VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

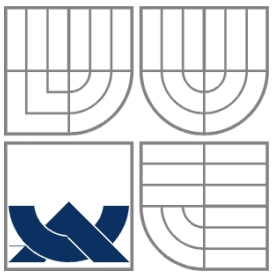
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ NA ZPRACOVÁNÍ FOTOGRAFOVANÉHO TEXTU

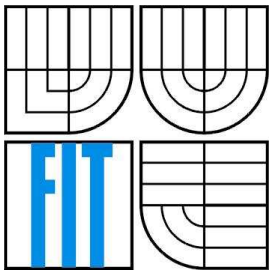
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ROMAN HOTAŘ
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ NA ZPRACOVÁNÍ FOTOGRAFOVANÉHO TEXTU

APPLICATION FOR PROCESSING OF A PHOTOGRAPHED TEXT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN HOTAŘ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. LUKÁŠ GRULICH

BRNO 2008

Abstrakt

Tato práce popisuje metody převodu barevného obrázku s textem (barevné přechody, různé odstíny stíny atd.) do černobílé podoby. Zaměřuje se převážně na adaptivní prahování. Uvádí použité datové struktury a popisuje prostření výsledné aplikace. Závěrem jsou uvedeny výsledky několika testů funkčnosti převodu.

Klíčová slova

Adaptivní prahování, globální prahování, histogram, Wall’s algoritmus, OCR .

Abstract

This work describe conversion methods full-colour picture with text (colour transition, various shades etc.) in black and white form. It focus mainly on adaptive thresholding. Mention used data structures and describe interface of resultant application. The thesis describe at the conclusion the results of several functionality tests.

Keywords

Adaptive thresholding,, global thresholding, histogram, Wall’s algorithm, OCR .

Citace

Hotař Roman: Nástroj na zpracování fotografovaného textu. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Zadání bakalářské práce/6345/2007/xhotar04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Hotař Roman**
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 nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), 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 2007
Datum odevzdání: 14. května 2008

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
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Roman Hotař**
Id studenta: 78988
Bytem: Uherčice 108, 671 07 Uherčice u Znojma
Narozen: 10. 08. 1986, Znojmo
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Nástroj na zpracování fotografovaného textu
Vedoucí/školitel VŠKP: Grulich Lukáš, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

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.

……
Roman Hotař
9. 5. 2008

Poděkování

Tímto bych chtěl poděkovat Ing. Lukáši Grulichovi za poskytnutí konzultací a důležitých informací pro tvorbu projektu.

© Roman Hotař, 2008.
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

Úvod

V dnešní době, kdy se počítače pro většinu lidí staly součástí každodenního života, již snad každý pochopil výhody elektronického textu. Jako nejvýznamnější bych označil výhody, kterými jsou jednoduchá změnitelnost, snadné kopírování, nezabírá fyzicky skoro žádné místo a lze v něm pohodlně a rychle cokoli vyhledat.

Jelikož dříve počítače nebyly, je stále většina dokumentů a knih tištěných a žádnou elektronickou verzi nemají, je proto potřeba řešit problém, jak tyto texty převést co nejrychleji a nejlevněji do elektronické podoby bez větších zásahů člověka.

Pro tento účel byly vyvinuty speciální OCR programy, z anglického Optical Character Recognition což lze přeložit do češtiny jako „optické rozpoznávání znaků“. První OCR systém byl vyvinut v USA roku 1951 kryptografem Davidem H. Shepardem a jeho přítelem Harvey Cookem, pod názvem Gismo. Od té doby bylo vyvinuto mnoho komerčních i nekomerčních OCR systémů s různou účinností. V dnešní době je rozpoznávání tištěného textu pokládáno za téměř vyřešený problém, systémy dosahují přesnosti 99%, což pro většinu užití postačuje. Rozpoznávání psaného textu a textu psaného kurzívou je však stále předmětem aktivního výzkumu.

Tyto rozpoznávací systémy mají zastoupení na poštách, kde slouží k automatickému třídění zásilek pomocí směrovacích čísel nebo adresy, dále pak v bankovnictví pro zpracování šeků, ve vzdělávacích institucích např. při vyhodnocování testů atd.

Proces převodu tištěného textu do elektronické podoby lze jednoduše popsat třemi kroky, kterými jsou naskenování stránky s textem do počítače a uložení do souboru, dále je tento soubor převeden do černobílé podoby, v poslední fázi je obrázek analyzován a jsou v něm rozpoznávány jednotlivé znaky.

Chceme-li dosáhnout vysoké účinnosti převodu musíme se zaměřit na bod číslo dvě, kterým je převod obrázku do černobílé podoby, protože pokud zvolíme nesprávnou metodu převodu, text ve výsledném černobílém obrázku bude znehodnocen, a tím již znemožníme správné rozpoznání znaků v následující fázi zpracování textu.

Tato práce podrobněji popíše problém převodu fotografovaného textu do černobílé podoby.

1 Definice základních pojmů

V této kapitole budou definovány základní pojmy spojené s binární reprezentací obrazu a jeho zpracováním, základní znalost těchto pojmů je důležitá k porozumění problematice tématu.

1.1 Digitální obraz

Skutečný reálný obraz lze zachytit a převést do digitální podoby např. pomocí skeneru, fotoaparátu, nebo kamery. Výsledný digitální obraz, který po převodu získáme, je reprezentován funkcí f(x,y) dvou proměnných, jimiž jsou souřadnice bodů v obraze. Funkční hodnotou f(x,y) je hodnota barevných složek bodu na souřadnicích x, y.

Binární reprezentace obrazu je uložena do matice, jejíž rozměry odpovídají rozměrům obrazu a jedna buňka matice se je nazývána pixel.

1.2 Barva

V předchozí podkapitole byl uveden termín barevné složky, který je pro další práci podstatný pro pochopení, proto zde bude stručný popis reprezentace barev v digitálním obraze.

Každý pixel matice obrázku má určitou barvu, tato barva se tvoří aditivním mícháním barevných složek (obr.1). Existuje několik barevných modelů např. RGB, HVS, CMY, tyto modely se liší podle počtu základních barevných složek a podle jejich barev, my použijeme RGB model.

Název RGB modelu je odvozen od anglických jmen jeho základních barevných složek červená, zelená, modrá (red, green, blue). Mícháním těchto složek získáme ostatní barvy možného barevného spektra.

Celkové množství barev v barevném spektru je závislé na tom, jakou nejvyšší hodnotu může mít každá barevná složka, to znamená kolik bitů je použito k uložení hodnoty složek barvy, někdy je používán termín bitová hloubka. Pro náš účel postačí 8 bitové číslo pro každou barevnou složku, jelikož máme tří barevné složky, dostaneme 24 bitový barevný prostor.

Podle hodnoty barevných složek můžeme rozlišit 3 typy bodů, barevné, šedé a černobílé.

- Barevné body – složky barev, takovýchto bodů mohou nabývat jakékoliv hodnoty v číselném intervalu 0 až 255 včetně. Jinak řečeno tyto body mohou mít jakoukoliv barvu ze všech možných barev RGB prostoru.

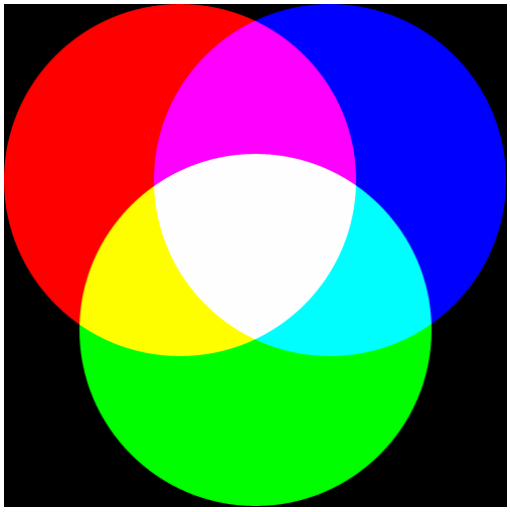
- Šedé body – barevné složky mohou nabývat hodnot 0 až 255 včetně, ovšem všechny tři složky musí mít stejnou hodnotu. Body se jeví jako šedé a jejich jas záleží na hodnotě

uložené v barevných složkách, krajní hodnoty jsou hodnota 0 pro černou a 255 pro bílou barvu. Barevný bod se převede na šedý následujícím vzorcem

$$I = 0{,}299\,R + 0{,}587\,G + 0{,}114\,B$$

Výsledné číslo I je intenzita šedi, nebo-li to číslo, které se přiřadí do všech barevných složek. Proměnné R, G, B zastupují hodnoty barevných složek červená, zelená, modrá.

- Černobílé body – černobílé body jsou podmnožinou množiny šedých bodů, s tím, že do této množiny patří pouze body černé (hodnota šedi 0) a bílé (hodnota šedi 255). Převést body do této množiny lze pouze pokud jsou již v odstínech šedi. Převod se nazývá prahování a jeho princip bude popsán v dalších kapitolách.



[obr. 1] Aditivní skládání barev prostoru RGB

2 Prahování

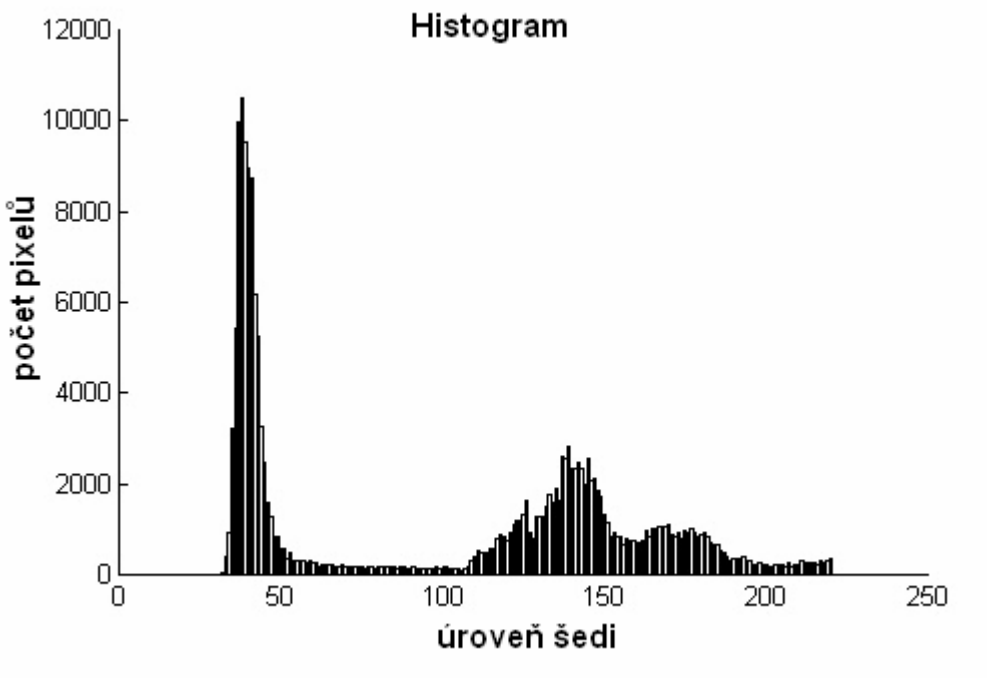
Jak již bylo napsáno výše, jedná se o převod, který je třeba provést, pokud chceme získat z šedého obrázku černobílý. Převod pracuje s hodnotami stupňů šedi, proto by měl být obrázek před prahováním převeden z barevného na šedý. Algoritmus převodu vychází z toho, že existuje hodnota T, které se říká práh, a s touto hodnotou se porovnává hodnota v jedné z barevných složek tzn. hodnota šedi. Je-li hodnota šedi menší, nebo rovna hodnotě prahu, zapíše se do všech barevných složek prahovaného bodu obrázku minimální hodnota (v našem případě 0), bod byl klasifikován jako černý. V opačném případě je přiřazena složkám hodnota maximální (pro náš případ je to hodnota 255), bod byl klasifikován jako bílý. Takto vznikne obraz pouze s dvěma typy bodů černými a bílými.

Porovnání hodnot bodů obrazu pomocí prahu je jednoznačné, rozdíly se projeví při prahování obrazu různými hodnotami prahu. Při jeho špatném nastavení se mohou ztratit důležité informace o původním obsahu obrazu.

Prahování se dále dělí do dvou hlavních skupin, kterými jsou Globální a Adaptivní prahování. Tyto dva přístupy se odlišují přístupem k obrázku při prahování T a jsou popsány v dalších kapitolách.

2.1 Globální prahování

Hodnota T je pro celý obrázek stejná. Tento způsob převodu se hodí pro obrázky s uniformním osvětlením, kde lze správným určením prahu dosáhnout dobrých výsledků. Body obrazů, na něž lze použít globální prahování, můžeme zařadit do dvou tříd, a to do třídy tmavých bodů, v našem případě text, a do třídy světlých bodů, v našem případě pozadí textu. Díky tomu jejich histogram vypadá podobně jako na obrázku (obr.2), je zde jasně vidět, že body tvoří dva shluky, kde shluk blíže nule na vodorovné ose představuje tmavé body čili text a shluk dále od nuly představuje světlé body čili pozadí. Práh je vhodné zvolit tak, aby se co nejpřesněji od sebe oddělily tyto dvě skupiny bodů.
Globální prahování je nevhodné pro obrázky, které mají různorodé osvětlení, velké barevné přechody nebo obrazový šum. V takovýchto případech nelze tento přístup použít, protože bychom nedosáhli kvalitních výsledků převodu.

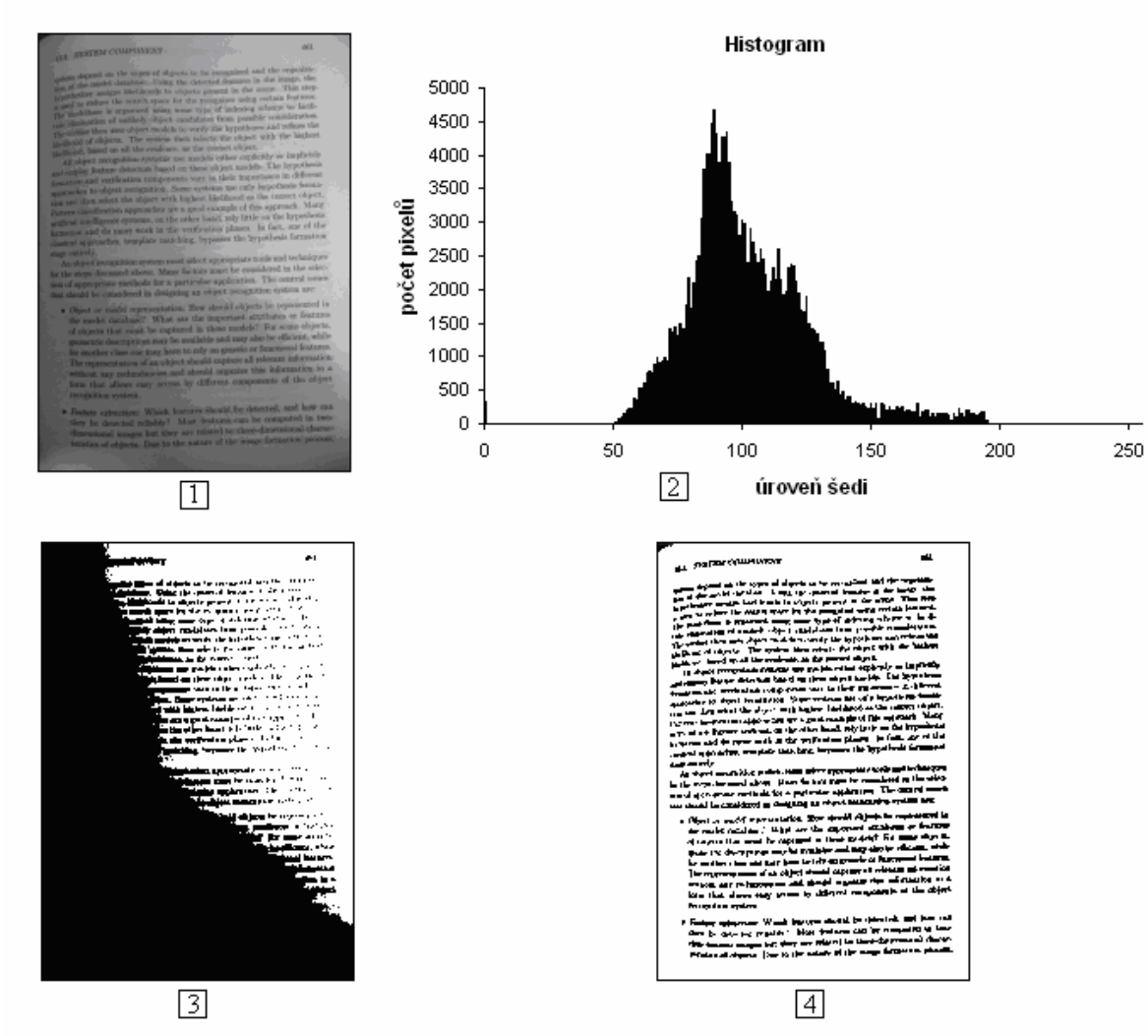


[obr. 2] Histogram obrázku vhodného pro globální prahování.

2.2 Adaptivní prahování

Hlavní myšlenkou adaptivního prahování je rozdělit obrázek do menších částí a pro každou použít takový práh, aby bylo dosaženo co nejlepšího výsledného obrazu. Adaptivní prahování vychází z předpokladu, že obrázek, který nemá stejnoměrné osvětlení, je možno rozdělit na části stejnoměrně osvětlené, a pro každou z těchto částí potom vypočítat vhodný práh. Vypočítaný práh je použít pro prahování konkrétní části obrázku.

Tímto způsobem je algoritmus schopen zpracovat i obrázky, které nelze převést globálním prahováním, jak je zobrazeno na obrázku (obr. 3).



[obr. 3] Porovnání globálního a adaptivního prahování: **1** originální obrázek, **2** histogram originálního obrázku, **3** originál převedený globálním prahováním T = 135, **4** originál převedený adaptivním prahováním

Je mnoho metod adaptivního prahování, většina z nich se liší především výpočtem optimálního prahu, patří mezi ně mimo jiné algoritmus Otsu, lokální prahování, Wall algoritmus, tyto metody lze použít za účelem oddělení textu od pozadí a budou dále podrobněji rozepsány. Existují také další metody jako metoda Chow a Kanenko, Tsai's prahování, Kapur et al's prahování, Prager's prahování a mnoho dalších.

2.2.1 Algoritmus Otsu (clustering)

Tento algoritmus předpokládá, že obrázek obsahuje dvě oddělitelné třídy (popředí a pozadí). Třídy jsou někdy nazývány také clustery. Algoritmus počítá optimální práh oddělující tyto dvě třídy tak, aby minimalizoval počet pixelů, které jsou nesprávně klasifikovány do jiné třídy. Výpočet vychází ze statistických výpočtů založených na relativním histogramu. Označme rozsah úrovní intenzit jako <0, N-1> a určeme, že dolní index B bude pro hodnoty pozadí a dolní index O pro hodnoty popředí. Nyní lze uvést výpočet variace uvnitř třídy $\sigma^2_{Within}(T)$, definovaný jako váhový součet variací

$$\sigma^2_{Within}(T) = n_B(T)\sigma^2_B(T) + n_O(T)\sigma^2_O(T) \tag{2.1}$$

kde $\sigma^2_B(T)$ je variace pixelů v třídě pozadí a $\sigma^2_O(T)$ je variace pixelů v třídě popředí. Dále platí vztahy

$$n_B(T) = \sum_{i=0}^{T-1} p(i) \tag{2.2}$$

$$n_O(T) = \sum_{i=T}^{N-1} p(i) \tag{2.3}$$

kde $p(i)$ reprezentuje i-tou hodnotu v relativním histogramu.

Výpočet této variace v třídě pro každou ze dvou tříd a pro každý možný práh je velmi časově náročné, proto je použit jednodušší způsob. Pokud odečteme variaci uvnitř třídy od celkové variace kombinovaného rozdělení získáme mezitřídní variaci

$$\begin{aligned}\sigma^2_{Between}(T) &= \sigma^2 - \sigma^2_{Within}(T) \\ &= n_B(T)[\mu_B(T) - \mu]^2 + n_O(T)[\mu_O(T) - \mu]^2\end{aligned} \tag{2.4}$$

kde σ^2 je kombinovaná variace a μ představuje kombinovanou střední hodnotu. Poznamenejme, že mezitřídní variace odpovídá váhové variaci střední hodnoty třídy od celkové střední hodnoty. Dosazením výrazu

$$\mu = n_B(T)\mu_B(T) + n_O(T)\mu_O(T) \tag{2.5}$$

a zjednodušením získáme vzorec

$$\sigma^2_{Between}(T) = n_B(T)n_O(T)[\mu_B(T) - \mu_O(T)]^2 \tag{2.6}$$

Pro každý potenciální práh provedeme následující operace:

1. Klasifikujeme pixely do dvou tříd podle prahu
2. Vypočteme střední hodnotu každé třídy
3. Umocníme rozdíl středních hodnot
4. Vynásobíme rozdíl počtem pixelů v jedné a v druhé třídě

Vypočítaný výsledná práh nyní závisí pouze na rozdílu mezi středními hodnotami dvou tříd, tím se vyhneme počítání rozdílů mezi jednotlivými intenzitami a střední hodnotou tříd. Optimální práh je ten, který maximalizuje mezitřídní variaci nebo naopak minimalizuje variance uvnitř třídy.

Pro zkrácení výpočtu lze výpočet vyjádřit rekurentně

$$n_B(T+1) = n_B(T) + n_T \tag{2.7}$$

$$n_O(T+1) = n_O(T) - n_T \tag{2.8}$$

$$\mu_B(T+1) = \frac{\mu_B(T)n_B(T) + n_T T}{n_B(T+1)} \tag{2.9}$$

$$\mu_O(T+1) = \frac{\mu_O(T)n_O(T) + n_T T}{n_O(T+1)} \tag{2.10}$$

kde n_T reprezentuje počet pixelů, které byly přeřazeny z jedné třídy do druhé při předchozím zvýšení hodnoty prahu.

Jak lze z předchozích vzorců poznat jedná se o výpočet poměrně složitý, ovšem na druhou stranu je tato metoda komplexní a můžeme ji používat také pro jiné obrázky než jen pro fotografovaný text.

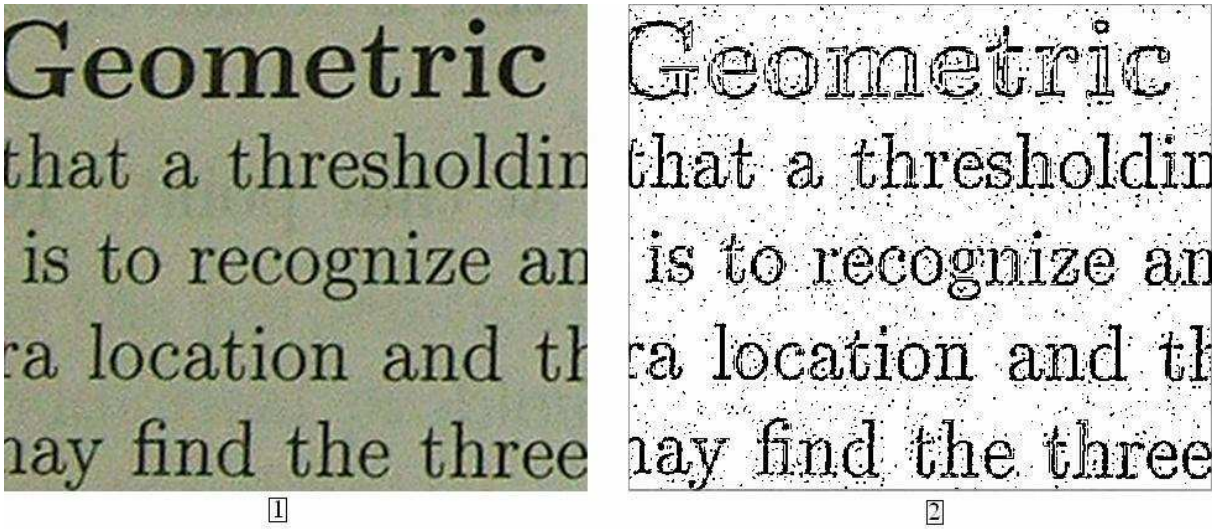
2.2.2 Lokální prahování

Metoda lokálního prahování dělí dokument na stejně velké části (většinou čtverce) a pro každou z těchto částí je určen vlastní práh, který je potom použit k převodu všech pixelů v dané části do černobílé podoby. Velikost těchto částí ovlivňuje kvalitu převodu, měla by mít takovou hodnotu, aby zahrnula jak body popředí, tak body pozadí. Pokud zvolíme malou hodnotu okolí, nebude do výpočtu prahu zahrnuta dostatečně kvalitní množina bodů a dojde k určení nesprávné hodnoty prahu, to se negativně projeví na výsledném černobílém obrázku. Naopak rozdělením obrázku do příliš velkých částí, nebude část uniformě osvětlena a nastane stejný problém jako při převodu globálním prahováním. Správně nastavená velikost okolí by měla mít přibližně stejnou hodnotu jako velikost fontu rozpoznávaných znaků, můžeme však nastavit vyšší hodnotu, čímž dosáhneme toho, že bude do výpočtu zahrnuta kvalitnější množina bodů a výsledek bude přijatelnější. Nesmíme však použít příliš velké okolí.

Práh pro dílčí část obrázku lze vypočítat třemi způsoby: jako průměr T = (součet hodnot)/(počet hodnot), jako medián *T = střední hodnota z intenzit bodů v okolí*, nebo mean

$T = \frac{\max - \min}{2}$, kde *max* je maximální hodnota a *min* minimální hodnota bodů z okolí. Každý z těchto způsobů se hodí pro jiný typ obrázku.

Lokální prahování je určeno primárně pro zpracování obrázků s textem, v nichž existují různé barevné přechody, stíny nebo obrazový šum, a proto tyto obrázky nemůžeme převádět globálním prahováním. Problém nastává, pokud je v textu vložen obrázek, nebo jsou v něm jiné grafické prvky, rovněž tehdy, pokud obrázek obsahuje velikostně velmi odlišné druhy textu. V takovémto případě dojde k tomu, že v některých oblastech obrázku, které budou obsahovat pouze popředí nebo pozadí, dojde ke špatnému určení prahu a oblast se převede na bílou barvu. Potom vznikají výsledky jako v obr.4, kde se vytvořila bílá místa ve velkých písmenech nadpisu. Tomuto jevu můžeme předejít zvolením hodnoty okolí odpovídající hodnotě vhodné pro největší písmena v textu, ovšem ne příliš velké, aby výsledné okolí stále mělo uniformní osvětlení. Grafické prvky v textu lze zpracovat zvlášť, odděleně od textu .



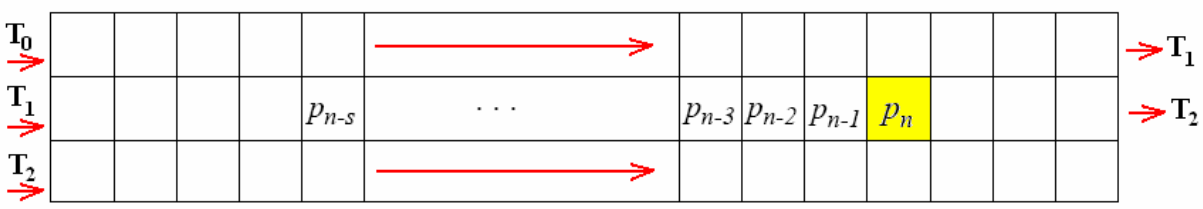
[obr. 4] **1** originální obrázek, **2** originál prahovaný malým okolím

2.2.3 Adaptivní prahování založené na Wall's algoritmu

Wall's algoritmus je technika výpočtu proměnného prahu při průchodu obrázkem, vyvinul jej R. J. Wall. Tohoto algoritmu je využíváno pro adaptivní prahování. Protože lokální prahování je časově náročné kvůli vícenásobnému průchodu obrazu, nelze ho použít v časově náročných aplikacích, jako například při převodu přímo v hardwaru scanneru. Tento algoritmus byl vyvinut hledáním rychlejšího a jednoduššího převodu než doposud známé metody, založené na několika průchodech obrázku, které jsou zbytečně komplexní.

Základní myšlenkou je průchod po řádcích a postupný výpočet průměrného prahu *s* posledních pixelů obrázku. Pokud je hodnota pixelu významně menší než vypočítaný průměr, je tento pixel převeden na černý, jinak je pixel bílý. K převedení je tedy nutný pouze jeden průchod obrázkem.

Tento způsob výpočtu prahu přistupuje k obrázku, jako by byl pouze jeden řádek pixelů, který se vytvoří průchodem obrázku zleva doprava shora dolů. Jak algoritmus prochází obrázkem počítá z posledních *s* pixelů (lze nazvat také okolím bodu) průměr a ten je použit jako hodnota prahu prahovaného pixelu p_n (obr. 5). Takovýto přístup je poměrně rychlejší než předchozí algoritmy, nelze jej ovšem použít bez větších úprav, protože dochází k několika problémům při převodu, jako např. chyby při přechodu na nový řádek.



[obr.5] Ukázka průchodu obrázkem, p_n je prahovaný pixel, *s* je počet průměrovaných bodů

Bereme-li funkci $f_S(n)$ jako průměr posledních *s* pixelů v bodě *n*, platí pro její výpočet vztah

$$f_S(n) = \sum_{i=0}^{s-1} p_{n-1} \tag{2.11}$$

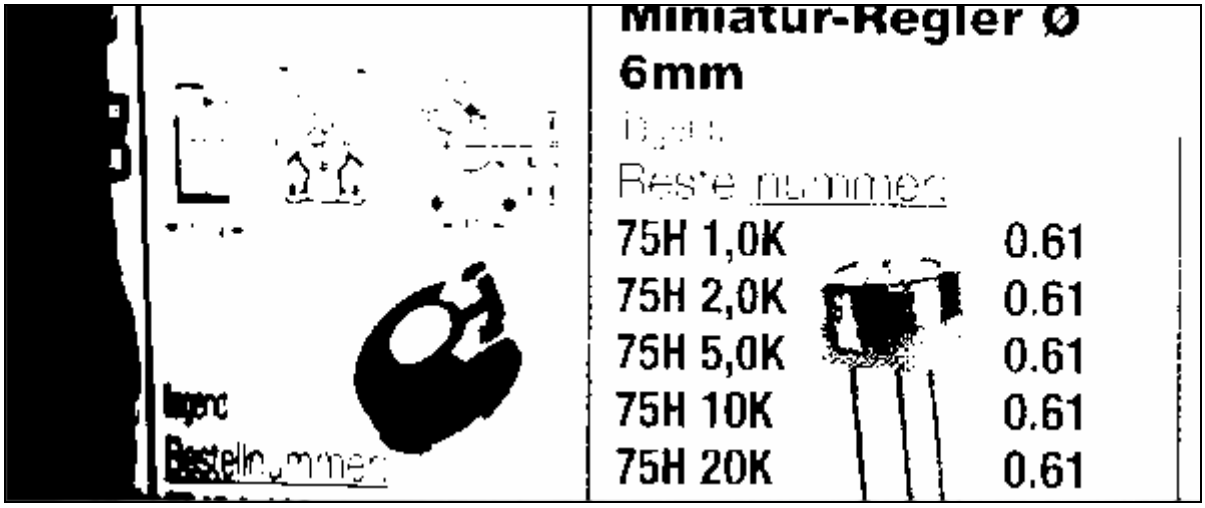
a vzorec pro výpočet výsledné hodnoty převedeného n-tého pixelu $T(n)$ lze zapsat jako

$$T(n) = \begin{cases} 255 & pokud \quad p_n < \left(\frac{f_s(n)}{s}\right)\left(\frac{100-t}{100}\right) \\ 0 & jinak \end{cases} \tag{2.12}$$

Výsledná hodnota pixelu $T(n)$ (0 černá, 255 bílá) se zde určuje podle toho jestli má prahovaný pixel p_n o *t* procent jinou hodnotu jasu než průměr jasu posledních *s* pixelů.

Tento neupravený původní algoritmus nelze použít pro všechny typy obrázků, protože nedosahuje potřebných výsledků. Při prahování obrázku 6 byl získán obrázek 7.

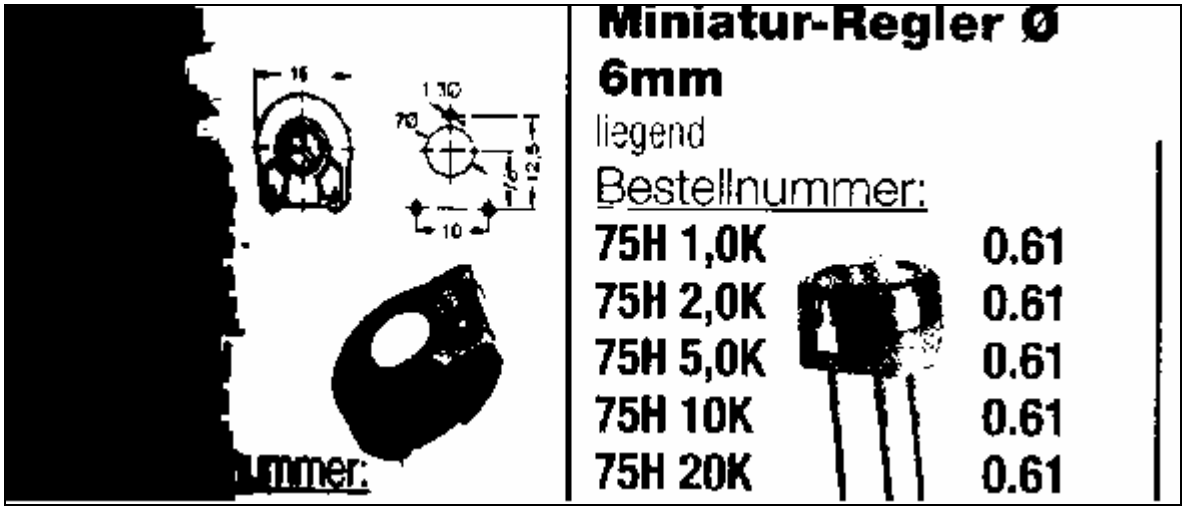
[obr.6] původní obrázek



[obr.7] prahovaný obrázek směrem zleva doprava s = 100, t = 10%

Z výsledného obrázku 7 můžeme poznat některé z nedostatků algoritmu, které je třeba napravit. Například ten, že levý okraj obrázku je černý, i když podle originálu by být neměl. To je způsobeno přechodem z jednoho řádku na další, kdy vzniká skokový přechod mezi intenzitami s velkým rozdílem, a jelikož probíhá tento přechod ze světlé části (pravé) do tmavé (levé), je vypočítaný práh příliš vysoký na to, aby mohl správně oddělit malý rozdíl intenzit v tmavé části. Pokud bychom obrázek prahovali po sloupcích, dosáhli bychom zaručeně lepšího výsledku díky tomu, že by měl obrázek změnu osvětlení kolmou na směr průchodu.

Metoda je méně citlivá na tenké vodorovné čáry, které z obrázku skoro zmizely, v našem případě je to levé schéma s kótami součástky, nebo podtržení slova Bestellnummer. Díky tomu, že metoda prochází obrázek směrem zleva doprava, je závislá na směru osvětlení na obrázku, při průchodu zprava doleva dostáváme jiný, v tomto případě o něco lepší výsledek viz. obr. 8.



[obr.8] prahovaný obrázek směrem zprava doleva s = 100, t = 10%

Výpočet průměru posledních *s* pixelů pro každy pixel může být zdlouhavý, rychlejší způsob jak počítat průměr je odečíst z jeho hodnoty část *1/s* a přičíst hodnotu dalšího pixelu. Tento průměr je označen $g_S(n)$ a platí pro něj vztah

$$\begin{aligned} g_S(n) &= g_S(n-1) - \frac{g_S(n-1)}{s} + p_n \\ &= p_n + (1-\frac{1}{s})\, g_s(n-1) \\ &= p_n + (1-\frac{1}{s})\, p_{n-1} + (1-\frac{1}{s})^2\, p_{n-2} \cdots \\ &= \sum_{i=0}^{n} (1-\frac{1}{s})^i\, p_{n-i} \end{aligned} \qquad (2.13)$$

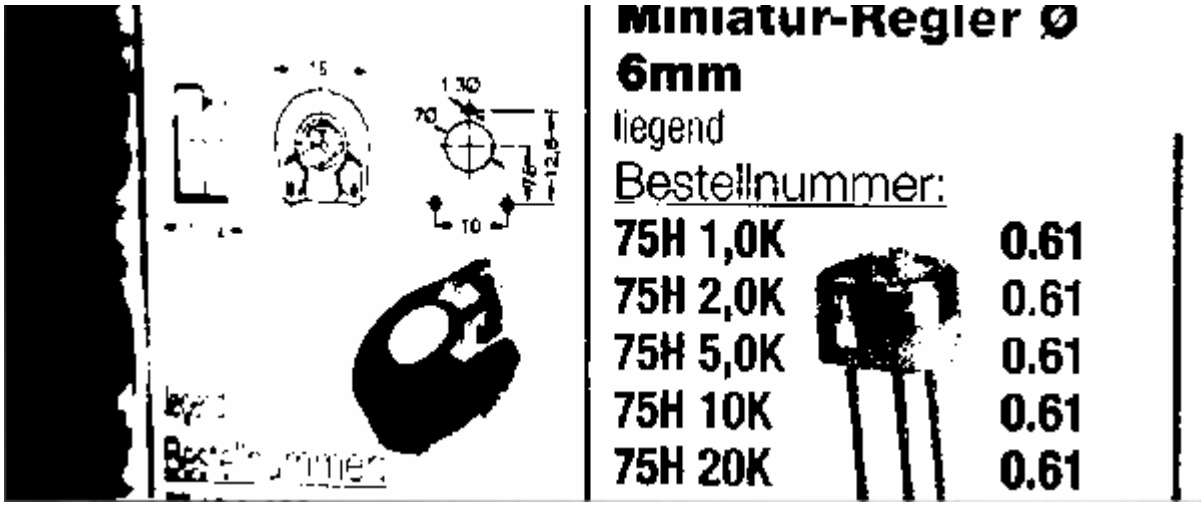
Hlavní rozdíl mezi funkcemi $f_S(n)$ a $g_S(n)$ pro výpočet průměrného prahu je ten, že funkce $g_S(n)$ přidává větší váhu na hodnoty bodů blíže prahovanému bodu, díky čemuž se funkce lépe přizpůsobuje změnám osvětlení.

Dále je potřeba pro prvních pár bodů předem určit počáteční práh a to nejlépe tak, aby negativně neovlivnil výsledek. Vhodné by bylo použít hodnotu sp_0 , nebo-li hodnotu prvního bodu násobenou počtem okolí *s*. Ovšem často jsou v rozích obrázku různé stíny nebo okraje, proto je lepší předem nastavit hodnotu 127s, jenž je odvozená jako poloviční hodnota z maximálního rozsahu 255 osmibitového čísla násobená počtem průměrovaných bodů *s*. Každopádně při zvolení jakékoliv

hodnoty, bude mít její velikost efekt pouze na několik následujících bodů. Zde je vzorec pro výpočet vlivu hodnoty n-tého pixelu na aktuální práh.

$$\frac{(1-\frac{1}{s})^n}{\sum_{i=0}^{n}(1-\frac{1}{s})^n} \tag{2.14}$$

Pro srovnání s předchozím výpočtem prahu funkcí $f_s(n)$, je zde uveden obrázek 9, který vznikl použitím Wall's algoritmu upraveného v předchozích krocích. Výsledek prahovaný v opačném směru je obdobný, ovšem stále s menšími odlišnostmi.



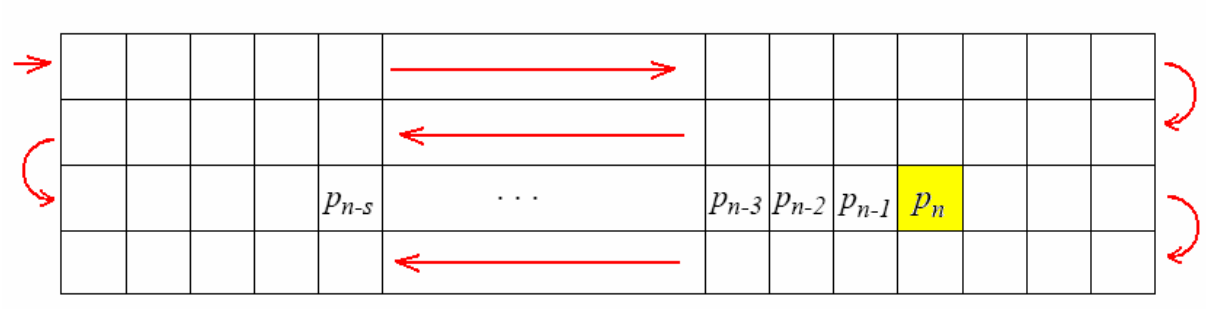
[obr.9] prahovaný obrázek směrem zleva doprava s = 100, t = 10%, použitím funkce $g_s(n)$

Je třeba odstranit závislost výsledných obrázků na směru prahování, abychom dosáhli jednoho algoritmu nezávislého na směru osvětlení. Jedním z možných řešení tohoto problému je považovat každý bod při průchodu jako střed průměrovaného okolí a tím zajistit, že průměr nebude závislý na směru, kterým jsme k bodu přišli. Vzorec pro takovýto přístup je následující

$$h_S(n) = \sum_{i=0}^{s-1} p_{n+\frac{s}{2}-i} \tag{2.15}$$

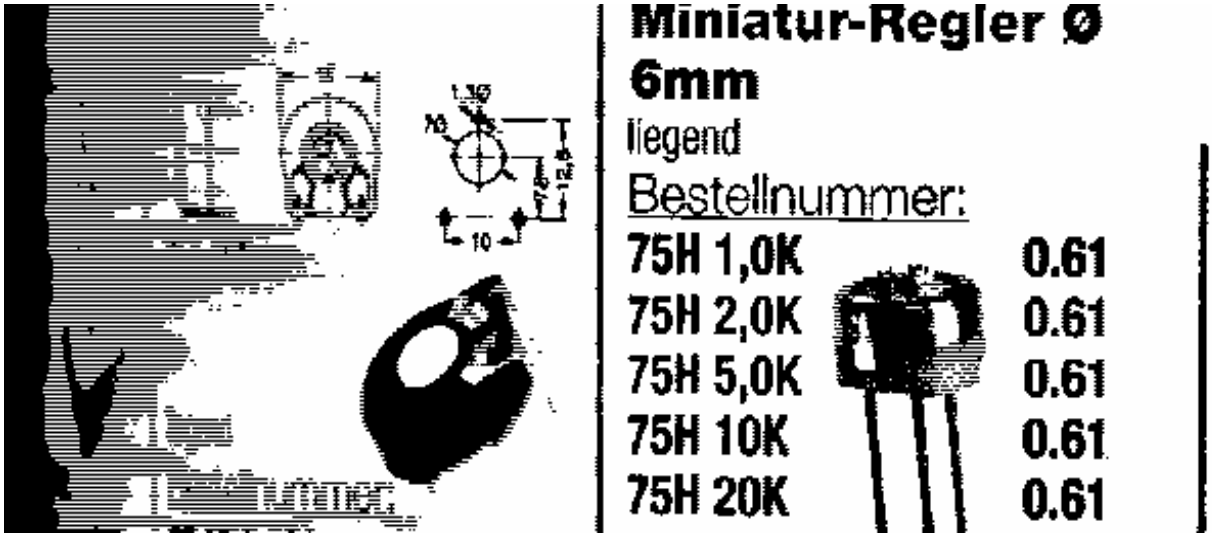
Kde $h_s(n)$ znamená funkci pro výpočet průměru jasových hodnot bodů, které jsou centrované okolo bodu p_n. Tímto vylepšením sice zlepšíme algoritmus tak, že již nebude závislý na směru průchodu obrázku, to znamená, že se již neliší výsledky průchodů zleva doprava nebo naopak. Pro takovýto výpočet musíme ovšem pro každý pixel projít pomocné pole okolních jasových hodnot a vypočítat jejich průměrnou hodnotu, čímž se několikrát prodlouží celkový převod.

Dalším možným vylepšením převodu je procházet obrázek střídavě oběma směry, jak je zobrazeno na obr. 10.



[obr.10] Ukázka průchodu obrázku oběma směry

Tímto průchodem by měl být částečně vyřešen také problém přechodu mezi řádky, kdy jedna strana je světlejší a druhá tmavší, který se projevil na obr. 7. Výsledný obrázek převedený střídáním směru průchodu vypadá následovně



[obr.11] prahovaný obrázek střídavě zleva a zprava, s = 100, t = 10%, použitím funkce g(n)

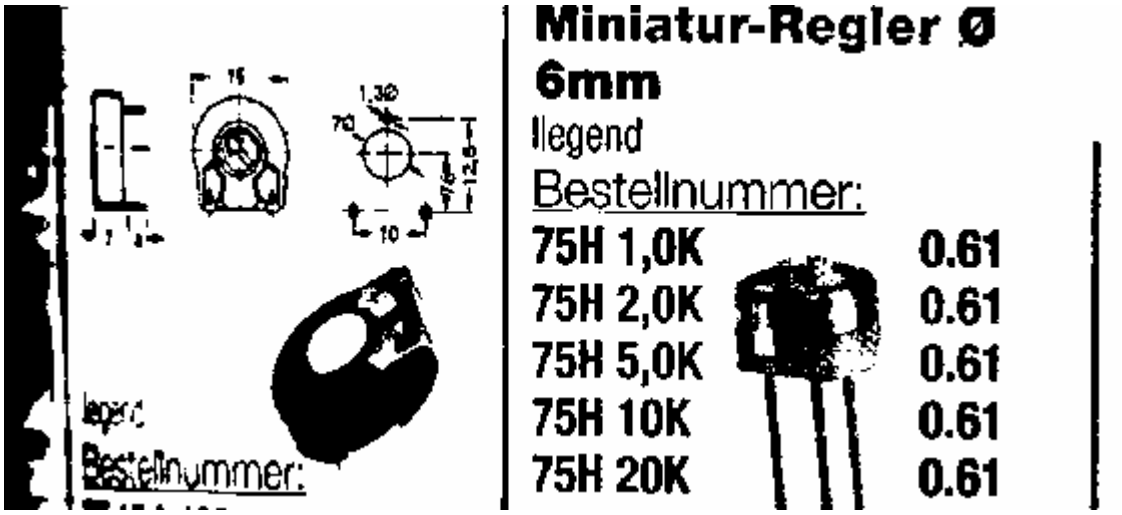
Výsledek je podobný jak pro prahování začínající zleva, tak zprava, a dosáhli jsme tím nezávislosti výsledku na zvoleném směru průchodu obrázku. Je zde také vidět většina objektů z obrázku. Ovšem oproti předchozím obrázkům si nelze nevšimnou rozdílu, kterým je nechtěné šrafování některých částí. Tento jev si lze vysvětlit tím, že při střídavém průchodu obrázku dochází ke složení obrázku ze dvou obrázků, kde liché řádky jsou brány z obrázku procházeného zleva doprava a sudé z obrázku procházeného opačným směrem. Toto šrafování se vyskytuje v šedých nerovnoměrně osvětlených oblastech obrázku, kde průchod jedním směrem vrací opačné hodnoty než průchod druhým směrem.

Tento jev bohužel výsledek převodu kazí a lze se jej zbavit tím, že do výpočtu prahu zahrneme i práh, který byl vypočítán ve stejném sloupci o řádek výše. Tím docílíme toho, že použitý práh bude vypočítán z výsledku prahů v obou směrech, i když na jiných řádcích. Vzorec pro výpočet prahu *i(n)*, který je průměrem vypočtených prahů je následující

$$i(n) = \frac{g(n) + g(n - width)}{2} \tag{2.16}$$

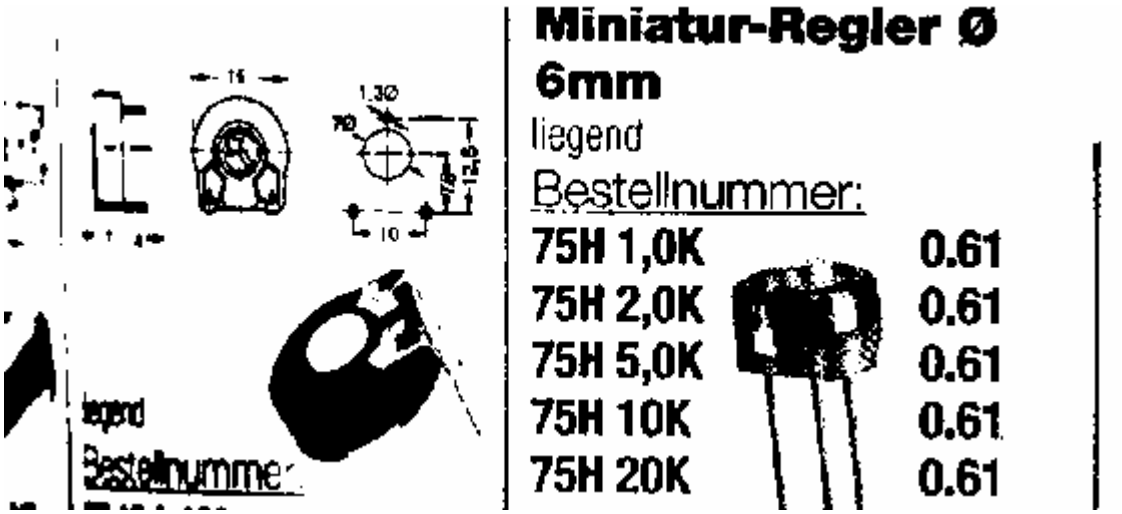
kde *width* je šířka řádku a *g(n - width)* je práh vypočtený ve stejném sloupci o řádek výše než je aktuální.

Konečný algoritmus pro převod se všemi doposud uvedenými vylepšeními použitý na obr.6 dosáhl výsledku zobrazeném na obr.12.



[obr.12] prahovaný obrázek střídavě zleva a zprava se započítáním prahu předchozího řádku, s = 100, t = 10%, použitím funkce g(n)

Výsledek na předchozím obrázku je ze všech uvedených nejpřesnější a není závislý na směru osvětlení ve vertikálním směru. Pokud je obrázek otočen o 90° a prahován se svislou změnou osvětlení, je dosaženo ještě lepšího výsledku (obr. 13) a to v levé části, kde vždy vznikal černý okraj.



[obr.12] prahovaný stejným způsobem jako na obr.11 ovšem otočený o 90°

O poslední verzi tohoto algoritmu se všemi úpravami lze tedy říci, že je dostatečně kvalitní, je také jednodušší a méně časově náročná než předchozí metody. Může být také implementována do hardwaru scanneru, nebo jiného snímacího zařízení.

3 Implementace programu

Tato kapitola popisuje použité vývojové nástroje a postup při tvorbě programu. V dalších částech jsou také podrobněji popsány použité algoritmy, konstrukce, funkce a modulové rozdělení programu. Budou zde také ukázáno uživatelské prostředí programu a popsána základní funkcionalita.

3.1 Použité nástroje

Program byl napsán v jazyce C++ s použitím standardní knihovny STL. Uživatelské rozhraní programu, nebo-li GUI, bylo vytvořeno ve vývojovém prostředí C++ Builder Enterprise suite verze 6, který byl vyvinut společností Borland Software Corporation. Toto prostředí umožňuje tvorbu jak konzolových, tak formulářových aplikací a je určeno systém pro MS Windows a je v něm zahrnuto mnoho hotových komponent pro tvorbu GUI, databázových, síťových nebo multimediálních aplikací. V prostředí je také kvalitně propracovaný systém nápovědy.

Přeložení programu do přenositelné verze požaduje určitá nastavení možností překladu, tyto možnosti nalezneme v nastavení vlastností projektu. Pro bezproblémové použití na počítačích bez C++ Builderu je třeba na kartě “Compiler“ zvolit “Release“ verzi kompilace, na kartě “Packages“ zrušit zatržení “Build with runtime packages“ a na kartě “Linker“ zrušit zatržení “Use dynamic RTL“.

3.2 Implementované algoritmy

Zadáním práce je vytvořit aplikaci, která upraví fotografovaný text do černobílé podoby, popřípadě využít OCR pro převod do textového formátu. Předem uvádím, že se mi sice podařilo implementovat převod do černobílé podoby použitím několika metod, ovšem převod do textového formátu jsem pouze nastínil tím, že jsem vytvořil funkci, která detekuje náklon řádků a otočí je do vodorovné polohy.

Od začátku návrhu programu jsem se snažil vytvořit program, který je schopný pracovat s více obrázky zaráz, jak jsem zvyklí z jiných programů na zpracování obrázků.

V programu jsem použil tyto algoritmy Lokální prahování s výpočtem prahu třemi způsoby (mean, průměr, medián), Wall’s algoritmus a globální prahování. Dále jsem implementoval funkce pro převod obrázku do šedi, otočení řádků, odstranění šumu, hromadný převod, zvětšování a zmenšování obrázku.

Jelikož tyto prahovací algoritmy minimálně jednou projdou obrázkem, může jejich výpočet trvat poměrně dlouhý čas, a proto by bylo dobré, kdyby uživatel nemusel čekat na dokončení převodu a mohl s programem dále pracovat. Toho jsem dosáhl použitím vláken, kdy pro každý výpočet se vytvoří jedno vlákno, které pracuje s obrázkem nezávisle na hlavním formuláři. Po ukončení převodu vlákno nahraje výsledek do cílového okna. Uživatelské rozhraní programu je díky tomu zcela použitelné i při převodu větších obrázků.

V dalších kapitolách jsou uvedeny zjednodušené algoritmy pro adaptivní prahování, které jsou v programu implementovány. Algoritmy jsou popsány v pseudokódu a není do nich zahrnuto zdaleka vše jako v programových funkcích.

3.2.1 Lokální prahování

Ve výsledném programu se můžeme lokálnímu prahování nastavit 3 parametry *velikost okolí*, udává velikost čtvercové oblasti, na které se obrázek rozdělí, *offset* je číslo, které se přičte k vypočtenému prahu, můžeme jím regulovat světlost obrázku a *min rozdíl*, který udává jak velký rozdíl musí být mezi pixely v okolí, aby nebylo okolí bráno jako pozadí. Algoritmus je popsán pouze pro průměr, další dva způsoby výpočtu prahu se liší jen nepatrně pozměněním způsobu výpočtu.

```
pro všechny oblasti obrázku
Begin
  pro každý pixel p[x][y] v oblasti
  Begin
    p[x][y] = šedá (p[x][y])
    suma = suma + p[x][y]
  End
  T = suma/počet_pixelů_v_oblasti
  pro každý pixel p[x][y] v oblasti
  Begin
    if p[x][y] <= T
       p[x][y] = černá
    else
       p[x][y] = bílá
  End
End
```

3.2.2 Wall's algoritmus

V programu lze tomuto algoritmu nastavit dva parametry okolí, které udává počet pixelů zahrnutých do výpočtu prahu, a procento udávající o kolik procent se prahovaný pixel od prahu musí odlišovat, aby mu byla přiřazena černá barva.

Následuje zjednodušený pseudokód Wall's algoritmu.

```
procházej obrázek střídavě po řádcích pro každý pixel p[x][y]
Begin
   p[x][y] = šedá(p[x][y])
   suma = suma - suma/velikost_okolí
   suma = suma + p[x][y]
   průměr = suma/ velikost_okolí
   T = (průměr + práh_o_řádek_výše[x])/2
   if p[x][y] <= T
       p[x][y] = černá
   else
       p[x][y] = bílá
   práh_o_řádek_výše[x] = T
End
```

3.3 Použité třídy a komponenty

Program se celkově skládá ze 4 modulů, pokud opomeneme hlavní modul vytvořený vývojovým prostředím. Z těchto modulů jsou 3 formulářové a jeden modul s funkcemi pro práci s obrázky. Stručně popíši všechny moduly.

MainForm.cpp: tento modul obsahuje definici třídy TFormMain hlavního formuláře

ChildForm.cpp: obsahuje definici třídy TFormChild formuláře typu MDIChild, který je zobrazen ve vymezené oblasti hlavního okna formuláře.

ThreadObrFce.cpp: v tomto modulu jsou třídy pro zpracování obrázků.

About.cpp: modul obsahuje deklaraci třídy pro formulář, který se zobrazí jako nápověda „O programu"

V dalších kapitolách popíši některé konstrukce v hlavních modulech programu.

3.3.1 Modul MainForm.cpp

V tomto modulu je již zmíněná třída pro hlavní formulář TFormMain, který se vytváří automaticky při startu programu. Tato třída obsahuje, kromě deklarace komponent umístěných na formuláři a

definice funkcí reagujících na událost komponent, také několik pomocných funkcí pro práci s hlavním, nebo child oknem. Jsou tu dále dvě proměnné typu TList, což je seznam implementovaný firmou Borland. Těchto seznamů je využito k uložení ukazatelů na child okna a jsou použity k realizaci výběru okna pomocí komponenty ListBox. Dále třída zahrnuje jednu proměnnou typu TPrevod, která slouží k uložení informací o hromadném převodu.

3.3.2 Modul ChildForm.cpp

Soubor ChildForm.cpp obsahuje kód třídy pro child okna TFormChild, tyto okna se nevytvářejí automaticky při spuštění, ale vždy při otevření nového souboru, nebo při použití některé z převodových metod.

Třída obsahuje několik metod pro práci s obrázkem, nebo formulářem, dále také obsahuje jednu proměnnou typu TInfoObr, kde jsou uloženy všechny potřebné informace o načteném obrázku.

3.3.3 Modul ThreadObrFce.cpp

V tomto modulu je deklarována hlavní třída TObrFceThread, která je odvozena z třídy TThread. Třída TThread je v použitém prostředí základní třídou pro tvorbu vícevláknových aplikací.

Princip implementace vláken v tomto programu je založen na třídě TObrFceThread, z této třídy dědí ostatní funkce a je v ní metoda Execute, která se vykoná po spuštění vlákna. Aby třídy, které jsou odvozené z TObrFceThread, mohli vykonávat svůj vlastní kód, je v hlavní třídě také virtuální funkce Akce, kterou lze v třídách odvozených předefinovat. Ve výsledku je celý postup takový, že po spuštění vlákna odvozené třídy se provede metoda Execute hlavní třídy, v ní se volá funkce Akce, která je v odvozené třídě předefinována.

Kromě hlavní třídy TObrFceThread, jsou v modulu deklarovány také třídy TGrayscale, TAdaptPrahovani, TWallPrahovani, TGlobPrahovani, TOtoceniObr, TZjistiOtoceni, TOdstranSum. Význam jednotlivých tříd odpovídá jejich názvům.

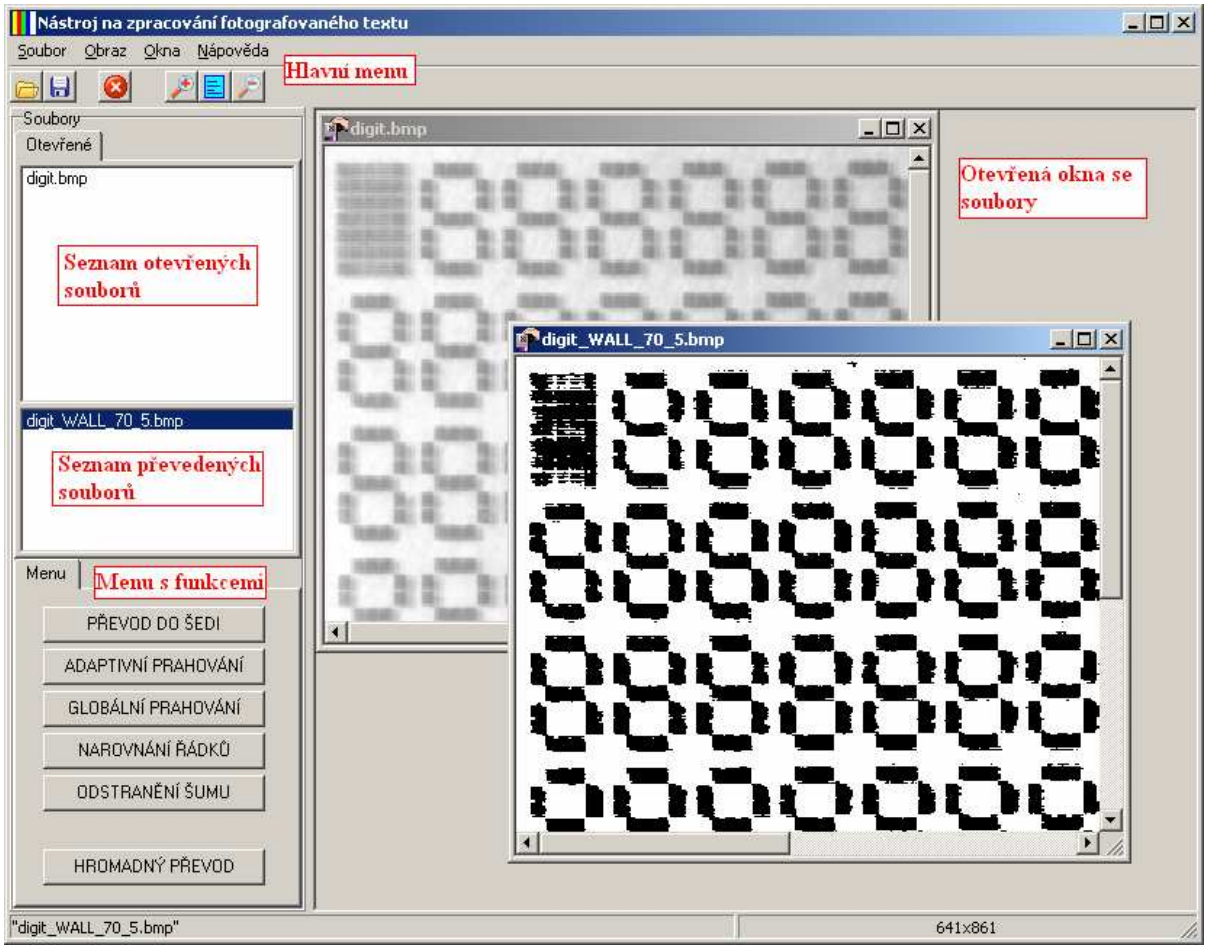
V modulu jsou také pomocné funkce pro otevření, uložení a přejmenování souboru, které jsou použity k práci se soubory v hlavním okně aplikace.

4 Prostředí programu

V této kapitole bude popsán výsledný program, který se mi podařilo vytvořit.

4.1 Ukázka hlavního okna programu

Na obr.13 je zobrazeno hlavní okno programu s popiskami nejdůležitějších částí.



[obr. 13] Hlavní okno programu

Hlavní menu je složeno z nabídky **Soubor,** kde jsou základní funkce jako uložení, otevření, zavření souboru a spuštění hromadného převodu obrázků. Nabídka **Obrázek** zahrnuje funkce pro zoomování, převod, rotaci a narovnání řádků. Nabídka **Okna** lze použít k uspořádání oken, nebo přepínání se mezi okny. Poslední nabídka **Nápověda** obsahuje základní uživatelskou nápovědu k programu.

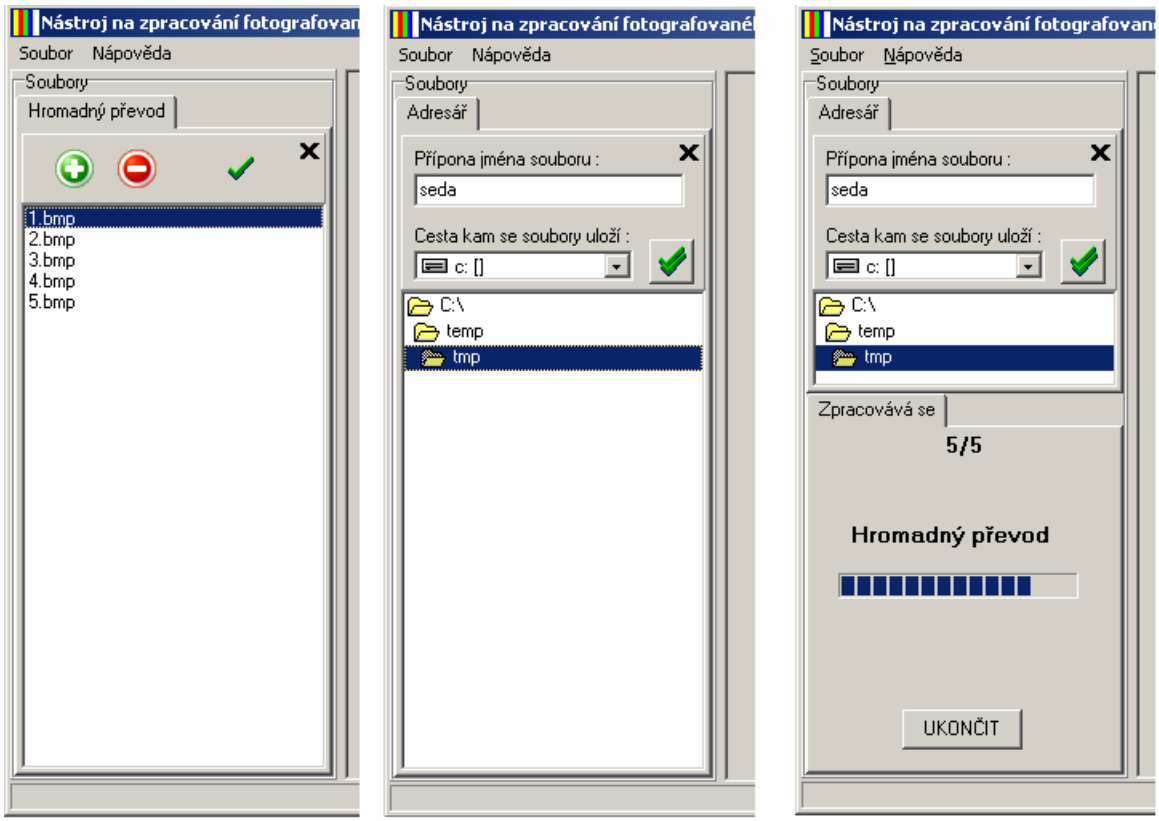
Otevřená okna souborů v této oblasti jsou zobrazena všechna otevřená okna s obrázky.

Seznam otevřených souborů je komponenta listbox, kde si lze vybrat jedno okno, které chce uživatel zobrazit jako aktivní. Jsou zde okna, s originálními otevřenými soubory.

Seznam převedených souborů je seznam oken, kde jsou zobrazeny obrázky, které byly získány z originálních otevřených souborů některou z převodových metod. Za název originálního souboru je přidána přípona podle použité metody.

Menu s funkcemi je skupina tlačítek pro převod obrázků, na tomto místě se zobrazí také volby pro nastavení parametrů převodu po vybrání konkrétní metody.

Program také umožňuje převést více obrázků najednou, aniž bychom museli všechny otevírat a převádět postupně. Pro tuto možnost je zde volba Hromadný převod, pro kterou se nám zobrazí tato nabídka chronologicky seřazená podle postupu převodu.



[obr. 14] Hromadný převod

Na prvním obrázku je dialog pro přidávání souborů, druhý obrázek zobrazuje okno pro výběr cesty, kam se mají převedené soubory ukládat a přípony, která se přidává za názvy souborů, na posledním obrázku se soubory již převádějí.

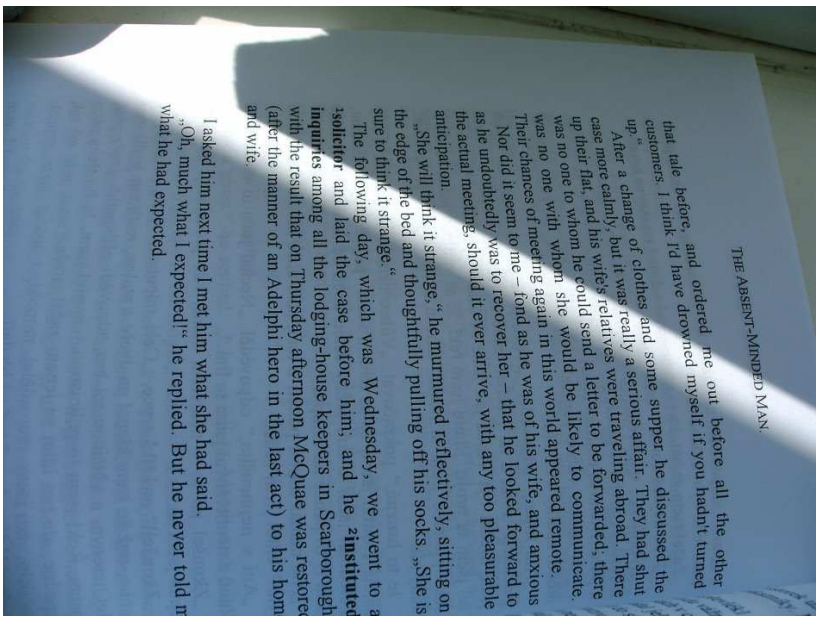
5 Testování programu

V této kapitole jsou popsány zkušební testy, byla vybrána taková vzorová data, aby se na nich daly demonstrovat nedostatky vzniklé při fotografování textu. Pro porovnání budou uváděny i časy

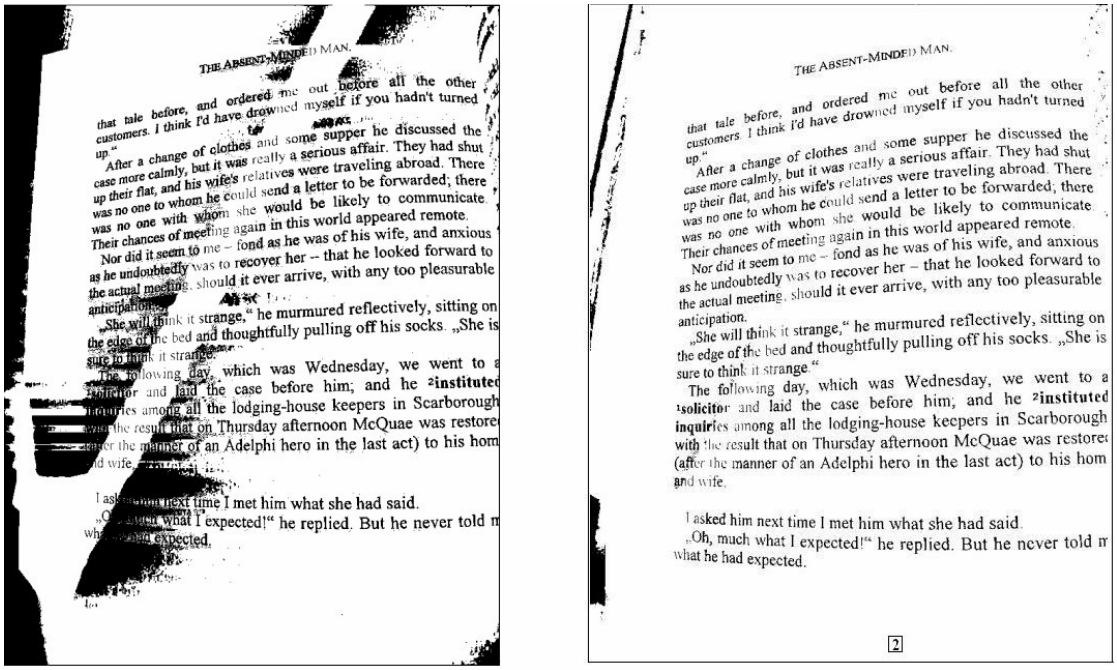
převodu, tyto časy jsou čistě informativní a jsou závislé na hardwarovém vybavení. Testovací obrázky byly převáděny na počítači s procesorem Intel Core 2 Duo o taktu 1,5 GHz a operační pamětí velikosti 1024 MB.

5.1 Test 1

Pro první test byl vybrán obrázek s vysokým kontrastem osvětlení, které může být způsobeno ostrým přechodem světla a stínu.



[obr. 15] Testovací obrázek 1 (rozlišení 994x708 px)

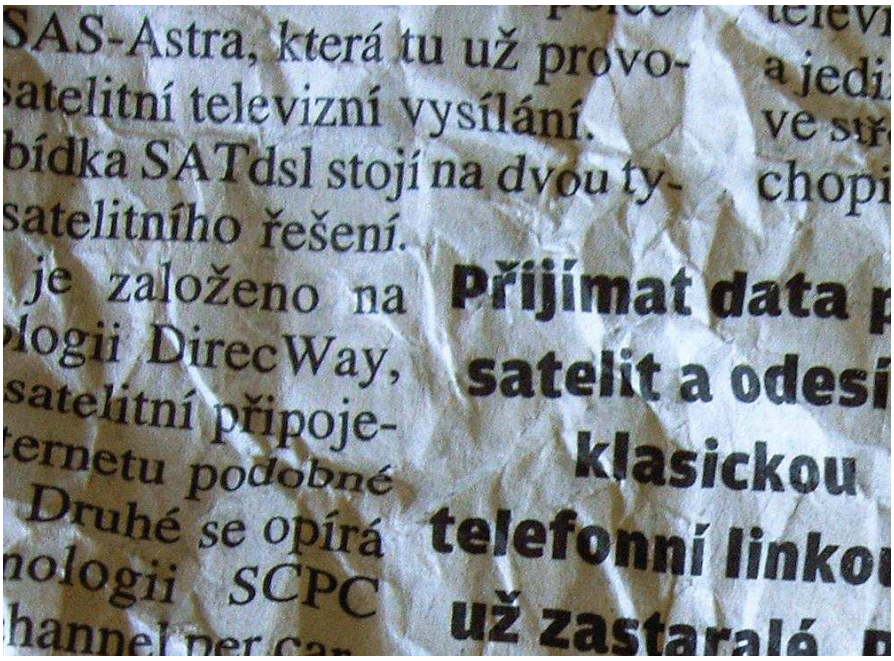


[obr. 16] Výsledky převodu obr.15 Wall's algoritmem: **1** okolí = 100, procento = 10, **2** okolí = 25, procento = 30

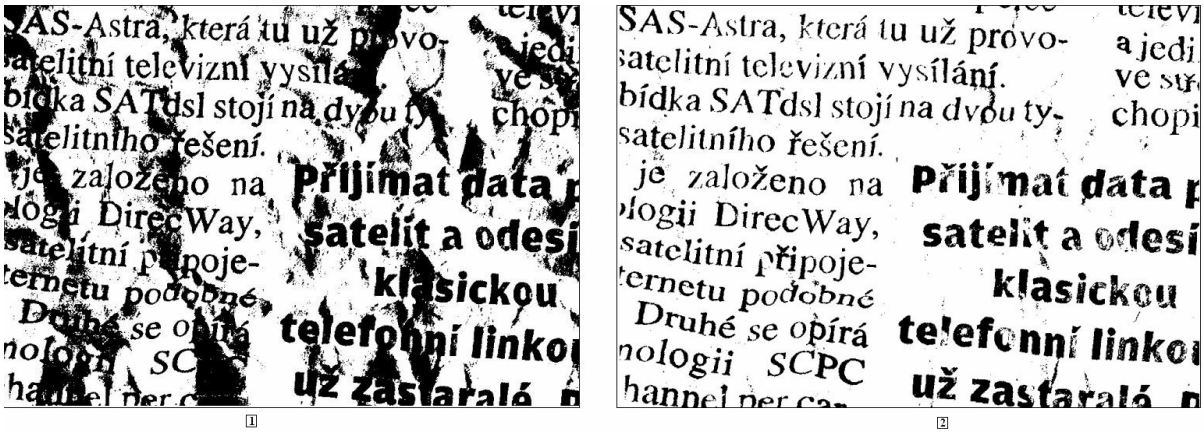
Na obrázku 16 jsou dosažené výsledky převodu Wall's algoritmem, celý převod trval 3,2s. Levý obrázek (1) byl prahován s použitím příliš velkého okolí, které se nestačilo přizpůsobit rychlé změně osvětlení v obrázku, a proto je zde přechod světlo-stín jasně ohraničen. V pravém obrázku je již nastavena vhodná velikost okolí a výsledek je o poznání lepší.

5.2 Test 2

Druhý test v pořadí je proveden na zmačkaném novinovém papíře, který byl vyfotografován (obr. 17). V tomto případě je důležité vybrat správné parametry pro převod, protože se v obrázku vyskytuje spoustu různých odlesků a stínů.



[obr. 17] testovací obrázek 2 (rozlišení 1000x733 px)

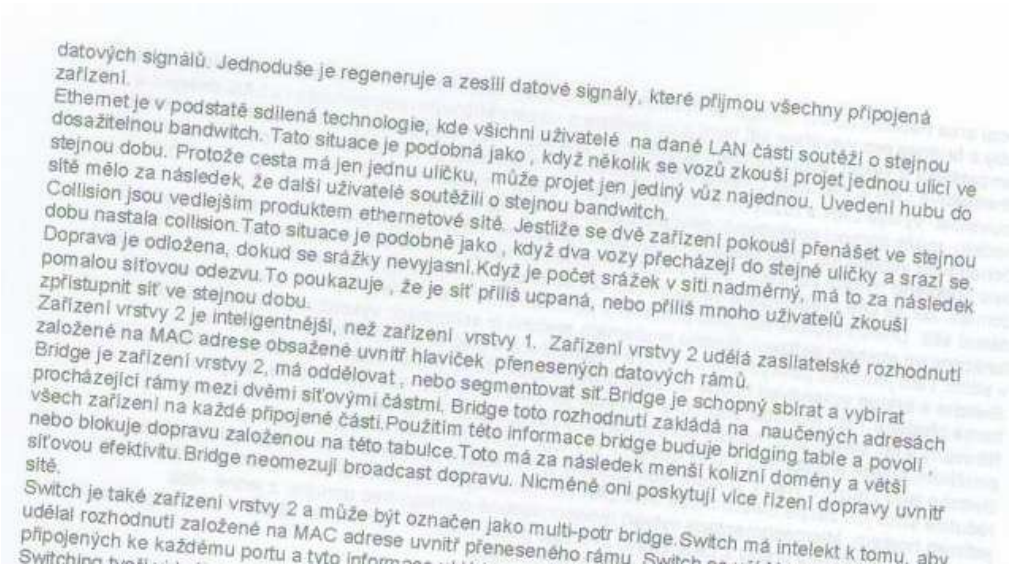


[obr. 18] převedený obrázek 17 Wall's algoritmem: **1** okolí 100, procento 10, **2** okolí 30, procento 35

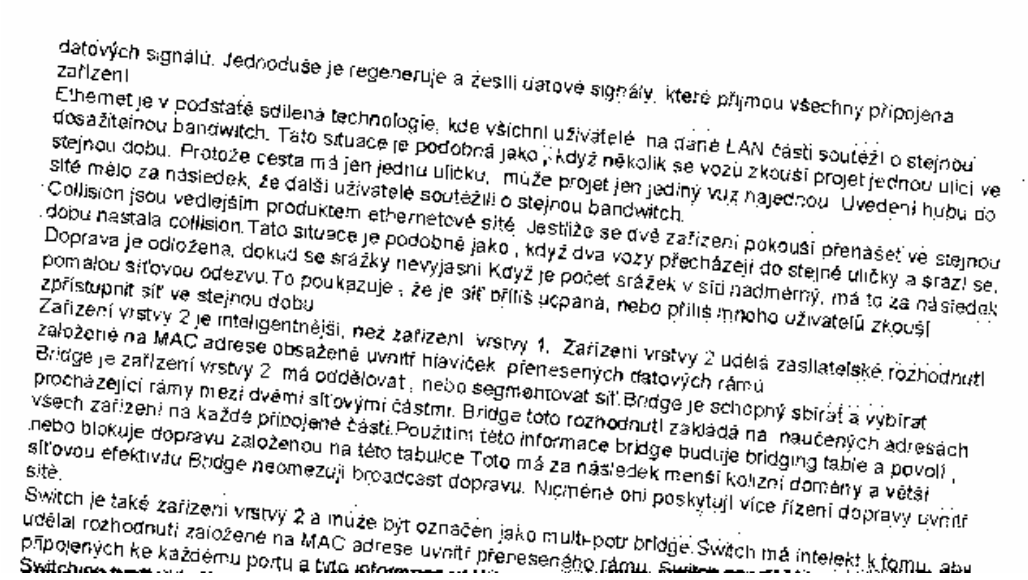
Převod obrázku 17 trval 3,5s a byl převeden opět dvěma způsoby. V levém z převedených obrázků (obr. 18) je nastavena opět špatná velikost okolí, která způsobila převedení stínů do černé barvy. Ovšem ani pravý obrázek není bez chyb, protože zde dochází k vynechání částí písmen, která jsou na rozhraní světla stínu.

5.3 Test 3

Tento test se zaměří na převod textu, který obsahuje nechtěné zabarvení do modra, odstín byl způsoben špatnou kvalitou skeneru. Je zde také zachycený stín textu na druhé stránce, který ruší text v popředí. Obrázek je schválně naskenován pootočený, aby jsme na něm mohli v dalším testu vyzkoušel algoritmus natočení řádků.


datových signálů. Jednoduše je regeneruje a zesílí datové signály, které přijmou všechny připojená zařízení.
Ethernet je v podstatě sdílená technologie, kde všichni uživatelé na dané LAN části soutěží o stejnou dosažitelnou bandwitch. Tato situace je podobná jako , když několik se vozů zkouší projet jednou ulicí ve stejnou dobu. Protože cesta má jen jednu uličku, může projet jen jediný vůz najednou. Uvedení hubu do sítě mělo za následek, že další uživatelé soutěžili o stejnou bandwitch.
Collision jsou vedlejším produktem ethernetové sítě. Jestliže se dvě zařízení pokouší přenášet ve stejnou dobu nastala collision.Tato situace je podobné jako , když dva vozy přecházejí do stejné uličky a srazí se. Doprava je odložena, dokud se srážky nevyjasní.Když je počet srážek v síti nadměrný, má to za následek pomalou síťovou odezvu.To poukazuje , že je síť příliš ucpaná, nebo příliš mnoho uživatelů zkouší zpřístupnit síť ve stejnou dobu.
Zařízení vrstvy 2 je inteligentnější, než zařízení vrstvy 1. Zařízení vrstvy 2 udělá zasílatelské rozhodnutí založené na MAC adrese obsažené uvnitř hlaviček přenesených datových rámů.
Bridge je zařízení vrstvy 2, má oddělovat , nebo segmentovat síť.Bridge je schopný sbírat a vybírat procházející rámy mezi dvěmi síťovými částmi. Bridge toto rozhodnutí zakládá na naučených adresách všech zařízení na každé připojené části.Použitím této informace bridge buduje bridging table a povolí , nebo blokuje dopravu založenou na této tabulce.Toto má za následek menší kolizní domény a větší síťovou efektivitu.Bridge neomezují broadcast dopravu. Nicméně oni poskytují více řízení dopravy uvnitř sítě.
Switch je také zařízení vrstvy 2 a může být označen jako multi-potr bridge.Switch má intelekt k tomu, aby udělal rozhodnutí založené na MAC adrese uvnitř přeneseného rámu. Switch má intelekt k tomu, aby připojených ke každému portu a tyto informace

[obr. 19] testovací obrázek 3 (rozlišení 636x353 px)

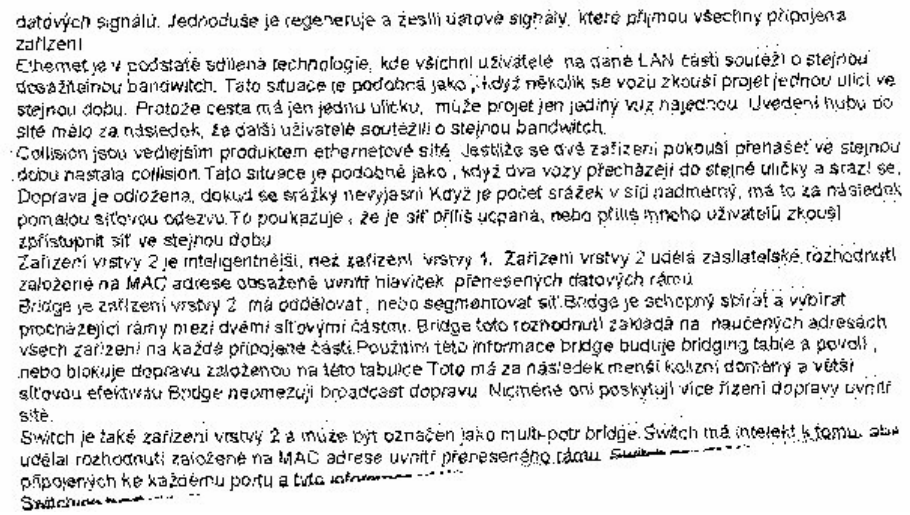

datových signálů. Jednoduše je regeneruje a zesílí datové signály, které přijmou všechny připojená zařízení.
Ethernet je v podstatě sdílená technologie, kde všichni uživatelé na dané LAN části soutěží o stejnou dosažitelnou bandwitch. Tato situace je podobná jako , když několik se vozů zkouší projet jednou ulicí ve stejnou dobu. Protože cesta má jen jednu uličku, může projet jen jediný vůz najednou. Uvedení hubu do sítě mělo za následek, že další uživatelé soutěžili o stejnou bandwitch.
Collision jsou vedlejším produktem ethernetové sítě. Jestliže se dvě zařízení pokouší přenášet ve stejnou dobu nastala collision.Tato situace je podobné jako , když dva vozy přecházejí do stejné uličky a srazí se. Doprava je odložena, dokud se srážky nevyjasní.Když je počet srážek v síti nadměrný, má to za následek pomalou síťovou odezvu.To poukazuje , že je síť příliš ucpaná, nebo příliš mnoho uživatelů zkouší zpřístupnit síť ve stejnou dobu.
Zařízení vrstvy 2 je inteligentnější, než zařízení vrstvy 1. Zařízení vrstvy 2 udělá zasílatelské rozhodnutí založené na MAC adrese obsažené uvnitř hlaviček přenesených datových rámů.
Bridge je zařízení vrstvy 2, má oddělovat , nebo segmentovat síť.Bridge je schopný sbírat a vybírat procházející rámy mezi dvěmi síťovými částmi. Bridge toto rozhodnutí zakládá na naučených adresách všech zařízení na každé připojené části.Použitím této informace bridge buduje bridging table a povolí , nebo blokuje dopravu založenou na této tabulce.Toto má za následek menší kolizní domény a větší síťovou efektivitu.Bridge neomezují broadcast dopravu. Nicméně oni poskytují více řízení dopravy uvnitř sítě.
Switch je také zařízení vrstvy 2 a může být označen jako multi-potr bridge.Switch má intelekt k tomu, aby udělal rozhodnutí založené na MAC adrese uvnitř přeneseného rámu. Switch má intelekt k tomu, aby připojených ke každému portu a tyto informace

[obr. 20] výsledek převodu adaptivním prahováním s výpočtem prahu funkcí mean, okolí = 9, offset = -10, min rozdíl = 10

Převod obrázku 19 není až tak náročný na experimentování s parametry jako předchozí, při špatném nastavení se ve výsledku mění světlost a četnost výskytu šumu v obrázku. Testovaný obrázek byl převeden pro změnu lokálním prahováním, převod trval 1,4s.

5.4 Test 4

V tomto testu byl zpracován výsledný obrázek z předchozího textu (obr. 20). Na obrázku byl použit algoritmus natočení řádků do vodorovné polohy.


datových signálů. Jednoduše je regeneruje a zesílí datové signály, které přijmou všechny připojená zařízení
Ethernet je v podstatě sdílená technologie, kde všichni uživatelé na dané LAN části soutěží o stejnou dosažitelnou bandwitch. Tato situace je podobná jako, když několik se vozu zkouší projet jednou ulicí ve stejnou dobu. Protože cesta má jen jednu uličku, může projet jen jediný vůz najednou. Uvedení hubu do sítě mělo za následek, že další uživatelé soutěžili o stejnou bandwitch.
Collision jsou vedlejším produktem ethernetové sítě. Jestliže se dvě zařízení pokouší přenášet ve stejnou dobu nastala collision. Tato situace je podobné jako , když dva vozy přecházejí do stejné uličky a srazí se. Doprava je odložena, dokud se srážky nevyjasní. Když je počet srážek v síti nadměrný, má to za následek pomalou síťovou odezvu. To poukazuje , že je síť příliš ucpaná, nebo příliš mnoho uživatelů zkouší zpřístupnit síť ve stejnou dobu
Zařízení vrstvy 2 je inteligentnější, než zařízení vrstvy 1. Zařízení vrstvy 2 udělá zasílatelské rozhodnutí založené na MAC adrese obsažené uvnitř hlaviček přenesených datových rámců.
Bridge je zařízení vrstvy 2 má oddělovat, nebo segmentovat síť. Bridge je schopný sbírat a vybírat procházející rámy mezi dvěmi síťovými částmi. Bridge toto rozhodnutí zakládá na naučených adresách všech zařízení na každé připojené části. Použitím této informace bridge buduje bridging table a povolí , nebo blokuje dopravu založenou na této tabulce. Toto má za následek menší kolizní domény a větší síťovou efektivitu. Bridge neomezují broadcast dopravu. Nicméně oni poskytují více řízení dopravy uvnitř sítě.
Switch je také zařízení vrstvy 2 a může být označen jako multi-port bridge. Switch má intelekt k tomu, aby udělal rozhodnutí založené na MAC adrese uvnitř přeneseného rámu.
připojených ke každému portu a tato

[obr. 21] výsledek otočení obrázku 20

Převod z obrázku 20 do výsledného obrázku 21 trval 4,4s a při zvětšení je na výsledku poznat, že některá písmena mají oproti originálu horší kvalitu, která se projevuje posunutím čar písmene. Takovéto zhoršení je způsobeno tím, že při otáčení je počítána pozice každého pixelu v reálných číslech a pro výsledné zobrazení se reálná hodnota musí převést na celočíselnou, kde vznikají chyby.

6 Závěr

Výsledkem mé bakalářské práce je program, který je použitelný pro převod fotografovaného textu obsahujícího různé stíny a barevné přechody do černobílé podoby. Je možné s ním provádět i operace jako natočení řádků, nebo hromadný převod více souborů.

Aplikace je vícevláknová, takže nedochází k zamrznutí při náročných výpočtech. To, že jsem od začátku vytvářel tuto aplikaci s možností práce s více soubory současně, do určité míry zpomalilo postup při implementaci, zvláště posledních dvou algoritmů, které díky špatnému přístupu vlákna k formuláři způsobovaly zamrznutí aplikace. Kdybych nepoužil vláken, určitě by mi zbylo více času na přidání dalších rozšiřujících funkcí.

Práce by se mohla dále rozvíjet tak, že by se doplnily funkce pro detekci řádků a jednotlivých písmen. Oddělená písmena bych porovnával se vzory a převáděl do textové verze.

K dokončení práce jsem prostudoval mnoho technických textů o adaptivním prahování, algoritmy, které jsem využil v programu jsou popsány v kapitolách 2.2.2 a 2.2.3. Seznámil jsem se také s principem funkce OCR systémů a s problémy, které musí řešit, aby dosáhli co nejvyšší úspěšnosti. V neposlední řadě jsem se zjistil výhody, ale i nevýhody vytváření vícevláknových aplikací v prostředí C++ Bulderu. Potvrdil jsem si také fakt, že velmi podstatnou roli v této aplikaci i ve všech rozsáhlejších aplikacích plní správné rozvržení programu do částí, díky čemuž si tvůrci aplikace ušetří velké množství času pozdějších úpravách programu.

Literatura

[1] Pierre D. Wellner, Adaptive Thresholding for the DigitalDesk. EuroPARC Technical Report 1993. Dostupné na URL: http://citeseer.ist.psu.edu/wellner93adaptive.html (květen 2008)

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

[3] Bryan S. Morse, Lecture 4: Thresholding, Brigham Young University, 2000 Dostupné na URL: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf (květen 2008)

[4] Wikipedie: Otevřená encyklopedie: OCR [online]. Dostupné na URL: http://en.wikipedia.org/wiki/OCR

Seznam příloh

Příloha 1. Uživatelský manuál pro převod obrázku do černobílé podoby adaptivním prahováním

Příloha 2. CD se zdrojovými soubory projektu a přeloženým programem

Adresářová struktura CD :

1. zdrojové soubory\	obsahuje zdrojové soubory pro překlad projektu
2. přeložený program\	obsahuje výsledný přeložený program a nápovědu k programu
3. technická zpráva\	obsahuje technické text zprávy v elektronické podobě ve formátu PDF a DOC
4. testovací obrázky\	obsahuje testovací obrázky pro ověření funkčnosti převodu a také obrázky použité v testech

Příloha 1

Uživatelský manuál pro převod obrázku do černobílé podoby adaptivním prahováním

1. Pomocí menu *Soubor->Otevřít* otevřete soubor s obrázkem (můžete i více souborů), lze použít i klávesové zkratky CTRL+O, nebo stejného tlačítka v horním panelu. Vybrané soubory se přidají do levého horního seznamu.
2. Výběrem souboru v seznamu vlevo nahoře označte obrázek, který chcete převést a stiskněte tlačítko ADAPTIVNÍ PRAHOVÁNÍ v panelu menu.
3. V otevřené nabídce vyberte metodu prahování, zvolte vhodné parametry převodu a stiskněte tlačítko PRAHUJ.
4. Obrázek v aktivním okně se začne převádět, průběh převodu můžete vidět na místě, kde jse nastavovali parametry převodu převodu. Průběh lze zastavit tlačítkem UKONČIT.
5. Pokud převod proběhne v pořádku v levém dolním seznamu s převedenými obrázky se přidá okno, v kterém se zobrazí výsledný obrázek.
6. Chcete-li obrázek uložit vyberte jej buď v jednom ze seznamů, nebo označením okna a zvolte volbu *Soubor->Uložit obrázek* (klávesová zkratka CTRL+S). Vyberte adresář, jméno a typ pro uložený soubor.
7. Budete-li potřebovat nápovědu k dalším funkcím, naleznete ji v souboru s nápovědou (Menu *Nápověda->Obsah*).