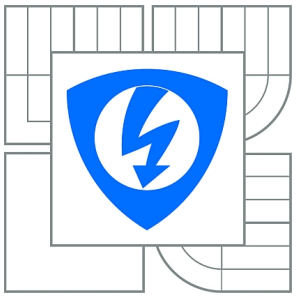
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ROZPOZNÁVÁNÍ TEXTU Z FOTOGRAFIÍ

TEXT RECOGNITION FROM IMAGES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

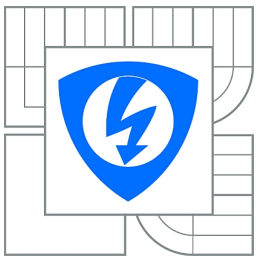
AUTOR PRÁCE — Bc. JAN MIXA

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN HASMANDA

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jan Mixa	***ID:***	119537
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Rozpoznávání textu z fotografií

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta v diplomové práci bude návrh aplikace, umožňující rozpoznávání textu z fotografie listu papíru s textem, položeném na vodorovné podložce. Studentovi je doporučeno používat pro předzpracování obrazu jasové a geometrické transformace. Dále by měl být podrobně popsán algoritmus využitý pro rozpoznávání textu v samotné práci. Výstupem studenta bude také zhodnocení kvality rozpoznání textu oproti původnímu dokumentu na několika pořízených obrazech a export rozpoznávaného textu do jazyka TeX pro LaTeX. Student může využívat volně dostupné knihovny pro rozpoznávání textu a knihovnu OpenCV.

DOPORUČENÁ LITERATURA:

[1] JAN, J. Medical Image Processing, Reconstruction and Restoration, 2006. 760s.

[2] Robert Laganiere. OpenCV 2 Computer Vision Application Programming Cookbook. Packt Publishing, May 2011.

Termín zadání:	11.2.2013	***Termín odevzdání:***	29.5.2013

Vedoucí práce: Ing. Martin Hasmanda
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Hlavním cílem této diplomové práce bylo vytvořit program pro rozeznání textu z fotografií. Dále pak podrobně se seznámit se základními technikami zpracování obrazu, jasovými a geometrickými transformacemi, hlavně pak s metodami, které jsou používané pro rozpoznávání textu. Na úvod byly popsány základní principy jasových a geometrických transformací. Poté byly představeny jednotlivé metody pro rozeznání textu. Více jsem se věnoval metodě Template Matching, která je použita v hlavním programu. V následujících kapitolách byly popsány otevřené knihovny pro práci s obrazem. V praktické části byly vytvořeny tři programy. První pro pouhé rozeznání řádků a znaků v programovacím jazyku C. Druhý, který byl vystavěn na základech prvního programu, již v jazyce C++. Tento program už byl o něco sofistikovanější. Mimo počítání řádků a znaků, je schopen korigovat natočení pořizovaného obrazu, najít a zobrazit kontury a konečně rozpoznat text a výstup uložit do souboru. Oba programy využívali knihovny OpenCV. Posledním programem byla jednoduchá aplikace, která představovala moderní knihovnu Tesseract.

KLÍČOVÁ SLOVA

Zpracování obrazu, jasové a geometrické transformace, Template Matching, OpenCV, Tesseract

ABSTRACT

The main objective if this diploma thesis was to create a program for recognizing text from pictures. Then, to become familiar with the basic techniques of image processing, luminance and geometric transformations, especially with methods that are used for text recognition. At the introduction were described basic principles of luminance and geometric transformations. Then were presented various methods for text recognition. I was more focused on Template Matching method, which is used in the program. The following chapters describe open libraries for image recognition. In the practical part were created three programs. First one, for mere recognition lines and characters in programming language C. The second, which was built on the foundations of first program, already in C++ language. This program has been a bit sophisticated. Except counting of lines and characters, is able to correct rotation of the acquired image, find and display contours and finally recognize text and this output save to file. Both programs used the OpenCV library. The last program was a simply application which represent a modern library Tesseract.

Keywords

Image processing, luminance and geometric transformations, Template Matching, OpenCV, Tesseract

Bibliografická citace mé práce:

MIXA, J. Rozpoznávání textu z fotografií. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 61 s. Vedoucí diplomové práce Ing. Martin Hasmanda.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma **Rozpoznávání textu z fotografií** jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4. Trestního zákoníku č. 40/2009 Sb.

V Brně dne …….. ……

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Martinu Hasmandovi, za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne ……..…….. ……

podpis autora

Seznam obrázků a tabulek

Seznam obrázků

Seznam Tabulek

Úvod

Svou diplomovou práci jsem zaměřil na rozpoznání textu z fotografií. Tato problematika spadá pod oblast počítačového vidění, které se zabývá zpracováváním informací ze zachyceného obrazu.

Počítačové vidění se dnes uplatňuje stále více ve všech oblastech praktického života. Ať už se jedná o rozpoznávání obličejů nebo registračních značek z průmyslových kamer, řízení autonomních systémů či při interakci počítače s člověkem.

Oblast počítačového vidění je na vzestupu také díky snadno přístupným kamerovým systémům a programovacích pomůcek. Mezi tyto pomůcky patří např. knihovna OpenCV, knihovna, která se svými funkcemi zaměřuje na real-time zpracování obrazu.

Tato diplomová práce má za úkol seznámit čtenáře s problematikou rozpoznávání a identifikací textu nasnímaného fotoaparátem, jednoduššími i pokročilými technikami detekce jednotlivých znaků a jejich rozpoznání.

Hlavním cílem této práce je vytvořit program, který rozezná text z nasnímaného obrazu a vyexportuje ho do souboru textového editoru Latex. Poté program použít na vybrané obrazy a zhodnotit jeho účinnost vzhledem k vlastnostem jednotlivých obrazů.

1. Cíl práce

V teoretické části se zabývám základními pojmy v oblasti rozpoznávání textu, počítačového vidění a předzpracování obrazu. Pozornost jsem věnoval metodám pro jasovou a geometrickou transformaci. Hlavním bodem teoretické části však bylo vysvětlení algoritmů pro detekci a rozpoznání textu a algoritmů použitých při vytváření výstupního programu. V poslední části teoretického úseku se má práce zaměřuje na oblast optického rozpoznávání znaků a vysvětluje základní vlastnosti knihovny OpenCV a Tesseract.

Praktickou část tvoří vytvoření prvního programu pro načítání obrázku a výpis počtu řádků a písmen. Dalším programem je již plnohodnotný program, který je schopen načítaný obraz upravit a rozeznat text. Také jsou v této části vysvětleny použité postupy a metody jednotlivých programů. Tyto programy jsou tvořeny pomocí knihoven OpenCV. Posledním programem je jednoduchá aplikace, která představuje schopnosti nástroje Tesseract.

2. Teoretická část

2.1. Snímání a získání digitálního obrazu

První operací v řetězci úpravy a zpracování obrazu je jeho nasnímání. Obecně se dá říci, že čím lepší je kvalita prvotního snímku, tím menší úpravy následně musíme provádět a vše se poté zjednodušuje. Pokud se světlo generované zdrojem odráží od nějakého předmětu, vzniká obrazový vjem. Toto odražené světlo se dá zachytit pomocí senzoru jako matice čísel. Každé číslo pak vyjadřuje jeden pixel. Na obrazový vjem poté pohlížíme jako na 2D funkci f(x,*y*), kde funkční hodnota funkce *f* odpovídá jasu obrazu v daném místě a *x, y* jsou prostorové souřadnice. Funkci *f*(*x*,*y*), lze tedy definovat pomocí dvou samostatných funkcí, a to – reflektance r(x,*y*) a iluminace i(x,*y*). Reflektance odpovídá odrazivost objektu a nabývá hodnot 0 až 1, kde minimální krajní hodnota odpovídá úplné absorpci světla a maximální hodnota je úplný odraz.

Abychom mohli obraz dále zpracovávat, musíme jej převést do digitální podoby. To zajistíme pomocí vzorkování a kvantování. Po těchto operacích je obraz sestaven z konečného počtu obrazových elementů – pixelů (z angl. picture element = obrazový prvek). Výsledný obraz je tedy utvořen z operace kvantování, která přiřadí jednotlivým pixelům hodnotu jasu a z operace vzorkování, která obraz rozdělí na konečný počet pixelů.

Obraz můžeme reprezentovat jako matici čísel. Pro f(*x*,*y*) tato matice obsahuje M řádků a N sloupců. Elementárním prvkem v této matici je pixel. U digitalizace je zapotřebí volit prostorové rozlišení i jasové. Každý pixel nese jasovou informaci obrazu. Světlá místa budou obsahovat vyšší čísla a naopak tmavá místa ve scéně čísla nižší [1].

2.2. Předzpracování obrazu

U fotoaparátů a kamer se mohou objevovat vady, které je nutné odstranit před samotným zpracováním obrázku.

Jednou z takových vad je tzv. „soudkovitost“ při použití širokoúhlých objektivů, kde se objeví vada při přenosu přímek. U některých speciálních objektivů je ale tato vada vytvářena záměrně, jelikož tím vzniká zajímavé prostorové zkreslení.

Také se může objevovat jev zvaný „vinětace“. Ten je způsoben tím, že čočka tlumí více paprsky, které svírají velký úhel s optickou osou, než ty, které podél této osy směřují do objektivu. Tímto jevem jsou postiženy jen krajní oblasti, tedy se jím nebudu ve své práci zabývat. Většina novějších fotoaparátů je vybavena automatickou korekcí této vady.

Dalším nežádoucím jevem u obrázku je šum. Ten bývá způsoben nejčastěji nečistotami na čočce nebo elektrickým šumem vznikajícím přímo v kameře.

Cílem předzpracování je tedy odstranit různé druhy zkreslení, potlačit šum či zvýraznit nebo zmírnit jisté rysy obrazu [1].

2.2.1. Bodové jasové transformace

Jasové transformace se dají rozdělit do dvou částí, dle toho zda ovlivňují jediný obrazový bod nebo pro všechny body obrazu:

- Jasové korekce
- Transformace jasové stupnice

2.2.1.1. Jasové korekce

Pro jasovou úpravu konkrétního pixelu výstupního obrazu se použije tento pixel vstupního obrazu [1].

Nejčastěji se tyto korekce používají k odstranění systematických chyb, jako jsou např. nerovnoměrné osvětlení, jiná citlivost digitalizačního a snímacího zařízení, vadné pixely.

Takovéto multiplikativní chyby $e(x, y)$ odstraníme, pokud nad každým bodem obrazu provedeme operaci:

$$f(x, y) = e(x, y).g(x, y) \quad . \tag{1}$$

Určení degradační funkce $e(x, y)$ je jednoduché, stačí jen pořídit obraz o známém průběhu jasové funkce $g(x, y)$ při stálých snímacích podmínkách (nejlépe obraz o konstantním jasu).

Další možností je, že pořídíme obraz s objektem I_0, poté obraz se zakrytým objektivem I_f a obraz za stejných světelných podmínek bez objektu I_b (pro korekci nelinearity snímače). Pak dostaneme vztah:

$$I_c(x, y) = M * \frac{I_0(x, y) - I_b(x, y)}{I_f(x, y) - I_b(x, y)} \quad , \tag{2}$$

kde *M* je konstantou, kterou měníme kontrast výsledného obrazu [1].

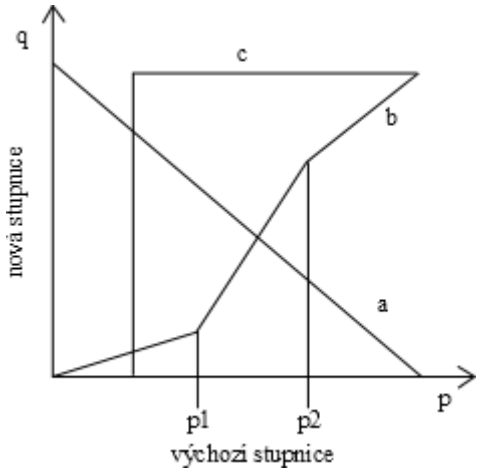
2.2.1.2. Transformace jasové stupnice

Bez ohledu na pozici je určitá hodnota jasu ve vstupním obrazu transformována na jinou hodnotu. Transformace *T* výchozí stupnice *p* na novou stupnici *q* je dána vztahem:

$$q = T(p) \quad .$$

Hodnoty transformační funkce bývají uloženy v jednorozměrném poli a mapování je provedeno s pomocí vyhledávací tabulky (look-up table) [1].

Základními typy jasových transformací jsou např.: a) negativ, b) po částech lineární průběh, c) prahování.



Obrázek 1: Základní typy jasových transformací

Jasové transformace jsou důležité z hlediska úpravy obrazu, pro které zajišťují lepší zvýraznění požadované informace a usnadňují pozdější zpracování. Například po částech lineární jasová transformace realizuje zvýšení kontrastu a prahování dovoluje nastavit různé úrovně šedi výstupního obrazu a tím ovlivnit jeho kontrast. Další hojně používanou transformací je logaritmická transformace, která se používá u zobrazovacích zařízení, jako jsou monitory atd. [1].

2.2.1.3. Prahování

Prahování (z angl. thresholding) [1] je základní metoda segmentace obrazu. Tato metoda se zakládá na hodnocení jasu každého pixelu. U prahování se nastaví určitá hodnota prahu v histogramu a poté všechny hodnoty menší než tento práh patří k pozadí a hodnoty větší k popředí. Jinak řečeno pixely patřící k pozadí jsou vynulovány a pixelům z popředí je přiřazena jednička. Tímto vznikne binární obrázek. Obecně se dá prahování rozdělit do dvou skupin:

- globální prahování
- adaptivní prahování

Při použití globálního prahování je prahovací hodnota použita pro celý obrázek, zatímco u adaptivního prahování je hodnota prahu zjištěna ze zvoleného okolí pixelu.

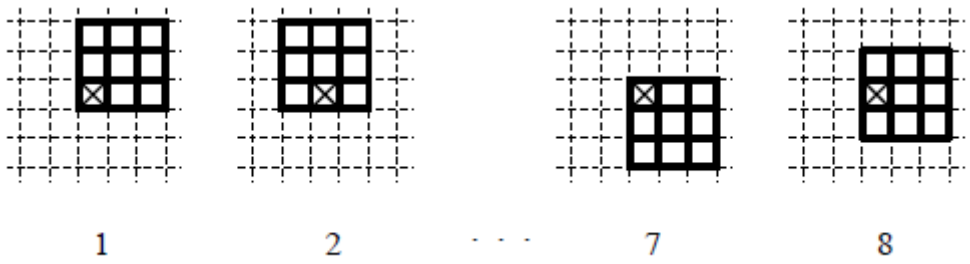
2.2.1.4. Filtrace obrazu

Jak již bylo řečeno, ve fázi předzpracování obrazu je nutné odstranit šum. K této operaci se používají filtry, které je možné rozdělit do dvou základních kategorií:

- lineární
- nelineární

Lineární filtry jsou takové filtry, kdy intenzita upravovaného bodu je dána váhovým průměrem okolních pixelů (které jsou dány filtrační maskou). Lineární filtry typu dolní propust (v analogii s filtrací ve frekvenční oblasti) odstraňují v obraze vysoké frekvence, čímž potlačují šum, ale zároveň s tím také detaily. Tyto filtry se často používají k odstranění zrnitosti. U těchto filtrů je typické, že součet váhových koeficientů v matici je roven 1. Lineární filtry typu horní propust, oproti předcházejícím filtrům, odstraňují nízké frekvence z obrazu, tedy zvýrazňují šum, ale i detaily. Jsou tedy vhodné jen pro některé druhy aplikací. Součet váhových koeficientů v matici u těchto filtrů je roven 0.

Nelineární filtry nepočítají hodnotu intenzity z okolních pixelů, ale pouze vybírají nejlepší hodnotu z okolí a tu pak dosazují do upravovaného bodu. Do těchto metod patří metoda rotující masky (obr. 2) či filtrace s mediánem [2].

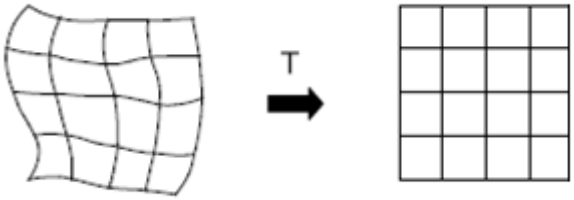


Obrázek 2: Metoda rotující masky[2]

2.2.2. Geometrické transformace

Tyto transformace popisují transformaci nosiče obrazové funkce $f(x, y)$, tedy souřadnic při zvětšení, rotaci nebo jiných operacích.

Geometrické transformace se používají nejčastěji v počítačové grafice, kde se pracuje s objekty ve vektorovém tvaru. Také se zde využívají k odstranění zkreslení vzniklé při pořízení obrazu.



Obrázek 3: Geometrická transformace[2]

Funkcí geometrické transformace je funkce *T*, která zobrazí původní obrazový bod (x, y) do nového bodu x', y'. Je definována vztahy:

$$x' = T_x(x, y), y' = T_y(x, y) \ . \tag{3}$$

Tyto vztahy se dají určit ze znalosti původního i transformovaného obrazu nebo mohou být známy předem, jako je tomu např. v případě rotace či zvětšení.

Pokud chceme určit souřadnice bodu, který je v obraze po geometrické transformaci, musíme použít vztah:

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k , y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k \ , \tag{4}$$

zde transformační vztah souřadnic aproximujeme polynomem *m*-tého stupně. Metodou nejmenších čtverců dokážeme určit koeficienty transformace a_{rk}, b_{rk}. Když se geometrická transformace v závislosti na pozici v obraze příliš nemění, stačí použít v rovnici polynomy nízkého stupně např. $m = 2$ nebo $m = 3$ [2].

2.3. OCR

Neboli optické rozpoznávání znaků (z angl. Optical character recognition) je systém pro převod tištěného nebo psaného textu do takové formy, která je zpracovatelná na PC či jiných zařízeních. Tyto programy vykazují různé stupně úspěšnosti rozpoznání textu v závislosti na kvalitě originálu či nasnímaného obrázku. Téměř vždy je třeba provést korekturu pro různé typy fontu, jelikož OCR program není schopen rozeznat každý font správně. Text pak převádí automaticky nebo si vytvoří vlastní „učební“ program a znaky se sám naučí.

Dnešní OCR programy mají již relativně vysokou úspěšnost převodu textu. Zvláště pak na komerční scéně jsou programy dosti vyspělé a umožňují i vysoký stupeň formátování textu [3].

2.3.1. Základní OCR metody

V této části jsou nastíněny některé základní metody pro rozpoznávání znaků v textu. Tyto metody většinou pracují na porovnávání vstupních dat s daty uloženými v databázi.

2.3.1.1. Template matching algorithm

Neboli algoritmus odpovídajících šablon. Tento algoritmus používá vytvořenou databázi šablon. V této databázi by měla být šablona pro jakýkoliv vstup. Pro rozeznání je šablona vstupu porovnána se všemi šablonami v databázi, přičemž se vyhodnocuje shoda či největší podobnost šablon.

Pokud je $\mathrm{I}(x,y)$ je vstupní znak a $T_n(x,y)$ je šablona n, poté funkce $s(I,T_n)$ vrátí hodnotu, která říká, jak dobře odpovídá šablona z databáze n vstupnímu charakteru.

Rozeznávání znaků je tedy docíleno zjištěním která šablona T_n má nejlepší hodnotu funkce $s(I,T_n)$. Tato metoda je úspěšně použitelná pouze pokud šablony vstupů a databáze mají stejný nebo podobný font. Také se tato metoda dá použít pouze na binární nebo šedotónové znaky. Pro tyto znaky je nejčastěji používaná metoda normalizovaná křížová

korelace (Normalized Cross Correlation – NCC), jelikož je odolná proti změnám v jasu a kontrastu mezi vstupním znakem a uloženou šablonou [4].

2.3.1.2. Template matching v OpenCV

K porovnávání šablon v OpenCV slouží funkce matchTemplate, která nabízí i výběr mezi různými metodami srovnávání šablon. V uvedených vzorcích je I původní obraz, T šablona, R je výsledek porovnávání a x', y' je pozice v šabloně.

SQDIFF:

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \tag{5}$$

SQDIFF NORMED:

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{6}$$

CCORR:

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')) \tag{7}$$

CCORR NORMED:

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{8}$$

CCOEF:

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y')) \tag{9}$$

CCOEF NORMED:

$$R(x,y) = \frac{(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{10}$$

Po skončení porovnávání jsou nejlepší výsledky nalezeny jako globální minima, tj. oblasti s nejmenší intenzitou jsou ty oblasti, které mají nevyšší shodu (metody CV_TM_SQDIFF a CV_TM_SQDIFF_NORMED). U ostatních metod (CV_TM_CCORR, CV_TM_CCORR_ NORMED, CV_TM_CCOEFF a CV_TM_CCOEFF_NORMED) je to naopak. Oblasti, které mají nejvyšší intenzitu, jsou oblasti s nejvyšší shodou [6].

2.3.1.3. CC, NCC a FastNCC algoritmus

V praktické části je u vytvořeného programu použit NCC algoritmus ve funkci matchTemplate z knihovny OpenCV. Je tedy potřeba popsat ho podrobněji.

Korelace mezi dvěma signály je standardní přístup k funkci detekce, stejně tak jako je komponentem dalších sofistikovaných technik. Normalizovaná křížová korelace je narozdíl od běžné korelace zaměřena na prostorovou oblast. Pro vyšší rychlost výpočtu shodných oblastí byly vyvinuty další algoritmy (např. FastNCC) [12].

Křížová korelace

Užití CC pro použití u srovnávání šablon je založena na měření vzdáleností pomocí Euklidovské vzdálenosti.

$$d^2{}_{f,t}(u,v) = \sum_{x,y} [f(x,y) - t(x-u, y-v)]^2 \quad , \tag{11}$$

kde *f* je obrázek a suma prochází přes pozice bodů v obrázku (x, y). V závorce je pak funkce *t* šablona a u, v je pozice v šabloně.

$$d^2{}_{f,t}(u,v) = \sum_{x,y}[f^2(x,y) - 2f(x,y)t(x-u,y-v) + t^2(x-u,y-v)] \quad . \tag{12}$$

Pojem $\sum_{x,y} t^2(x-u,y-v)$ je konstantní. Pokud je pojem $\sum_{x,y} f^2(x,y)$ přibližně konstantní, poté zbývající pojem křížové korelace, který je pro měření podobnosti mezi obrázkem a funkcí.

$$c(u,v) = \sum_{x,y} f(x,y)t(x-u,y-v) \tag{13}$$

Euklidovská vzdálenost

Euklidovská vzdálenost D_E je známá z elementární geometrie. Pokud máme dva body se souřadnicemi $(x_1,x_2),(y_1,y_2)$, pak:

$$D_E((x_1,x_2),(y_1,y_2)) = \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2} \; . \tag{14}$$

Normalizovaná křížová korelace

U křížové korelace je několik nevýhod jejího použití pro porovnávání šablon:

- Velikost $c(u,v)$ záleží a velikosti funkce
- Je neinvariantní ke změnám (např. změnám světelných podmínek)

Korelační koeficient překonává tyto problémy, tím že normalizuje obraz a funkce vektorů na délku jednotky, čímž získá kosinu podobný korelační koeficient.

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\}} \quad , \tag{15}$$

kde $\bar{t}$ je střední hodnota funkce a $\overline{f}_{u,v}$ je v oblasti pod funkcí [12].

Rychlá křížová korelace

Pokud vezmeme čitatel (z rovnice 15) a předpokládejme, že máme obrazy $f'(x,y) \equiv f(x,y) - \overline{f}_{u,v}$ a $t'(x,y) \equiv t(x,y) - \bar{t}$ v kterých již byly střední hodnoty odstraněny:

$$\overset{num}{\gamma}(u,v) = \sum_{x,y} f'(x,y)t'(x-u,y-v) \quad . \tag{16}$$

Pro vyhledávací okno velikosti M^2 a funkci velikosti N^2 vyžaduje přibližně $N^2(M-N+1)^2$ násobení a sčítání.

Rovnice je konvolucí obrazu s obrácenou funkcí $t'(-x,-y)$ se dá vypočítat pomocí vzorce:

$$F^{-1}\{F(f')F\cdot(t')\} \quad , \tag{17}$$

kde F je Furierova transformace. Komplexně sdružená funkce dosáhne obrácení funkce prostřednictvím vlastností Furierovy transformace $Ff\cdot(-x) = F\cdot(\omega)$ [15].

2.3.1.4. Region-Based Invariants

Systém momentů založený na hraničních invariantách, jako je Furierovo popisování, zkoumá jen informace o kontuře – nemohou zachytit vnitřní obsah obrazu. Pro systém momentů založených na regionálních invariantách, jsou všechny pixely obrazu zohledněny pro reprezentaci obrazu. Jelikož tyto invarianty nemají informace jen o ohraničených pixelech, mohou tak zaznamenat více informací obrazu.

Metody založené na momentech jsou nejběžněji používanou formou region-based

invariant. Hu jako první představil set moment-based invariant založených na nelineární kombinaci pravidelných momentů. Metody Hu mají dobré vlastnosti v tom, že jsou invariantní při změnách obrazu, rotaci a změně měřítka. Výpočet vyšších momentů, je ale velmi náročný. Zrekonstruovat obraz z momentů je také velmi náročné [13].

Hu's Seven Moments Invariants

Jsou založené na normalizovaných centrálních momentech. Sedm momentů je definováno níže:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{18}$$

$$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2 \tag{19}$$

$$\phi_3 = (\eta_{30} + 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2 \tag{20}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{21}$$

$$\phi_5 = (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} + \eta_{03}) + (\eta_{21} + \eta_{03}) [(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{22}$$

$$\phi_6 = (\eta_{20} + \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{23}$$

$$\phi_7 = (3\eta_{21} + \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} + \eta_{30})(\eta_{21} + \eta_{03}) [(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{24}$$

Bohužel tyto momenty nejsou invariantní ke změnám jasu [13].

2.4. Algoritmus pro detekci dominantních bodů

Informace o křivce, která je sledována, je soustředěna na extrémy bodů zakřivení, detekce je důležitou součástí metod pro zjišťování kontur tvarů obrazů. Tyto body jsou nazývány tzv. dominantní body. V reálné Euklidovské rovině, je zakřivení definováno jako rychlost

změny mezi tangentou úhlu a délkou oblouku.

$$k = \frac{d\varphi}{ds}. \tag{25}$$

Uzavřenou digitální křivku lze popsat sledem bodů *n* celočíselných souřadnic:

$$C = \{p_i = (x_i, y_i) i = 1,...n\}, \tag{26}$$

kde $p_i + 1$ je sousední k p_i (modulo n). P_i je lineární bod, je bodem zlomu a je kandidátem na dominantní bod[14].

2.4.1. Teh-Chin algoritmus

Tento algoritmus používá poměr vzdáleností d_{ik} mezi P_i a $\overline{P_{i-k}, P_{i+k}}$ (označené jako l_{ik}) ke zjištění regionu podporované oblasti kandidátních bodů:

$$r_{ik} = \frac{l_{ik}}{d_{ik}}. \tag{27}$$

Pro odhad délky podporované oblasti, proces začne s $k = 1$, k se poté zvyšuje o jedna dokud není splněna jedna z následujících podmínek, poté *k* určuje délku podporované oblasti v bodě P_i.

1. $l_{ik} \geq l_{ik+1}$
2. $r_{ik} \geq r_{ik+1}$ pro $d_{ik} \geq 0$

   $r_{ik} \leq r_{ik+1}$ pro $d_{ik} \leq 0$

Tento poměr může být považován jako ekvivalent měřítka pro zakřivení. Po vypočítání podporovaného regionu pro každý bod P_i, je určeno významné měřítko $|S(P_i)|$. Poté se potlačí nemaximální hodnoty $|S(P_i)|$.

Teh-Chin algoritmus není odolný vůči šumovým konturám, jelikož lokální maxima zakřivení mohou být způsobeny variací šumů na křivce [14].

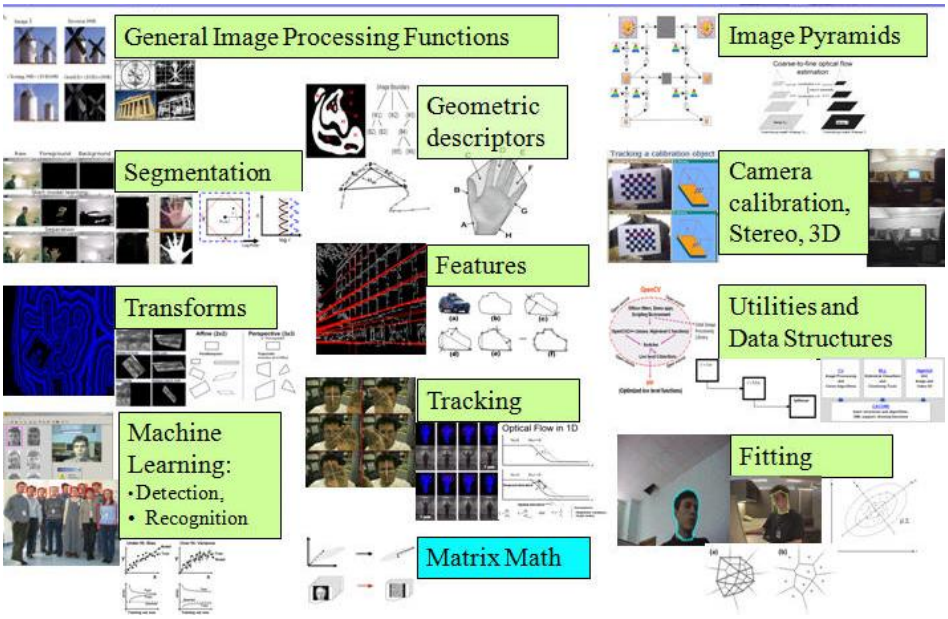
2.5. OpenCV

OpenCV (z angl. Open Source Computer Vision) je otevřená knihovna pro práci s obrazem. Je zaměřena na práci s obrazem v reálném čase. K úspěšnému používání těchto knihoven je nutné znát význam jednotlivých příkazů a jejich syntaxi. Tyto informace se nalézají v četné dokumentaci, knihách a článcích. Tato dokumentace je velice rozsáhlá a není téměř možné se všechny příkazy naučit. Proto je lepším přístupem ten, při kterém víme, jaké operace si přejeme s obrazem (videem) udělat a jaký by měl být výstup. Samozřejmě tento přístup vyžaduje znalosti o práci s obrazem. Podle těchto nároků pak stačí si v dokumentaci najít i potřebnou funkci.

Ve starších verzích knihoven (např. OpenCV 2.0) bylo použito pouze syntaxí programovacího jazyka C, kde se pracovalo se strukturami iplImage (Intel Image Processing Library). V novějších verzích je od tohoto upuštěno a používá se programovací jazyk C++, ve kterém se používají struktury typu Mat. Hlavním rozdílem mezi iplImage a Mat, je v tom, že při použití Mat nemusíme ručně alokovat a dealokovat paměť, Mat to dělá za nás automaticky. Samozřejmě je možné konvertovat iplImage do Mat.

K používání knihoven OpenCV je nutné zavést je do programovacího prostředí. Tato procedura je popsána v četných návodech na internetu [6].

Knihovna OpenCV se dnes nejčastěji v praxi používá např. k rozpoznávání obličejů či registračních značek automobilů (další použití na obr. 5). Díky OpenCV je na vzestupu oblast počítačového vidění, kde tato knihovna umožňuje snadný přístup ke všem potřebným nástrojům. Nyní má knihovna již přes 2500 algoritmů a 500 funkcí. Podporuje rozhraní běžící na platformách Windows, Linux, Android a Mac. Využívat se dá v programovacím prostředí C++, C, Python... [6] Přehled základních funkcí knihovny OpenCV je na obr. 4.



Obrázek 4: OpenCV[6]

2.6. Tesseract

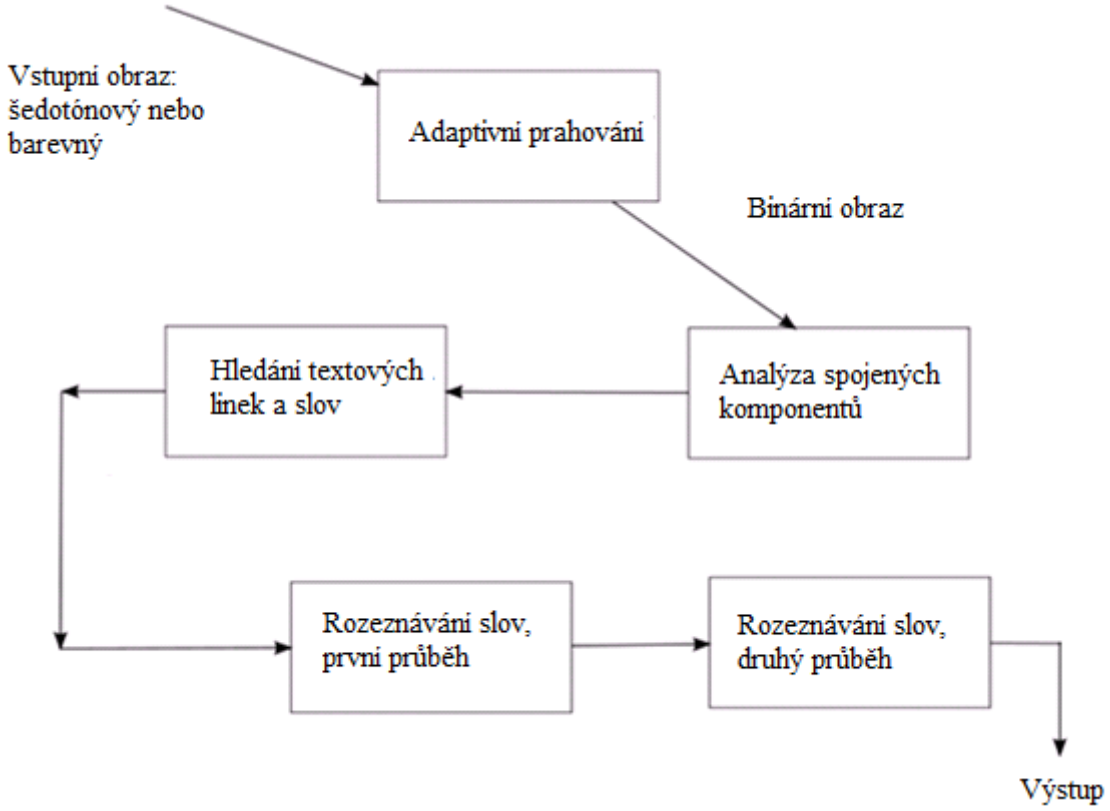
Tesseract je nejspíše jeden z nejvíce přesných volně dostupných OCR nástrojů. Je třeba si říci, že tento nástroj není celý program. V kombinaci s obrazovou knihovnou pro processing, Leptonica, umožňuje čtení mnoha formátů obrazu, fontů a jejich následných převod do textu. Umí převádět více než do šedesáti jazyků (včetně češtiny). Dnes je sponzorován firmou Google a jejich aplikace jsou v trojici nejlepších OCR programů od roku 1995. Podobně jako knihovny OpenCV podporuje platformy Linux, Windows, Mac OSX, Android [9].

2.6.1. Postup při rozeznávání textu

1) Úvodní linky (Outlines) jsou analyzovány a uloženy.
2) Linky jsou shromážděné dohromady jako bloby.
3) Bloby jsou organizovány do linek textu.
4) Linky textu jsou rozděleny na slova.
5) Při prvním části průběhu programu se proces pokusí rozeznat každé slovo.
6) Slova, která mají uspokojující kvalitu, přejdou do adaptivní tréninkové části.

7) Výstupy, které vznikly z adaptivního tréninku, přechází do druhé části programu. Ten se pokusí rozeznat slova, jenž nebyli rozeznané během prvního průběhu.
8) Zde jsou vyřešeny neostré mezery a přidávají se kapitálky.
9) Výstup.

2.6.2. Algoritmy nástroje Tesseract



Obrázek 5: postup nástroje Tesseract

2.6.2.1. Hledání řádků a slov

Algoritmus pro vyhledávání řádků je uzpůsoben tak, že pokud máme pootočený obraz, algoritmus nalézá řádky v originální pootočené formě a tímto nedochází ke ztrátě kvality. Hlavní částí procesu je „blob filtering“ a sestavování řádek [9].

2.6.2.2. Vhodnost základních čar

Pokud už jsou jednou nalezeny řádky. Usazují se základní čáry. Základní čáry jsou usazeny s větší přesností pomocí rozdělení blobů.

Obrázek 6: Příklad usazené zakřivené základní čáry[9]

Tento obrázek je ukázkou řádky textu spolu s usazenými základními čarami a s určujícími čarami sklonu.

2.6.2.3. Detekce pevné rozteče písmen a jeho rozdělení

Tesseract testuje řádky textu pro zjištění pevných úseků. Pokud najde úsek, kterým je slovo, rozdělí slovo v jednotlivé znaky. Tyto znaky mají také fixní rozteč.

Obrázek 7: Text s pevnou délkou rozteče

2.6.2.4. Hledání proporcionálních slov

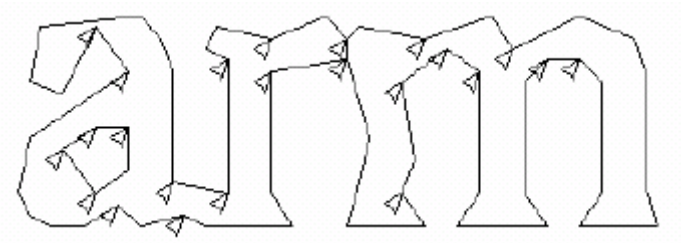
Dalším krokem je hledání takových znaků, která nemají pevnou rozteč a není u nich možné rozdělit je pomocí horizontální čáry. Pokud máme překrývající se znaky, Tesseract toto řeší tak, že změří mezery mezi základními čarami a čarami sklonu jednotlivých znaků. Místa, která jsou blízko prahu, jsou v této fázi rozmazány a hlavní rozhodnutí v rozpoznání je vykonán až po rozeznání slov [9].

2.6.2.5. Rozeznání slov

Částí rozpoznávacího procesu pro každý nástroj pro rozeznávání znaků je jak by slova měli být rozděleny na znaky. Prvním procesem je klasifikace řádku, který byl rozeznán během určování řádků.

2.6.2.6. Rozdělení spojených znaků

Pokud výsledek z rozpoznaného slova je neuspokojující, tedy místo jednotlivých znaků je několik spojených. Tesseract se pokusí zlepšit výsledek rozdělením blobů s nejhorší klasifikací znaků. Pomocí konkávních vrcholů polygonální aproximace jsou nalezeny kandidátní rozdělující body. Pokud mají znaky spojené úseky, je možné, že bude potřeba až šest párů rozdělujících bodů.



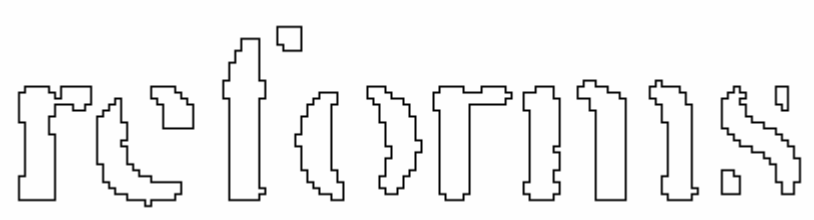
Obrázek 8: Spojené znaky s kandidátními body[9]

Tento obrázek ukazuje kandidátní body pomocí šipek a rozdělující linku v části, kde se stýká „r“ a „m“ [9].

2.6.2.7. Spojování rozbitých znaků

Pokud jsou vyčerpány potenciální rozdělující znaky a slovo není stále kvalitní, je předáno do *associatoru*. Associator vykoná hledání v segmentačním grafu možných kombinací maximálně rozdělených blobů v kandidátních znacích. Toto dělá bez stavění aktuálního segmententačního grafu, ale místo toho si vytvoří hash tabulku viděných stavů. Vyhledávání pokračuje vybíráním nových kandidátních stavů z prioritní fronty. Vyhodnocuje je pomocí klasifikací neklasifikovaných kombinací fragmentů.

Tento přístup, ale nemusí plně efektivní, v nejhorším případě může minout důležité rozdělená místa. Výhodou tohoto přístupu je, že zjednoduší datové struktury, které mohou být potřeba k vytvoření celého segmentačního grafu [9].



Obrázek 9: Lehce rozpoznatelný text[9]

2.6.2.8. Statická klasifikace znaků

Klasifikační proces má dvě části. V prvním kroku rozdělovač tříd vytvoří seznam tříd znaků, které by mohli odpovídat neznámým. Každý prvek načte z nahrubo kvantizované vyhledávací tabulky bit - vektor tříd, které by se mohli shodovat. Tento bit - vektor je přičten ke každé třídě. Třída s nejvyšším počtem se stane seznamem pro další krok. Každý prvek vyhledá bitový vektor z prototypů přidělené třídy, která by mohla odpovídat a jsou vypočítány podobnosti. Každý prototyp znaku třídy je reprezentován expresí - logickým součtem, který se nazývá konfigurace. Výpočetní proces ukládá podobnosti každého prvku a prototypu. Nejlepší kombinace odstupu, která je vypočtena z prototypu a součtové funkce, je označena jako nejlepší konfigurace třídy [9].

2.6.2.9. Trénování dat

Vzhledem k tomu, že klasifikátor schopen jednoduše rozeznat poškozené znaky, klasifikátor nemusí mít trénování poškozených znaků. Klasifikátor má trénování na pouhých dvacet vzorků, devadesáti čtyřech znaků, osmi fontů jednotné velikosti, ale má čtyři atributy (normal, bold, italic, bold italic) [9].

2.6.2.10. Analýza jazyka

Tesseract obsahuje relativně málo jazykové analýzy. Kdykoliv modul na rozpoznávání slov obsahuje nový segment, modul analýzy jazyka vybere nejlepší možný řetězec slov v každé jedné z následujících kategorií :

- Nejčastěji frekventované slovo.
- Nejčastější slovo ze slovníku
- Nejčastější numerické slovo
- Nejčastější slovo s velkými písmeny
- Nejčastější slovo s malými písmeny
- Slovo s nejlepším výsledkem klasifikace

2.6.2.11. Adaptivní klasifikátor

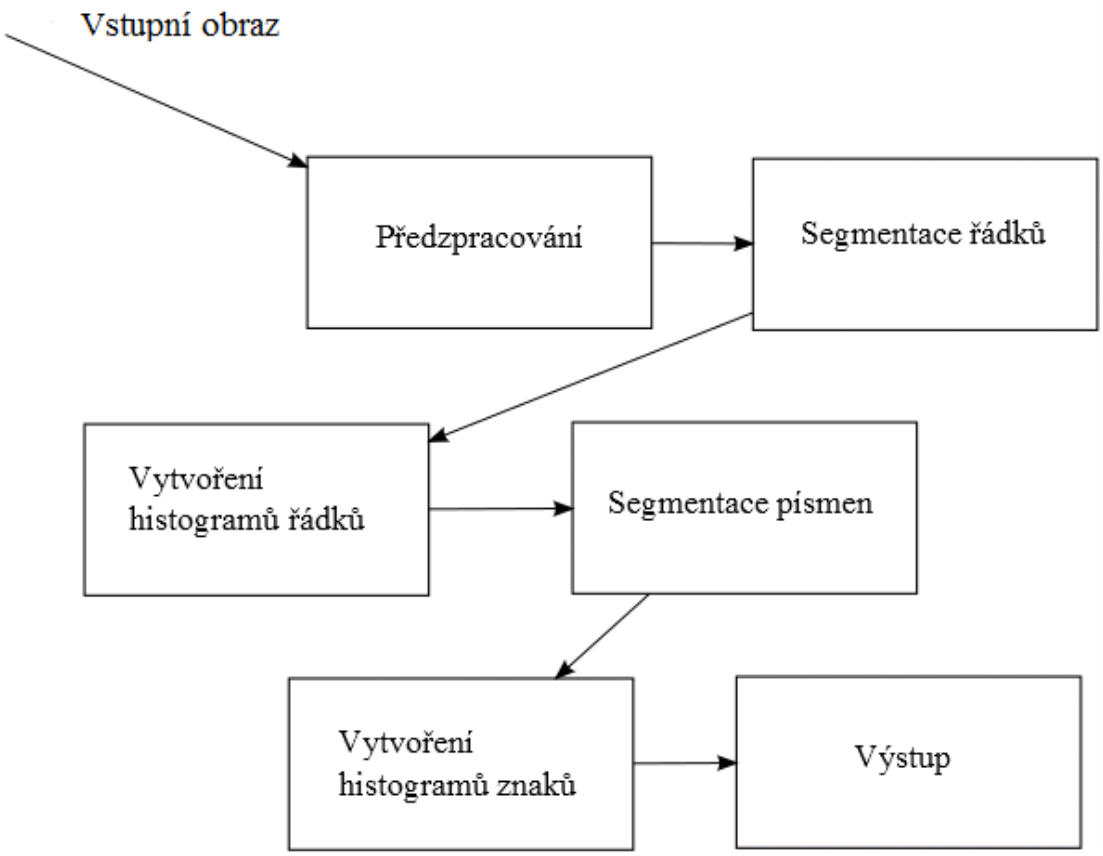
Vzhledem k tomu, že statický klasifikátor je dobrý k zobecňování jakéhokoliv typu fontu, jeho schopnost rozlišovat mezi jinými znaky nebo znaky a „neznaky“ je slabá. Adaptivní klasifikátor je více citlivý na font a je trénován pomocí výstupu ze statického klasifikátoru. Je používán pro získání větší schopnosti rozlišovat v dokumentech, kde je omezený počet fontů.

Jediným podstatným rozdílem mezi statickým a adaptivním klasifikováním, mimo trénování znaků, je adaptivní klasifikátor používá izotropickou normalizaci základní čáry [9].

3. Praktická část

4. Program 1

První program používá k předzpracování obrazu funkce z knihovny OpenCV. Počítání řádků a písmen je zde vytvořeno pomocí jednoduchého počítání nenulových pixelů. Tento postup je zvolen jen pro ukázku základního počítání pixelů a jejich následovné použití.



Obrázek 10: Postup při tvoření programu 1

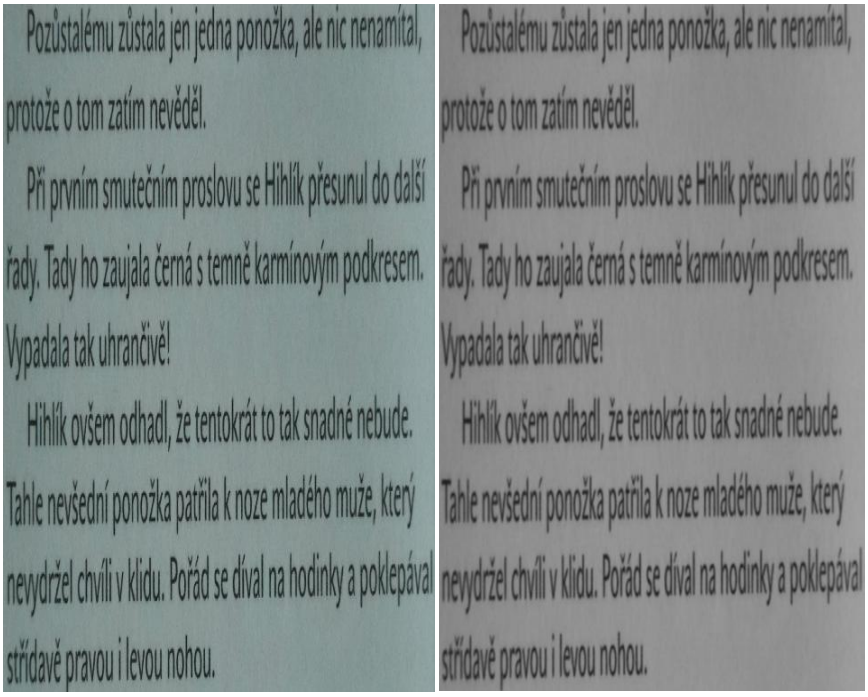
4.1. Předzpracování obrazu

V sekci předzpracování obrazu jsou použity takové funkce, které mají za úkol upravit originální obraz na výstupní upravený. Výstupní obraz by pak měl být použitelný pro další úkony.

V prvním kroku jsem u obrázku změnil velikost na 1024x768 pixelů. Důvodem byla standardizace pro další použití. Dále jsem obrázek konvertoval na šedotónový.

4.1.1. Vyhlazení šumu

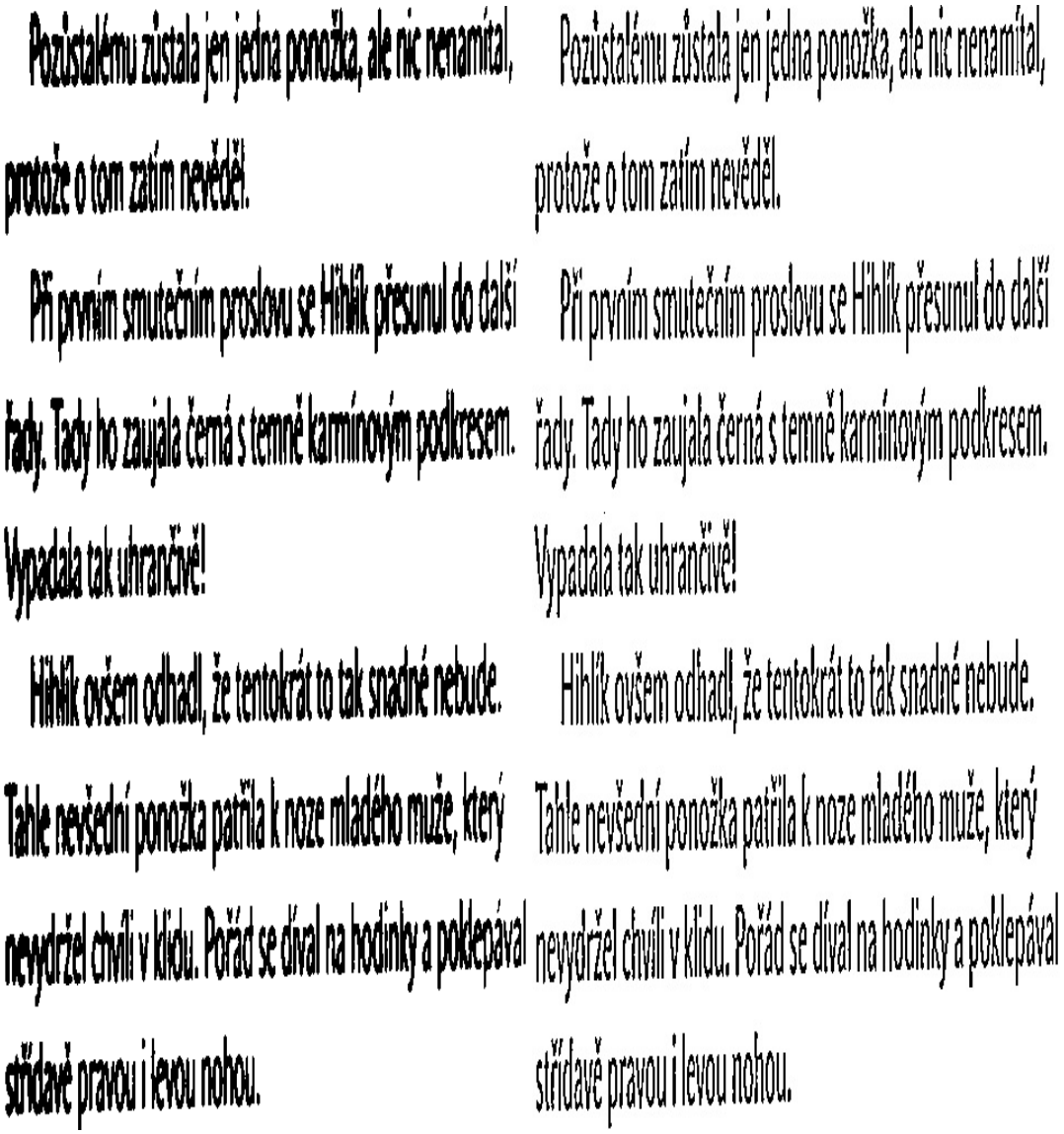
Pro vyhlazení šumu jsem použil filtr s maskou 3x3 pixelů, jelikož má dobrý poměr vyhlazení šumu a rozostření obrázku. Obraz před použitím filtru a po jeho použití a převodu na šedotónový jsou na obr. 11.



Obrázek 11: Porovnání obrázků bez vyhlazení a s vyhlazením

4.2. Prahování

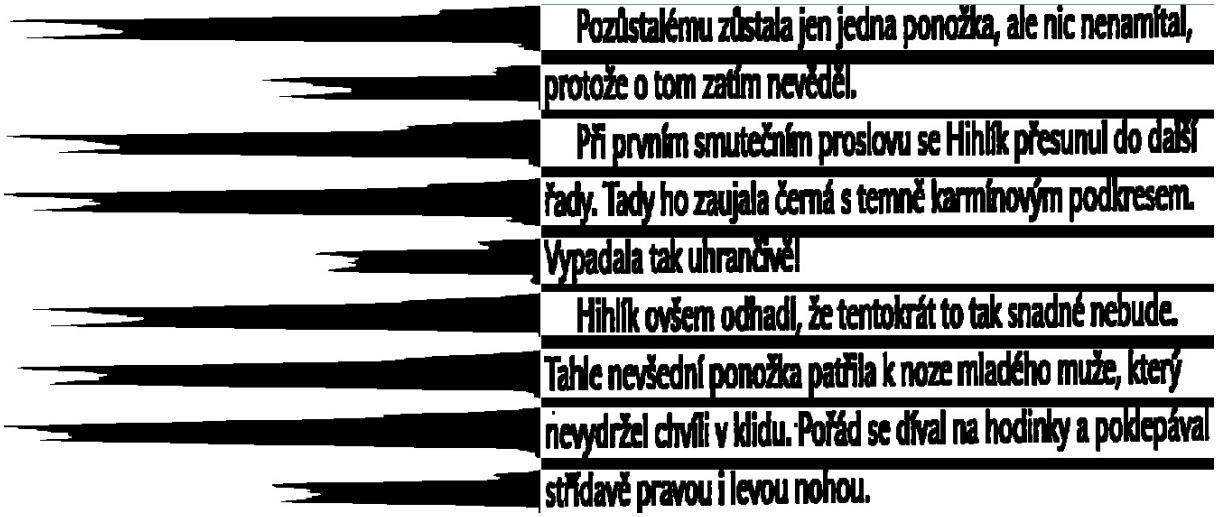
Výstupem prahování by měl být binární obraz, kde je jasně oddělené popředí od pozadí. Testovány byly jak globální, tak i adaptivní metody. Na konci testů jsem zvolil adaptivní prahování s velikostí testovaného okolí 5x5 pixelů, jelikož oproti globálním metodám dobře reagovalo na změnu podmínek při pořizování snímku (jas, barevnost světla, blesk fotoaparátu). Srovnání výstupů jsou na obr. 12.



Obrázek 12: Porovnání globálního a adaptivního prahování

4.3. Segmentování řádků

Úkolem této části je nalézt a spočítat řádky v textu. Základním krokem bylo spočítat počet černých bodů v obrázku. Toho jsem docílil tak, že jsem použil funkci, která počítá sumu pixelů v řádcích. Tedy pokud máme např. v řádku 5 černých bodů a 2 bílé body, hodnota tohoto řádku je 5*0+2*255=510. Tyto hodnoty jsem vynesl do grafu. Poté jsem nastavil pevnou hodnotu prahu, který v daném místě vertikálně procházel grafem a odečítal hodnotu jednotlivých pixelů. Pokud narazil na hodnotu 0 (černý pixel) vykreslil horizontální čáru do výstupního obrázku. Dále jsem to samé s drobnou úpravou udělal ve výstupním obrázku a jednoduše tak odečetl počet řádků. Problémem tohoto postupu bylo, že pokud počet černých bodů v histogramu překročil danou mez, ale nejednalo se o řádek (indexy, čárky, spodní části písmen atd.) vykreslila se horizontální čára i zde a výsledek byl zkreslen. Z tohoto důvodu jsem použil funkci eroze [7], která mi jednotlivé řádky „slila“ v celek a vše již fungovalo správně. Výsledek i s histogramem je na obr. 13.



Obrázek 13: Segmentace řádků

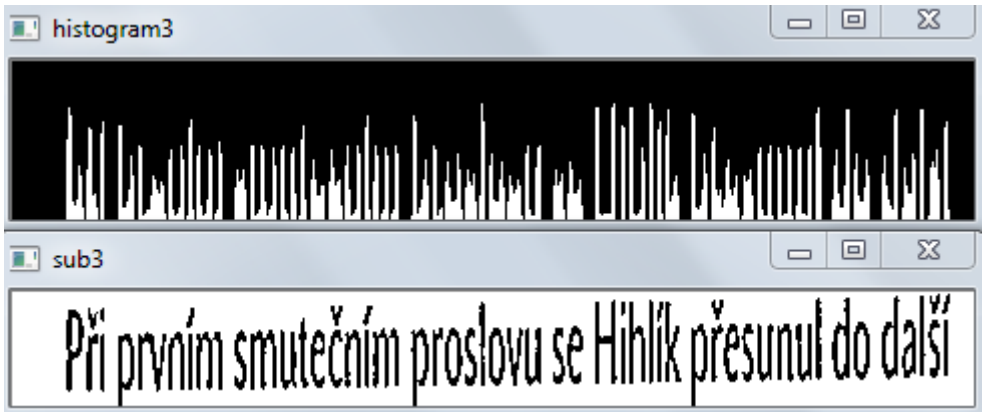
4.4. Segmentování znaků

Segmentování znaků probíhalo obdobně, jako tomu bylo u řádků, jen s několikerými úpravami. Nejdříve jsem musel odstranit ono slití částí textů, jelikož by to mělo za následek

velkou chybovost u pozdějšího určování znaků. Tedy u výstupního obrázku použil pouze adaptivní prahování bez eroze. Dále jsem zjišťoval velikost řádku, a to pouze tím že jsem kontroloval černé a bílé body ve výstupním obrázku. Pomocí této velikosti jsem vytvořil tzv. „oblast zájmu“ (z angl. Region of Interest) pro každý řádek a v každém řádku opět provedl vyčítání histogramu a z něj celkový počet znaků. Počty znaků v každém řádku se nakonec sečtou a zobrazí ve výstupní konzoli. Vysegmentované řádky i s jejich histogramy jsou na obr. 14-22.

Obrázek 14: Segment 1

Obrázek 15: Segment 2



Obrázek 16: Segment 3

Obrázek 17: Segment 4

Obrázek 18: Segment 5

Obrázek 19: Segment 6

Obrázek 20: Segment 7

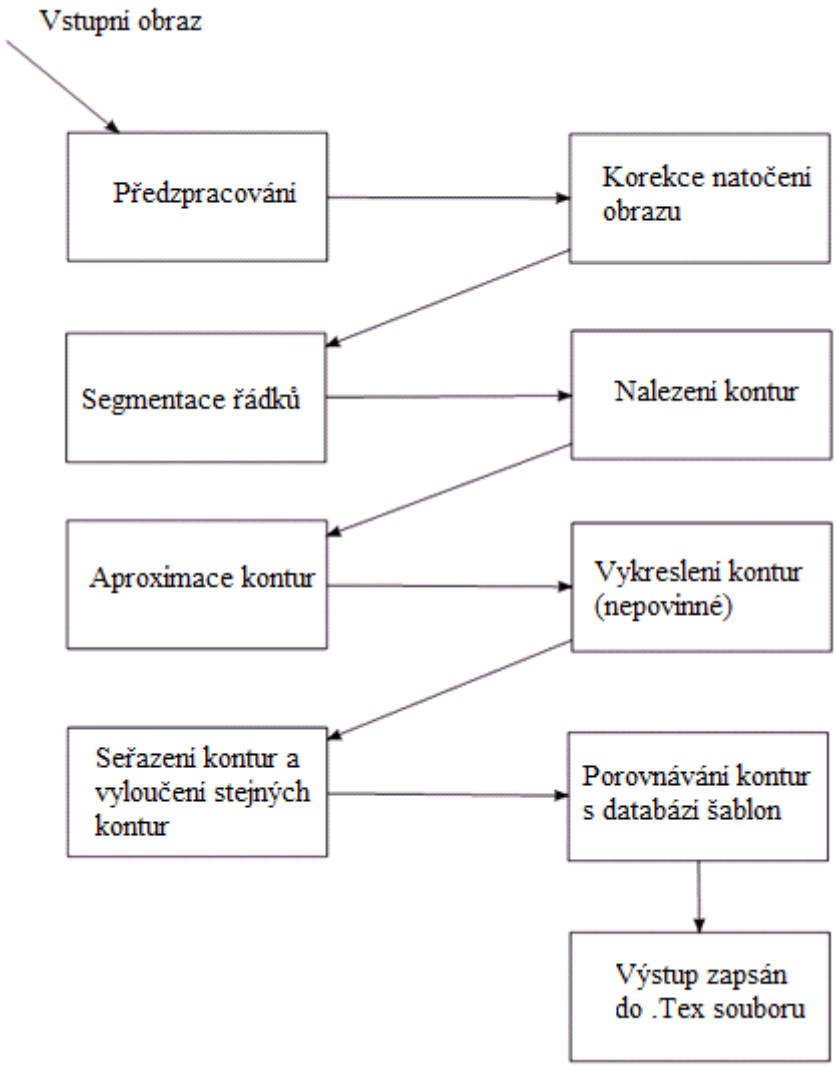
Obrázek 21: Segment 8

Obrázek 22: Segment 9

Při určování řádků a znaků je realizováno „hrubou silou“. K rozpoznávání textů není vhodný, a proto jsem vytvořil druhý program, který problém řeší lépe.

5. Program 2

Druhý program je už o něco náročnější vzhledem k použitým funkcím. Předzpracování je podobné jako u prvního programu, ale další funkce jsou již složitější. K rozeznávání jednotlivých písmen slouží funkce pro zjištění kontur v obraze. U každého písmene je tedy zjištěna kontura a ta je následně (nepovinně) orámována. Tento program je navíc obohacen o funkci, která koriguje natočení obrazu. Po těchto úpravách je připojena funkce k rozeznání jednotlivých znaků pomocí porovnávání šablon a zapsání výstupu do souboru.

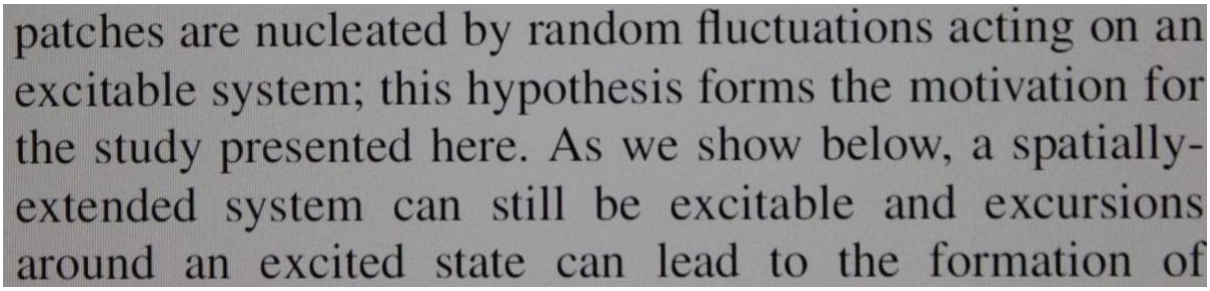


Obrázek 23: Postup při tvoření programu 2

V tomto programu se vstupní obraz načítá jeho jménem z konzole. K předzpracování jsem použil podobné funkce jako v předchozím případě. Dále bylo nutné korigovat natočení obrazu. Toho jsem docílil tak, že jsem našel ohraničující box obrazu a jeho vrcholy, a poté obraz boxu korigoval pootočením boxu dle jeho středu.

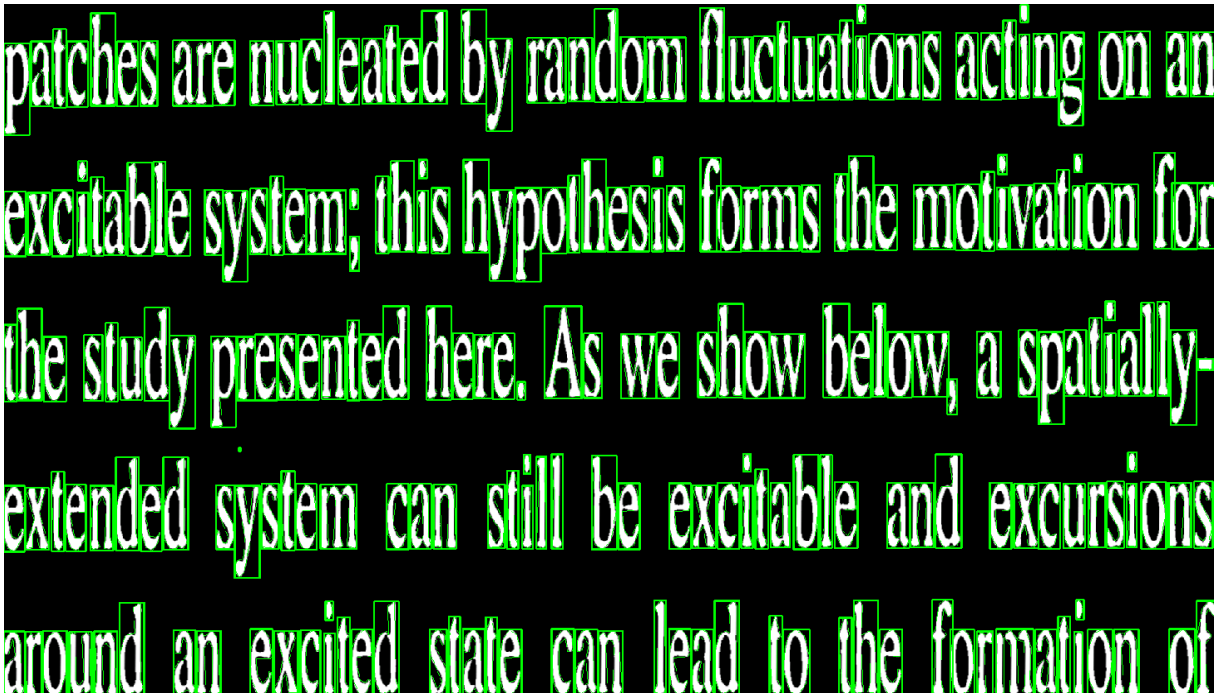
Problémem u této korekce bylo správné korigování, jelikož pokud bylo zkosení u obrazu pod úhlem menším jak -45°, tak úhel referenčního obdélníku má výšku větší než šířku. Pokud je tedy takovýto výstupní úhel, pro správné vyhodnocení stačí po všech úpravách prohodit výšku za šířku.

5.1. Vstupní obraz 1



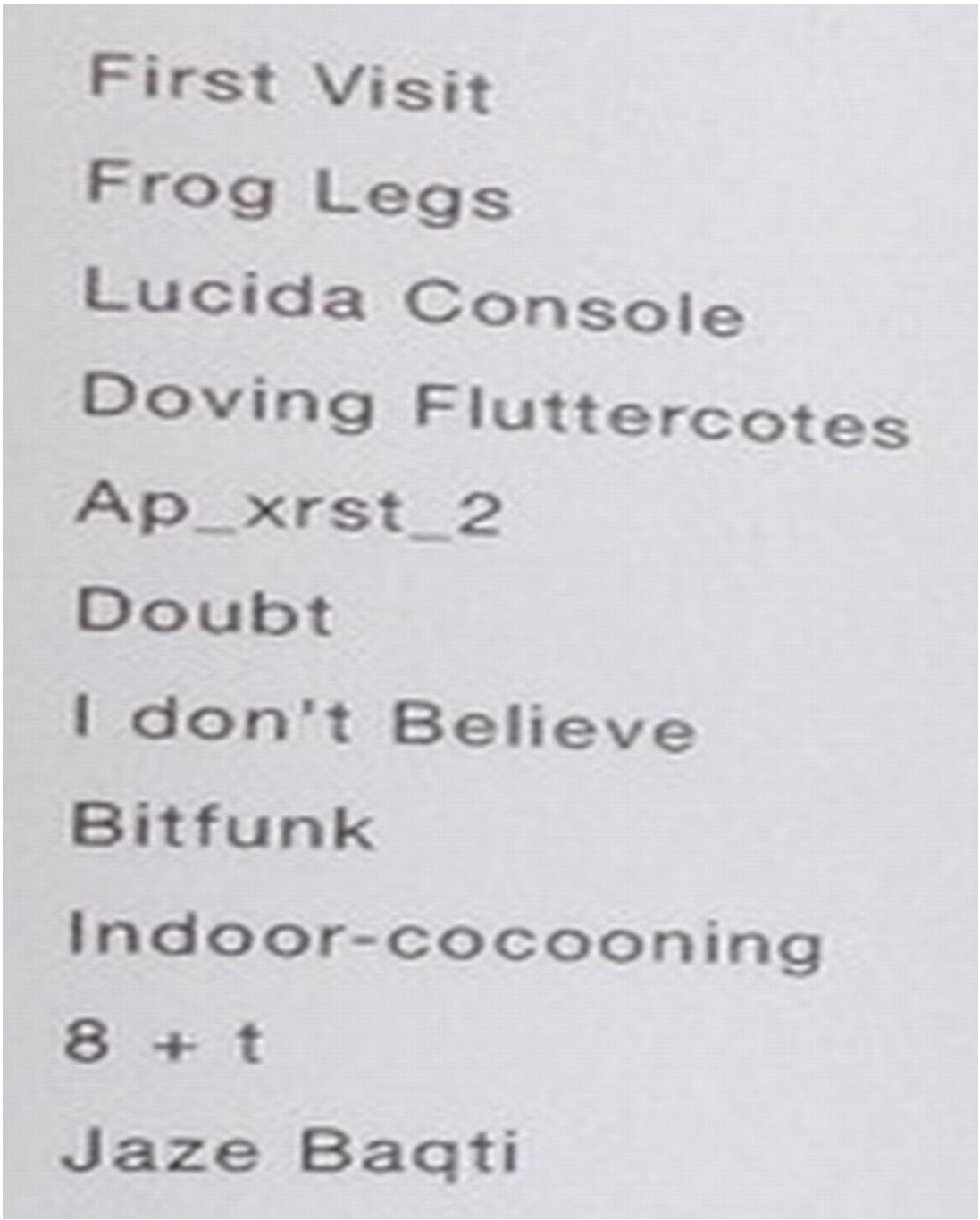
Obrázek 24: Vstupní obraz 1

Dále jsem použil pro vyhledávání kontur funkci findContours se zjišťováním jen vnějších extrémů kontur spolu s algoritmem Teh-Chin (kap. 2.4.1). Poté jsem použil funkci approxPolyDP, která přiblíží křivku k jiné křivce s méně vrcholy, takže vzdálenost mezi nimi je menší nebo stejná dle dané přesnosti. Po tomto kroku se kontury vykreslily a okolo každé kontury se vykreslil obdélník. Další vstupní obrazy jsou uvedeny níže (obr. 26, 28, 30).

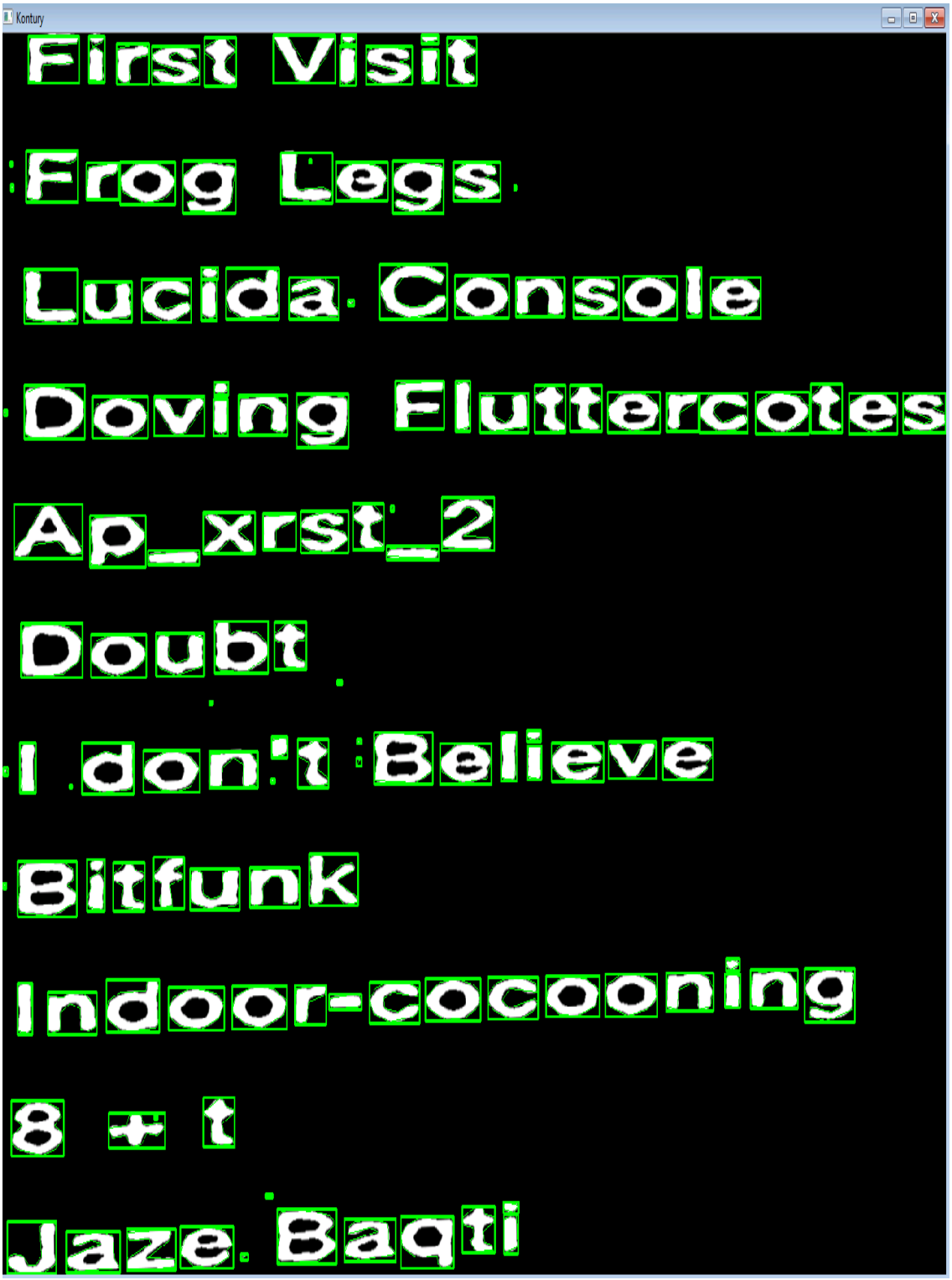


Obrázek 25: Výstup obrazu 1 s vyznačenými konturami

5.2. Vstupní obraz 2

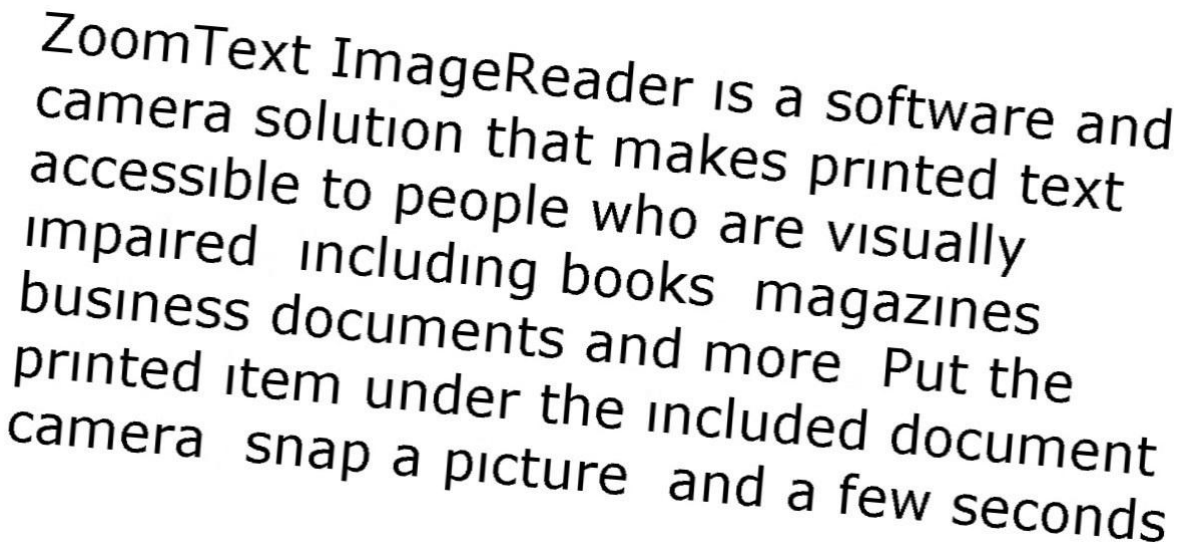


Obrázek 26: Vstupní obraz 2

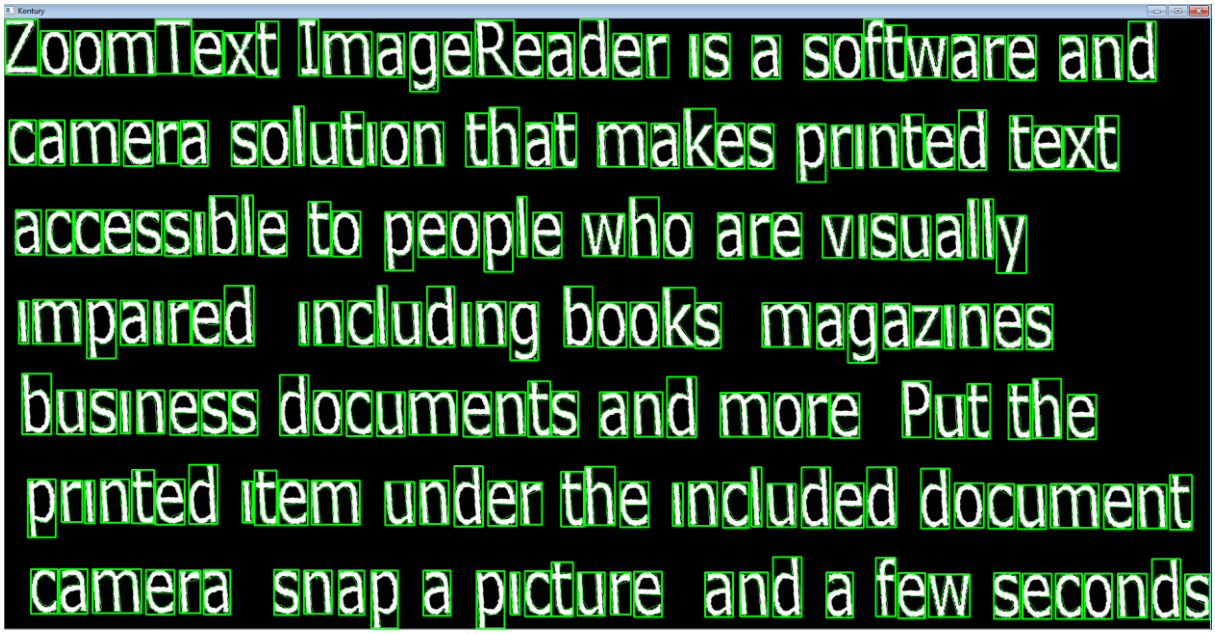


Obrázek 27: : Výstup obrazu 2 s vyznačenými konturami

5.3. Vstupní obraz 3

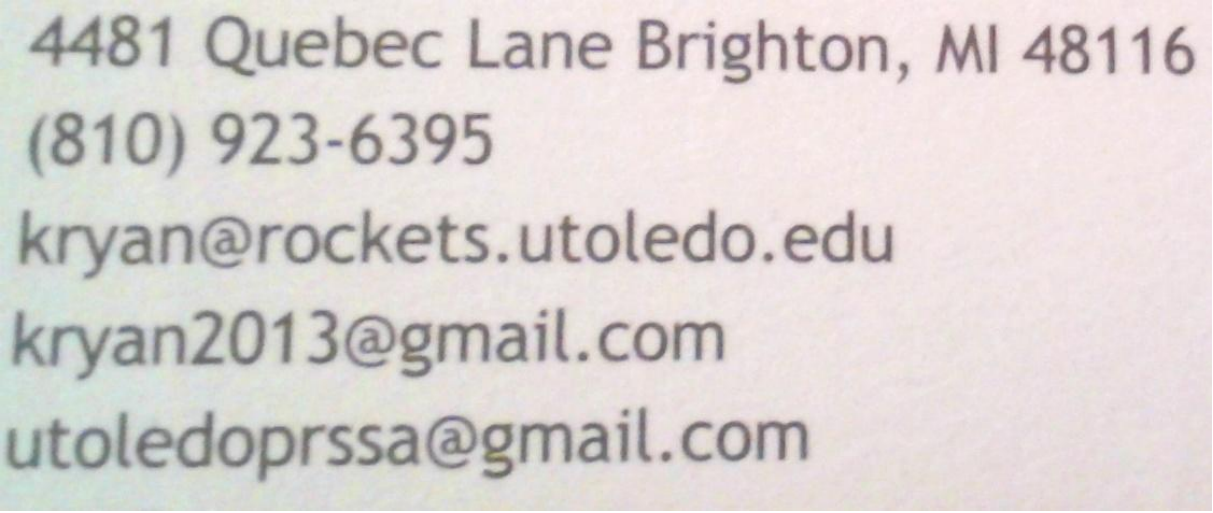


Obrázek 28: Vstupní obraz 3



Obrázek 29: Výstup obrazu 3 s vyznačenými konturami

5.4. Vstupní obraz 4



Obrázek 30: Vstupní obraz 4

Obrázek 31: Výstup obrazu 4 s vyznačenými konturami

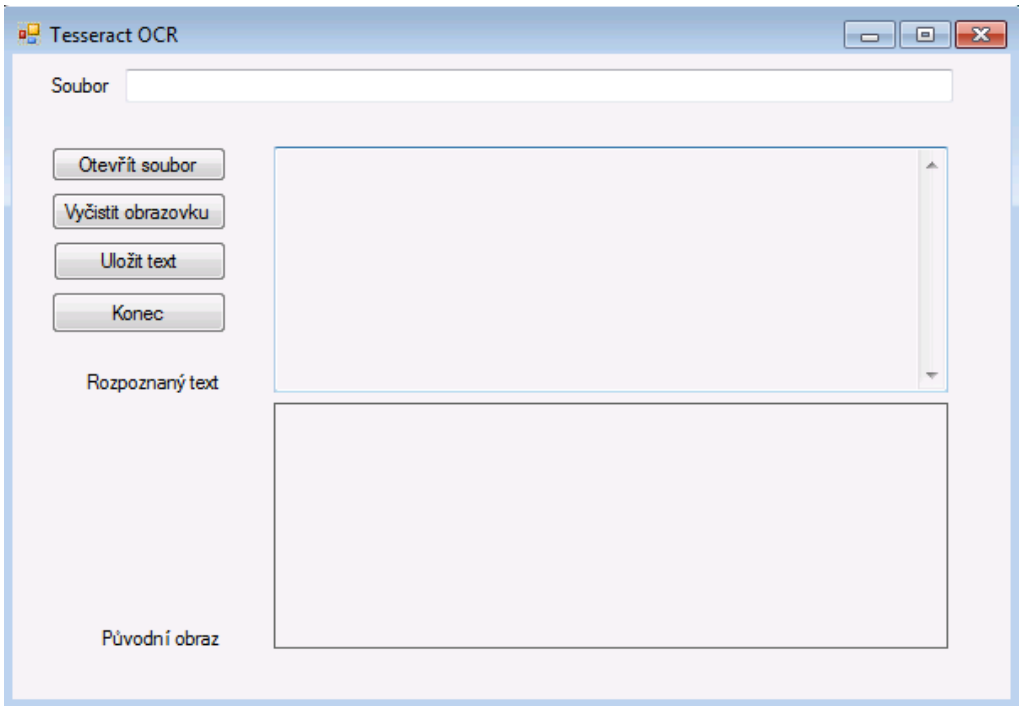
V dalším kroku jsem seřadil a vytřídil jednotlivé kontury pomocí příkazů sort a erase. Po všech úpravách je konečně použito metody matchTemplate (kap. 2.3.1.2), která srovnává nalezené kontury a šablony jednotlivých písmen. Jednotlivým konturám je přiřazen vlastní RoI. Písmeno s nejlepším výsledekem shody je pak zapsáno do výstupního .Tex souboru.

Metoda srovnávání se stává méně spolehlivou při klesající kvalitě obrazu. Nevýhodou je, že pro každý obraz a font je nutno vytvořit samostatnou knihovnu šablon. Dále je nutné u každého obrazu zadat vlastní prahovací úroveň. Jak předzpracování, tak pevnou hodnotu podle které se sleduje výsledek porovnávání. Výhodou naopak je, že tato metoda je invariantní vůči změnám jasu pořizovaného obrazu. Další možnou metodou pro rozpoznávání textu je určování momentů pomocí Hu metody. Tato metoda je náročnější na výpočet, ale má mnoho dobrých vlastností (kap. 2.3.1.4). Nevýhodou je pak to, že výsledek snižují jasové podmínky při pořizování obrazu.

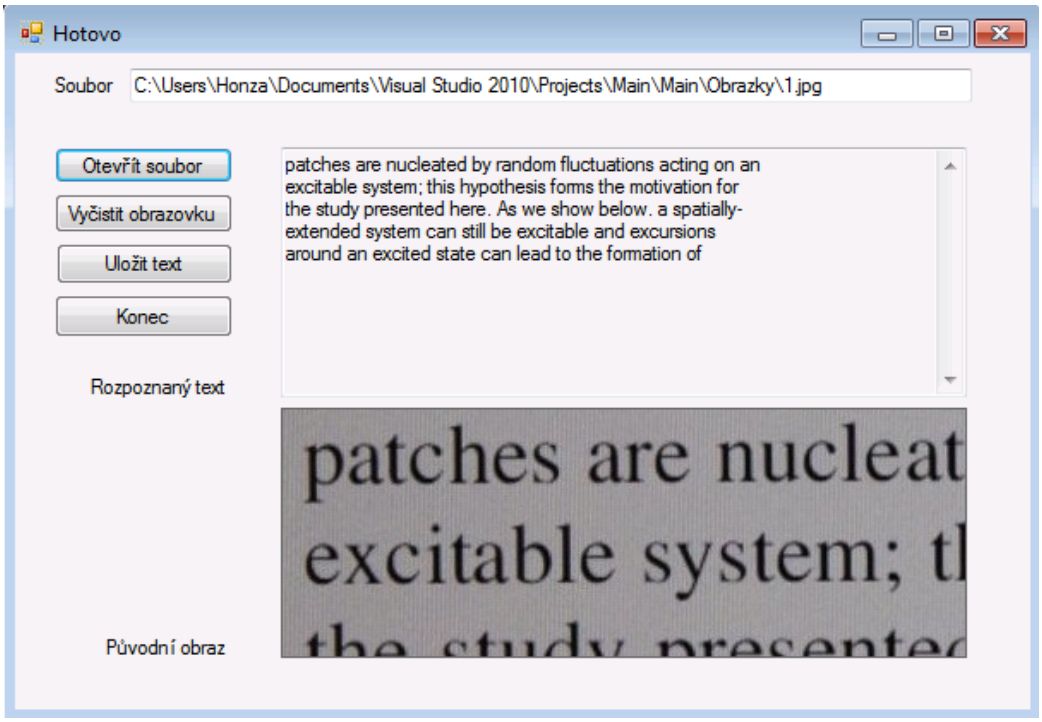
Jak je z výstupních obrazů vidět, i přes dobré rozeznání znaků jsou stále slitá spojení jako „vy“ nebo „ky“. Pokud snížíme hodnotu adaptivního prahování, spojení se přeruší, ale dochází k rozpadu některých písmen. Je tedy nutné volit takovou hodnotu, která je kompromisem mezi těmito vlastnosmi. Taktéž je patrno, že pokud musíme korigovat vysoké natočení obrazu, obraz poté ztrácí kvalitu. Proto moderní knihovny Tesseract při rozpoznávání textu natočení nekorigují.

6. Program 3

Tento program předvádí zavedení knihoven Tesseract do aplikace a jejich vysokou účinnost na většinu obrazů. Pro zavedení stačí pouze uvést cestu ke knihovnám, přidat funkce jednotlivých tlačítek a boxů. Všechny funkce od předzpracování až po trénování (kap. 2.6) jsou již obsažené v knihovnách. Knihovny k programu jsou anglické, proto je možné rozpoznávat jen text bez diakritiky. Tento program také umožňuje zápis textu do souboru.



Obrázek 32: Hlavní okno aplikace

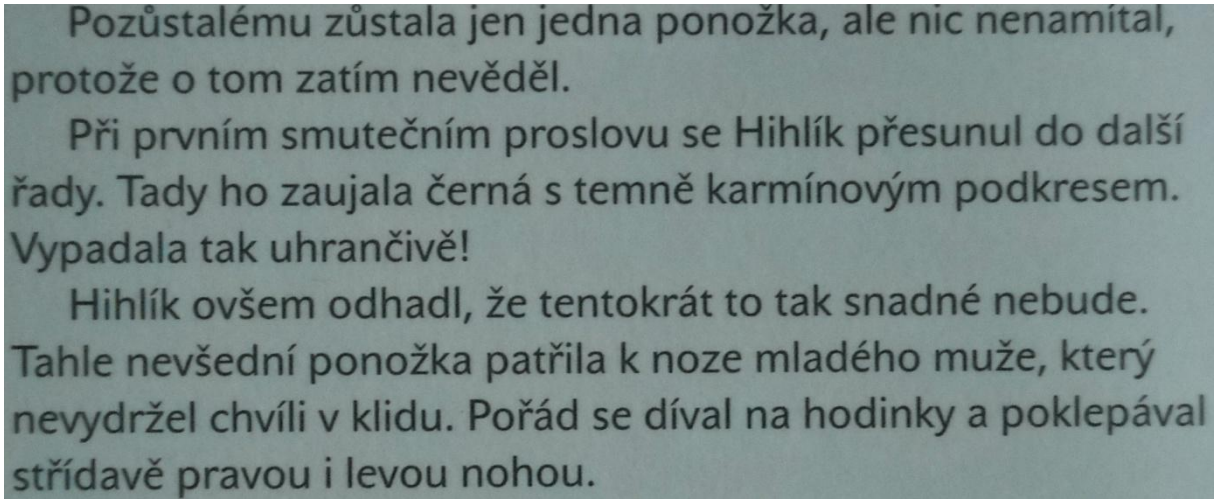


Obrázek 33: Aplikace s rozeznaným textem

7. Zhodnocení výsledků

7.1. Testované obrázky – program 1

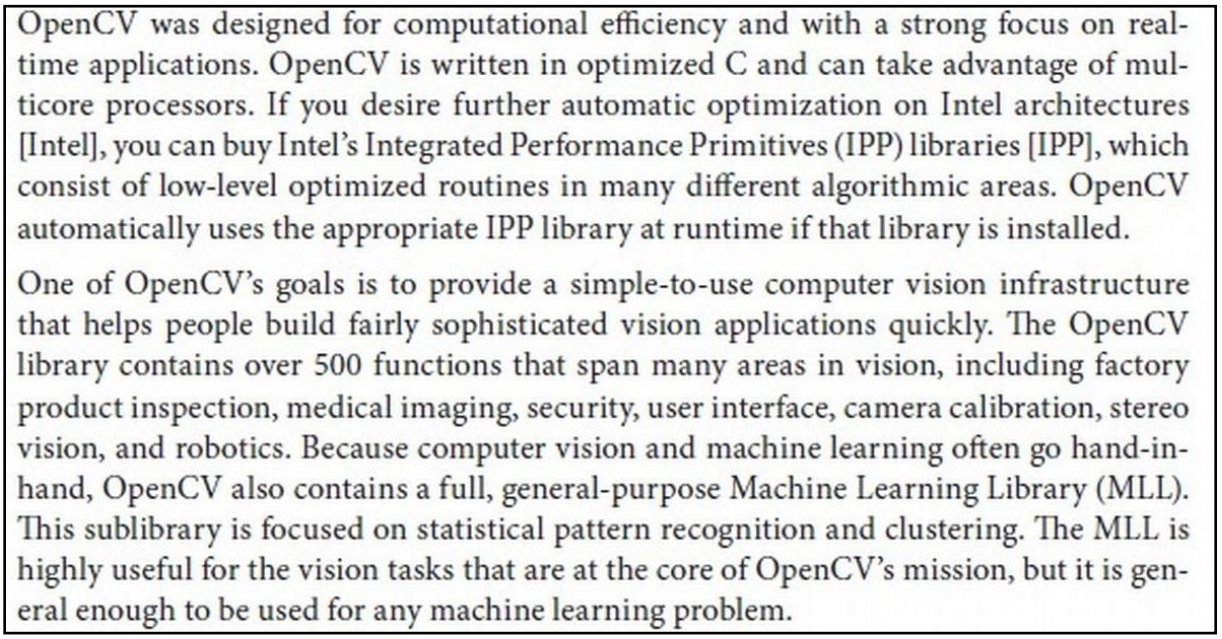
V této části jsem testoval několik vybraných obrazů různých velikostí a kvalit (obr. 35 – 39).


Pozůstalému zůstala jen jedna ponožka, ale nic nenamítal, protože o tom zatím nevěděl.

Při prvním smutečním proslovu se Hihlík přesunul do další řady. Tady ho zaujala černá s temně karmínovým podkresem. Vypadala tak uhrančivě!

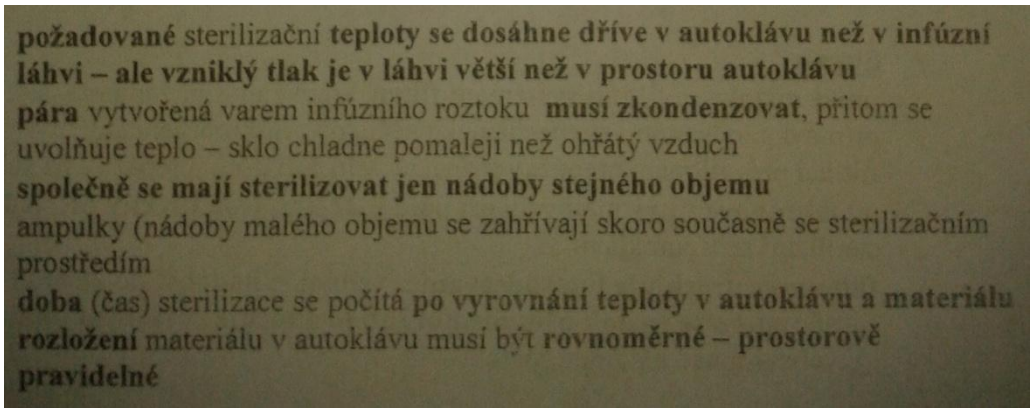
Hihlík ovšem odhadl, že tentokrát to tak snadné nebude. Tahle nevšední ponožka patřila k noze mladého muže, který nevydržel chvíli v klidu. Pořád se díval na hodinky a poklepával střídavě pravou i levou nohou.

Obrázek 34: Testovaný obraz č. 1

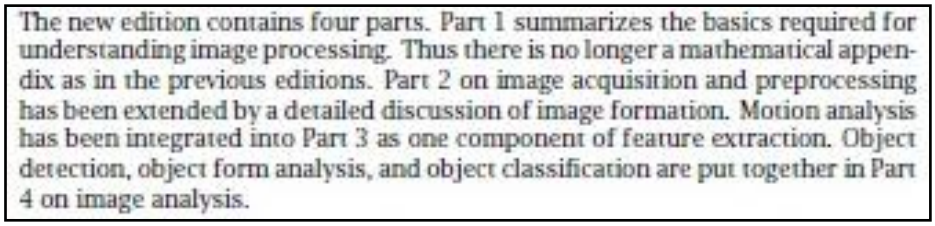

OpenCV was designed for computational efficiency and with a strong focus on real-time applications. OpenCV is written in optimized C and can take advantage of multicore processors. If you desire further automatic optimization on Intel architectures [Intel], you can buy Intel's Integrated Performance Primitives (IPP) libraries [IPP], which consist of low-level optimized routines in many different algorithmic areas. OpenCV automatically uses the appropriate IPP library at runtime if that library is installed.

One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. The OpenCV library contains over 500 functions that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics. Because computer vision and machine learning often go hand-in-hand, OpenCV also contains a full, general-purpose Machine Learning Library (MLL). This sublibrary is focused on statistical pattern recognition and clustering. The MLL is highly useful for the vision tasks that are at the core of OpenCV's mission, but it is general enough to be used for any machine learning problem.

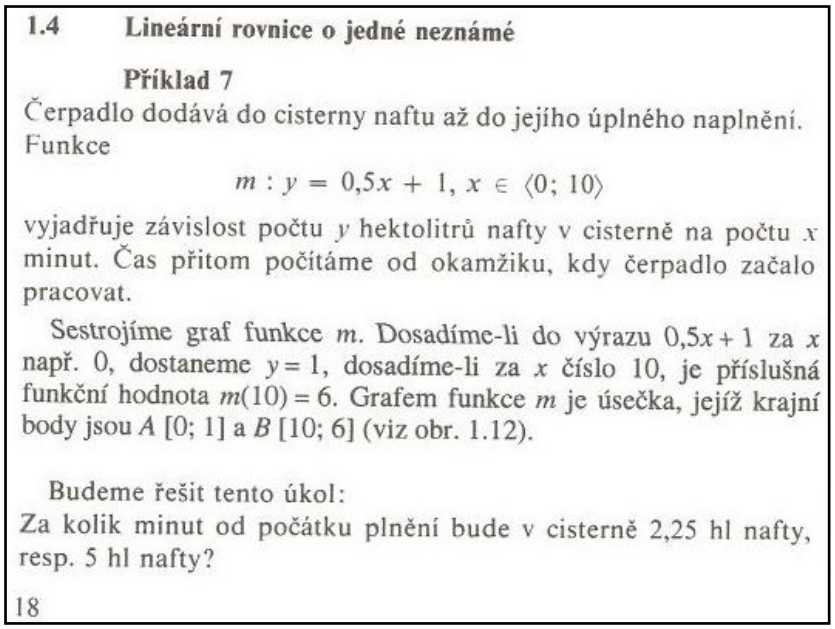
Obrázek 35: Testovaný obraz č. 2



Obrázek 36: Testovaný obraz č. 3



Obrázek 37: Testovaný obraz č. 4



Obrázek 38: Testovaný obraz č. 5

7.2. Počet řádků

Tabulka 1: Úspěšnost počtu rozpoznaných řádků

Číslo testovaného obrázku	Počet řádků	Rozpoznaný počet řádků	Úspěšnost [%]
1.	9	9	100
2.	15	15	100
3.	10	9	92,3
4.	7	7	100
5.	16	11	68,75

Úspěšnost rozpoznání řádků závisela na kvalitě pořízeného snímku. Také velikost řádků hrála roli. Pokud byla mezera mezi řádky velice malá, pak se z histogramu odečítaly špatné hodnoty.

7.3. Počet znaků

Tabulka 2: Úspěšnost počtu rozpoznaných znaků

Číslo testovaného obrázku	Počet znaků	Počet rozpoznaných znaků	Úspěšnost [%]
1.	370	349	94,32
2.	1249	1225	98,07
3.	481	432	89,81
4.	491	443	90,22
5.	516	404	78,29

Zde bylo na závadu při překrývání písmen. Tedy kombinace písmen typu „Vy“ nebo „Ty“ apod., tyto znaky pak vytvářeli jeden slitý znak.

7.4. Testované obrázky – program 2

Testované obrazy programu dva jsou uvedeny výše (obr. 25, 27, 29 31).

7.5. Počet řádků

Tabulka 3: Úspěšnost počtu rozpoznaných řádků

Číslo testovaného obrázku	Počet řádků	Rozpoznaný počet řádků	Úspěšnost [%]
1.	6	6	100
2.	13	11	84,61
3.	5	5	100
4.	5	4	80

7.6. Počet znaků

Tabulka 4: Úspěšnost počtu rozpoznaných znaků

Číslo testovaného obrázku	Počet znaků	Počet rozpoznaných znaků	Úspěšnost [%]
1.	251	250	98,59
2.	111	136	-
3.	246	266	-
4.	116	114	98,27

Z tabulky je patrné, že v některých obrázcích bylo nalezeno více (či méně) znaků, než doopravdy obsahovaly. To je způsobeno tím, že některá písmena se při úpravě slila či rozpadla. Tento problém se také promítl do úspěšnosti rozpoznávání textu.

7.7. Rozpoznaný text

7.7.1. Vstupní obraz 1

Původní text

patches are nucleated by random fluctuations acting on an excitable system; this hypothesis forms the motivation for the study presented here. As we show below. a spatially-extended system can still be excitable and excursions around an excited state can lead to the formation of

Rozpoznaný text

patches e nucleated by nom uctuations actin on an bihhexcitae system tis ypothesis orms the motivation or the study presented here As we show beiow a spatiaiiytsceani exended sytem ca stil be excitbie and excursions adroun an excited state can iead to the oratlon of

7.7.2. Vstupní obraz 2

Původní text

First Visit Frog Legs Lucida Console Doving Fluttercotes Ap_xrst_2 Doubt I don't Believe Bitfunk Indoor-cocooning 8+t Jaze Baqti

Rozpoznaný text

Firt vsst Fgro Legs Lucda cobole Deovng Flutteorcotes Ap_xrs_2 Doubt l Beileve Deont Biltfunk lndoor coceoconng 8+t Jaze Baqt

7.7.3. Vstupní obraz 3

Původní text

ZoomText ImageReader is a software and camera solution that makes printed text accessible to people who are visualy impaired, including books, magazines business documents and more. Put the printed item under the included document camera, snap a picture, and a few seconds

Rozpoznaný text

PoomZext Image Reader is a software and camera soilutilon that makes prnted tet accessilbie to peoie wo ae vsuailipy ilmpailred ilnciludilng books magazilnes bhusilness documents and more Put tbhe prilnted iltem under the ilncilude document camera snap a pilcture and a few seconds

7.7.4. Vstupní obraz 4

Původní text

4481 Quebec Lane Brighton, MI 48116 (810) 923-6395 kryan@rockets.utoledo.edu kryan2013@gmail.com utoledoprssa@gmail.com

Rozpoznaný text

481 Quebec Lane righto0n MI 48116 (810) 923 6395 an@rckets utledo0 edu 2@an03 gmail co0m ut0led pssa @o0gmail co0m

U tohoto obrazu udělal chybu i program s knihovnami Tesseract.

7.8. Úspěch rozpoznání textu

Tabulka 5: Úspěšnost rozpoznaného textu

Číslo testovaného obrázku	Počet znaků	Počet úspěšně rozpoznaných znaků	Úspěšnost [%]
1.	251	222	88,44
2.	111	96	86,48
3.	231	209	90,47
4.	116	102	87,93

Z rozpoznaného textu je patrné, že nejčastější chybou bylo slití kontur, špatné srovnání kontur či špatné rozeznání kontury a jejich zdvojení. Původní text „i“ rozeznán jako „il“.

8. Závěr

Cílem této diplomové práce bylo se seznámit se základními pojmy v oblasti počítačového vidění. Technikami pro rozpoznání text a nejčastěji používanými knihovnami pro práci s obrazem. Vytvořit program odečítající počet řádků a znaků. Hlavním úkolem však bylo vytvořit program pro rozeznávání textu s vybranou rozeznávací metodou a porovnat úspěšnost rozpoznávání u testovaných obrazů s různými vlastnostmi.

V programu jedna jsem použil řadu transformací pro úpravu vstupního (originálního) obrazu. Základní úpravou bylo zbavení se nadbytečných informací v obraze, čehož jsem docílil tak, že jsem obraz převedl na šedotónový. Použitím adaptivního prahování, s velikostí testovaného okolí 5x5 pixelů, jsem dále převedl obraz do binární podoby. Také bylo nutné odstranit šum v obraze. Ten jsem odstranil použitím filtrováním s maskou o velikosti 3x3 pixelů, která má dobrý poměr odstranění šumu a rozostření obrázku.

Problematiku rozpoznávání řádků a znaků jsem řešil pomocí odečítání bílých a černých bodů z obrázků a vynášení jejich výsledné sumy do histogramu. Z histogramu jsem následně vyčítal počty řádků a to tak, že jsem jednotlivé řádky rozpoznával pomocí mezer mezi nimi. Tyto řádky jsem dále vysegmentoval a podobně jako v předchozím případě vyčítal počty rozpoznaných znaků. Spočtené řádky a počty rozpoznaných znaků se zobrazují v konzoli. Tento program využíval starší variantu knihovny OpenCV a operoval se strukturami iplImage.

Program jsem testoval na pěti obrázcích různých velikostí a kvalit. Odečítání řádků bylo kromě dvou případů správné. U určování znaků se správnost pohybovala mezi 78 až 98 procenty. Chyby u určování znaků byly způsobené hlavně tím, že se znaky překrývaly (kombinace písmen „Vy“ nebo „Ty“).

Program dva k předzpracování obrazu používá podobné funkce jako u programu jedna, ale pracuje již se strukturami Mat. Velkým problémem prvního programu byla neschopnost rozeznat znaky z natočeného obrazu. Toto je u tohoto programu vyřešeno pomocí algoritmu, který najde vrcholy bounding boxu řádku a správně ho usadí. Po této

úpravě následovala funkce, která nachází kontury v obraze. Dále jsem použil funkci pro aproximaci kontur s menšími nebo stejnými vrcholy, dle nastavené hodnoty. Pro správné porovnávání kontur je nutné, aby kontury byly správně seřazeny a jednoznačně určeny. Toho jsem docílil funkcí pro setřídění kontur a příkazem pro jejich vymazání. K rozeznání textu je použita metoda pro srovnávání kontur a šablon písmen. Pro každou konturu jsem určil vlastní RoI, který se porovnával s šablonou. Výstupem z porovnávací funkce byla maximální hodnota shody. Pomocí této hodnoty se pak ke znakům přiřazovala jednotlivá písmena. Každé písmeno je pak uloženo do výstupního souboru.

Tento program byl testován na čtveřici obrazů. Odečítání řádků se pohybovalo v rozmezí od 80 procenty do 100 procenty. Přesnost rozpoznávání znaků je lepší než u prvního programu, jak je patrné z počtu rozeznaných znaků, kde jsou hodnoty přes 98 procent. Bohužel rozeznávání jednotlivých znaků je značně ovlivněno rozpadem či sléváním jednotlivých kontur. Tento problém jsem u rozpoznávání textu vyřešil potlačením oblastí kontur, které byly menší než stanovený práh. Úspěšnost rozpoznání textu je celkem vysoká, pohybuje se mezi 86 procenty a 90 procenty. Tato statistika byla ovlivněna sléváním textu a zdvojení podobných kontur.

Nevýhodou metody pro srovnávání šablon je nutnost pro každý obraz a font vytvořit samostatnou knihovnu šablon. Výhodou je, jak je patrno z testovaných obrazů, že tato metoda je invariantní ke změnám jasu.

V diplomové práci jsem vytvořil tři programy, které pracují s knihovnami pro zpracování obrazu. Výstupem prvního bylo počet rozpoznaných řádků a znaků. Ve druhém programu byl přidán algoritmus pro korekci natočení obrazu, nalezení kontur a rozpoznání textu. Výstupem poté byl počet rozpoznaných řádků, znaků a rozpoznaný text uložený v souboru. Posledním programem byla jednoduchá aplikace pro názornou ukázku moderního OCR nástroje Tesseract.

9. Použitá literatura

[1] HORÁK, Karel, KALOVÁ, Ilona, PETYOVSKÝ, Petr, RICHTER, Miloslav. *Počítačové vidění*. Brno: VUT, 2008.

[2] HLAVÁČ, Václav, SEDLÁČEK, Miloš. *Zpracování signálů a obrazů.* Praha: ČVUT, 2005. 255 s. ISBN 80-01-03110-1.

[3] HOLLEY, Rose. *How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs*. D-Lib Magazine: Retrieved 5 January 2011.

[4] MAI, Luong Chi. *Template Matching Algorithm.* Hanoi, Vietnam: Computer Vision, Imaging, Introduction to Computer Vision & Image Processing. Department of Pattern Recognition & Knowledge Engineering IIT.

[5] SVITAK, Joseph John, Jr.. *Genetic Algorithms for Optical Character Recognition*. The City University of New York: 2008. 378 s.

[6] OpenCV [online]. Dostupné z www: http://opencv.org/

[7] BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV*. [s.l.] : O'Reilly, 2008.

[8] Rubrika C/C++ [online]. Dostupné z www: http://www.builder.cz/rubriky/c/c--

[9] SMITH, Ray. *An Overview of the Tesseract OCR Engine.* Google Inc.

[10] Prata, Stephen. *Mistrovství v C++*. 2007. 1120 s.

[11] LEWIS, J.P., *Fast Normalized Cross-Correlation.* Vision Interface: 1995. 7 s.

[12] TSAI, D.M; LIN, C.T.. *Effects of normalized cross correlation on defect detection*. Department of Industrial Engineering and Management. 20 s.

[13] QING, Chen. *Evaluation of OCR Algorithms for Images with Different Spatial Resolutions and Noises*. Ottawa-Carleton Institute for Electrical Engineering. Ottawa. 2003. 122 s.

[14] DI RUBERTO, Cecilia; MORGERA, Andrea. *A new iterative approach for dominant points extraction in planar curves.* University of Cagliari. Cagliari. 12 s.

[15] BRIECHLE, Kai; HANEBECK D. Uwe. *Template Matching using Fast Normalized Cross Correlation*. Institute of Automatic Control Engineering. Germany. 8s .

10. Seznam zkratek

CV	Computer Vision
CC	Cross Correlation
CCOEF	Correlation Coeficinet
CCOEF NORMED	Normalized Correlation Coeficient
CCORR	Cross Correlation
CCORR NORMED	Normalized Cross Correlation
FastCC	Fast Cross Correlation
iplImage	Intel Image Processing Library
NCC	Normalized Cross Correlation
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision
PC	Personal Computer
RoI	Region of Interest
SQDIFF	Square Difference
SQDIFF NORMED:	Normalized Square Difference

11. Seznam příloh

PŘÍLOHA 1 – ZDROJOVÝ KÓD PRO PROGRAM 1

PŘÍLOHA 2 – ZDROJOVÝ KÓD PRO PROGRAM 2

PŘÍLOHA 3 – ZDROJOVÝ KÓD PRO PROGRAM 3

PŘÍLOHA 4 – VSTUPNÍ OBRAZY A ŠABLONY

PŘÍLOHA 5 – OBSAH CD

PŘÍLOHA 6 – CD