jsou k dispozici tři šablony pro různá písma: Arial, CourierNew a TimesNewRoman. Jednotlivé znaky byly segmentovány mřížkou o rozměrech 7x7 polí způsobem znázorněným na obrázku 8.3.
Každé takovéto pole obsahuje skupinu pixelů a je vypočítána průměrná barevná hodnota v této oblasti. Tyto informace pro každé pole, každého znaku a pro každý druh písma byly uloženy do textových souborů pro další použití. Následné rozpoznávání znaků probíhá tak, že znaky obrazu pro zpracování jsou segmentovány stejným způsobem a jsou porovnávány s údaji uloženými v textových souborech. Jako správný znak se poté bere ten, který se nejméně liší od předlohy.

Obr. 8.3 - Mřížka znaku

9 Závěr

Cílem této práce bylo vytvořit aplikaci pro převod fotografovaného textu do černobílé (dvoubarevné) podoby, popř. s využitím optického rozpoznávání znaků do textového formátu. Pro každý z těchto úkolů byly prostudovány různé metody, které operace provádějí. Za účelem převodu obrazu do dvoubarevné podoby byl vybrán postup adaptivního lokálního prahování (kapitola 4.2.3) a pro rozpoznávání znaků metoda učení znaků z šablon (kapitola 8).

Řešení prvního z úkolů demonstruje vytvořená aplikace. S její pomocí byla provedena sada testů. Uživatel může v programu zadávat parametry převodu, které jsou závislé na zpracovávaném obraze. Po provedení převodu je možno výsledek uložit pro pozdější zpracování jiným programem. Parametry zadanými aplikaci je možno ovlivnit kvalitu výstupu, jenž vznikne převodem. Tyto parametry byly označeny jako Gradient a Velikost okolí a jejich význam plyne z použité metody. Hodnota prvního z nich závisí na kontrastu mezi písmeny a pozadím obrazu, druhý zase na velikost rozlišení vzorku a na velikosti jednotlivých písmen.

Poznatky z provedených testů by se daly shrnout následovně: Aby mohly být určeny přednastavené hodnoty parametrů, které lze v aplikaci nastavit, byly zjištěny nejvíce vyhovující hodnoty v určitém rozsahu. Pro gradient je to 20 až 100 a pro velikost okolí 5x5 až 21x21 pixelů. Ostatní hodnoty pro tento případ prakticky nemají význam, jelikož zvolené rozmezí postačuje. Doba převodu závisí na rozlišení obrazu a na zvolené velikosti okolí. Pokud jsou nastaveny připravené hodnoty parametrů a rozlišení obrazu je v řádu stovek pixelů v horizontálním i vertikálním směru, převod trvá jednotky sekund. Při rozlišení v řadu tisíců pixelů v obou směrech se doba prodlužuje na desítky sekund. Aplikace je při správně nastavených parametrech schopna převést velké množství různých typů obrazů, co se týče změn osvětlení, barevnosti textu a pozadí, a stínů v obraze obsažených. Použitelnost programu je tak na dobré úrovni.

Druhým z úkolů práce bylo pokusit se vytvořit aplikaci schopnou převodu dvoubarevného obrazu do textové podoby. Tento úkol se podařilo splnit jen částečně. Programová část provádějící úlohu nebyla zahrnuta do výsledné podoby hlavní aplikace, nýbrž vznikl samostatný program. Tento program dovede detekovat jednotlivé řádky a písmena v obraze a dovede také pootočit obraz v případě, že je špatně nasnímán. Samotný převod ovšem nepracuje s dostatečnou přesností.

Hlavní výsledný program tedy produkuje obraz ve dvoubarevné podobě, který je vhodný pro následné optické rozpoznávání znaků. Tímto směrem by se měl ubírat i další vývoj projektu, ať už s využitím funkcí pro detekci řádků a písmen nebo bez nich.

Literatura

[1] Bob Fisher, Simon Perkins, Ashley Walker, Erik Wolfart
Hepermedia Image Processing Reference, 1994
http://www.cee.hw.ac.uk/hipr/html/hipr_top.html

[2] Martin Šmat
Metody digitální holografické interferometrie
Diplomová práce, Plzeň, 2002

[3] Jiří Žára
Počítačová grafika - principy a algoritmy, kapitoly o rozptylování a barvách
Grada, 1992

Seznam příloh

Příloha 1. Uživatelská příručka k programu

Příloha 2. CD se zdrojovými soubory projektu, testovacími daty atd.

Adresářová struktura CD:

1. OCR\ - obsahuje zdrojové texty a spustitelný projekt programové nadstavby pro OCR
2. Spustitelný program\ - obsahuje spustitelný soubor projektu s příponou ".exe"
3. Technická zpráva\ - obsahuje text technické zprávy v elektronické podobě (ve formátech DOC a PDF, dále obsahuje uživatelskou příručku pro práci s programem
4. Testovací data BP\ - obsahuje obrázky, které byly zpracovány při testech
5. Zdrojové soubory\ - všechny zdrojové soubory hlavní aplikace včetně obrázků pro ikony

Příloha 1

Uživatelská příručka k programu

Postup převodu obrazu do dvoubarevné podoby:

- Načtěte jeden nebo více souborů do programu tlačítkem "Otevřít" nebo zvolte ekvivalentní volbu z rozbalovacího menu. Objeví se okno, ve kterém můžete pohodlně vybrat soubory pro zpracování. Zvolené soubory jsou nyní zobrazeny v okénku s názvem "Soubory pro převod"
- Nyní můžete zadat parametry nebo ponechat jejich přednastavené hodnoty (doporučeno).
- Převeďte fotografovaný text do dvoubarevné podoby tak, že vyberete jeden ze souborů v horním okénku a stisknete tlačítko "Převést vybraný" nebo alternativní volbou "Převést vše", kdy jsou převedeny všechny soubory z horního okénka. Vybraný obrázek je možné převést i tak, že kliknete pravým tlačítkem myši na příslušnou položku a vyberete volbu "Převést do B/W". V tomto menu si zároveň můžete zobrazit základní informace o souboru.
- Po provedení převodu jsou zpracované položky přidány do dolního okénka. Výsledek převodu zobrazíte kliknutím na jeden ze souborů v dolním okénku, kam byly po převodu přidány. Můžete si změnit velikost zobrazení v pravém horním rohu aplikace.
- Uložte výsledek do souboru jednou z voleb "Uložit vybraný","Uložit vše" nebo "Uložit vše auto". Aby byla přístupná volba "Uložit vybraný", musí být v dolním okénku zvolen jeden ze souborů. Při volbě "Uložit vše" můžete postupně zadat jména pro každá ze souborů v dolním okénku. Volba "Uložit vše auto" uloží všechny soubory tak, že k jejich názvu přidá koncovku _bw. Formátem pro uložení je JPEG. Jednotlivé obrázky je možné uložit také tak, že kliknete pravým tlačítkem myši na položku a v menu zvolíte "Uložit soubor".