VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ ZPRACOVÁNÍ RUČNĚ OPRAVENÝCH TESTŮ

AUTOMATIC PROCESSING OF TESTS CORRECTED BY HAND

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUCIE PELANTOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Pelantová Lucie**
Obor: Informační technologie
Téma: **Automatické zpracování ručně opravených testů**
Automatic Processing of Tests Corrected by Hand
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte postupy segmentaci obrazu, detekce čar a OCR metody.
2. Navrhněte metodu, která v naskenovaném dokumentu provede detekci mřížky a předdefinovaných polí, vysegmentuje z těchto polí ručně psaný text a ten automaticky rozpozná.
3. Vyberte vhodné existující nástroje a navrženou metodu implementujte.
4. Navrhněte a naprogramujte experimentální aplikaci, která automaticky zpracuje mnoho testů naskenovaných do pdf souboru.
5. Proveďte experimenty na stabilitu a přesnost metody i celého systému.
6. Vytvořte plakát a krátké video prezentující klíčové výsledky vašeho řešení.

Literatura:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- Gary R. Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 10: 0-596-51613-4, September 2008.
- Dále dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2, 3 a částečně bod 4.

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: **Beran Vítězslav, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této bakalářské práce je hromadné zpracování ručně opravených testů. Podstatou řešení tohoto problému je detekce tabulky. Struktura tabulky je nalezena nejprvne na vzorové straně a následně je vyhledávána ve zpracovávaných formulářích. Odpovídající si nalezené části tabulky jsou využity k výpočtu homografie. Po aplikování transformační matice jsou vysegmentována data a předána k rozpoznání ručně psaného textu. Práce obsahuje vyhodnocení přesnosti jednotlivých částí systému a možnosti využití v praxi. Výsledná aplikace má umožnit efektivní efektivní segmentaci dat ze strukturovaných dokumentů

Abstract

The aim of this bachelor's thesis is mass processing of exams corrected by hand. The essence of proposed solution is table detection. Structure of table is first detected in sample page, then this pattern is searched in processed page. Corresponding parts of table are used to compute homography. After application of transformation matrix are data segmented and send for handwritten text recognition. The thesis contains evaluation of accuracy for individual system components and possibility of use in practice. Resulting application is supposed to effectively segment data from structured documents.

Klíčová slova

zpracování obrazu, detekce hran, segmentace, renderování PDF, Houghova transofrmace, Homografie, OCR, detekce tabulky

Keywords

Image processing, edge detection, segmentation, PDF rendering, Hough transformation, homography, OCR, table detection

Citace

PELANTOVÁ, Lucie. *Automatické zpracování ručně opravených testů.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Automatické zpracování ručně opravených testů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Lucie Pelantová
14. května 2018

Poděkování

Ráda bych poděkovala Ing. Vítězslavu Beranovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích v průběhu vytváření této bakalářské práce.

Obsah

Kapitola 1

Úvod

Zadávání výsledků testů do školního informačního systému je v současné době nedílnou součástí zkušebního procesu téměř na všech stupních vzdělávacího systému. V současnosti je trendem co nejvíce informací zpracovávat a uchovávat v elektronické podobě, což usnadňuje přístup k datům z jakéhokoliv místa a v libovolný čas.

Jakmile jsou data v elektronické podobě, je kladen důraz na rychlou a jednoduchou manipulaci s nimi. Zároveň je výhodné umožňovat jejich hromadné zpracování. Kritickou sekcí v rychlosti zpracování těchto dat ovšem zůstává jejich zadávání do systému. Obzvlášť pokud se jedná o stovky údajů a jejich vložení do systému spočívá čistě na bedrech člověka.

Vezmeme-li si například bakalářské studium na Fakultě informačních technologií Vysokého učení technického v Brně, kde je v ročníku kolem 600 studentů, je jasné, že hned po samotném opravování testů je druhým nejvíce časově konzumujícím procesem právě vkládání dat to systému. Současně v této fázi zpracovávání testů může vlivem velkého množství údajů a monotónnosti práce docházet k občasným chybám. Řešení takto vzniklých problémů mrhá časem jak hodnotitele, tak hodnoceného.

Použití cílové aplikace ovšem není omezeno pouze na zpracování výsledků testů. Tento postup může být použit i pro hromadné zpracování libovolných dokumentů strukturovaných do tabulky. Toto řešení si klade za cíl při rozpoznání tabulky sledovat pouze tištěné přímky náležící tabulce a nebrat v potaz čtecím zařízením způsobené rušivé vlivy ani objekty přidané při vyplňování formuláře.

Technická zpráva je rozdělena do 3 stěžejních kapitol. První z nich se zabývá teoretickými znalostmi, které shrnují zdroje nastudované při tvorbě této bakalářské práce a které jsou potřebné pro dostatečné pochopení prezentované problematiky zpracování obrazu. V další kapitole je popsán návrh řešení aplikace, který obsahuje jak samotné zpracování obrazu, tak i část popisující návrh metody pro hromadné zpracování testů. V poslední kapitole je popsána samotná implementace výsledné aplikace a její testování.

Kapitola 2

Teorie

Před samotným návrhem řešení daného problému bylo nejdříve nutné nastudovat související teorii. Tato kapitola shrnuje znalosti získané v rámci studování řešeného problému a potřebné k lepšímu porozumění navrženého řešení.

V první části je popsána reprezentace dat ve formátu PDF a extrakce obrazových dat z takovýchto dokumentů na data obrazová. Dále vybrané metody ze zpracování obrazu a následně se tato kapitola bude věnovat způsobům počítačového rozpoznávání textu.

2.1 Formát PDF

Dříve, než se vůbec můžeme začít zabývat obrazem a jeho zpracováním, je nezbytné převést data do formátu, který to umožňuje. Vzhledem k tomu, že očekávanými vstupními daty jsou soubory typu PDF, musí se provést odpovídající převod do nějakého z formátů obrazu – např. png nebo jpg. Z tohoto důvodu je zde shrnut popis formátu PDF a extrakce obrazových dat z dokumentů v tomto formátu.

PDF je formát, který reprezentuje dokument takovým způsobem, aby nijak nezáleželo na aplikaci, zařízení nebo operačním systému, na kterém je vytvářen nebo zobrazován. Tento formát je založen na jazyce PostScript. Soubory v tomto formátu se skládají z kolekce objektů, které společně popisují vzhled jedné nebo hned několika stránek. Zároveň s objekty představujícími obsah dokumentu můžeme v souboru najít také informace o jeho struktuře.

Jednotlivé stránky mohou obsahovat kombinaci textu, grafických elementů a obrázků. Výsledný vzhled stránky potom popisuje tzv. *content stream*, který obsahuje sekvenci objektů, které mají na stránce být zobrazeny. Výsledný vzhled stránky je jednotný ve všech případech, protože veškeré údaje o rozložení objektů na stránce byly plně specifikovány aplikací, která vytvářela content stream.

Kromě statických objektů může dokument obsahovat i interaktivní elementy. Dokument tak může být obohacen o poznámky k obsahu, hypertextové odkazy i různé přílohy.

Druhy objektů

Jak bylo již výše zmíněno, hlavní obsah dokumentu představuje skupina objektů, ovšem ne všechny objekty se nacházejí na stejné úrovni.

Téměř jakýkoliv objekt v PDF může být použit jako *nepřímý objekt*. Takové objekty obdrží unikátní identifikátor, pomocí kterého mohou být odkazované z jiných objektů. Identifikátor se skládá ze dvou částí. První z nich je kladné celé číslo objektu, které se většinou

přiděluje sekvenčně v rámci jednoho PDF. Druhou část tvoří nezáporné celé generační číslo. V nově vytvořeném souboru mají všechna generační čísla nepřímých objektů hodnotu 0, s jinými hodnotami se můžeme setkat pouze v případě, že byl soubor později změněn.

V PDF formátu existuje 8 základních typů objektů [1]:

- Booleovské objekty
- Číselné objekty
- Řetězce
- Jména
- Pole
- Slovníky
- Proudy
- Nulové objekty

Booleovské objekty nabývají pouze dvou hodnot - *true* a *false*. Tyto objekty se často vyskytují právě jako nepřímé objekty, protože mohou reprezentovat hodnoty prvků v polích a slovnících. Také se s nimi můžeme setkat v PostScriptových výpočetních funkcích, kde představují výsledek booleovských a relačních operací, nebo v tělech *if* podmínek, kde reprezentují operandy.

Číselné objekty mohou být dvou typů - celá čísla a čísla reálná. Celá čísla mohou nabývat hodnot z omezeného intervalu se středem v 0. Pokud nějaký celočíselný objekt tento interval překročí, je automaticky převeden na reálný číselný objekt, který má rozsah hodnot větší. Reálná čísla jsou většinou reprezentována v pevné řádové čárce a jejich rozsah a přesnost závisí na zařízení, na kterém je PDF zpracováváno.

Řetězce jsou uloženy jako série bytů, kdy každý byte nabývá hodnoty od 0 do 255 a představuje jeden znak. I přesto, že nabývá celočíselných hodnot, je v paměti uložen úspornějším způsobem. Maximální povolená délka řetězce se v různých verzích může lišit a závisí na implementaci.

Řetězcům reprezentujícím specifický údaj může být přiřazen jednotný formát, který definuje výslednou podobu řetězce. Takový formát potom definuje konvenci interpretování řetězce a je popsán už při definici řetězcového objektu. Příkladem, kdy přiřazení formátu může být užitečné, je například řetězec představující časovou značku nebo datum.

Jmenné objektyjsou nedělitelné symboly, které jsou jednoznačně definované posloupností znaků. Jakékoliv dva takové objekty definované stejnou posloupností znaků musejí být naprosto shodné.

Definice jmenného objektu je uvozena lomítkem, které není považováno za součást jména. Lomítko je bezprostředně následováno samotným jménem, které nesmí obsahovat bílé znaky.

Stejně jako u řetězců i povolená délka jména závisí na konkrétní implementaci. S výsledným jménem je posléze pracováno jako s nedělitelnou jednotkou a téměř nikdy s ním není zacházeno jako s běžným textem.

Pole reprezentují sekvenci objektů. PDF formát přímo podporuje pouze pole o jedné dimenzi. Nicméně na rozdíl od běžné konvence zde je pole považováno za heterogenní typ. Jednotlivé prvky obsažené v poli mohou tedy být různých datových typů včetně vnořených polí.

I u pole maximální povolená délka závisí na konkrétní implementaci.

Slovníky jsou chápány jako soubor dvojic objektů. První objekt představuje klíč a druhý příslušnou hodnotu. Klíč musí povinně být typu jmenného objektu. Hodnota však může být jakýkoliv z typů objektů včetně dalšího slovníku. V případě, že se v jednom slovníku vyskytne více záznamů se stejným klíčem, hodnota tohoto klíče nebude definovaná.

Slovníky jsou často používány k vytváření komplexnějších objektů jako je font nebo stránka, protože umožňují spojit dohromady jejich atributy. Každý záznam ve slovníku potom představuje jméno jednoho atributu a jeho příslušnou hodnotu.

Proudy jsou reprezentovány sekvencí bytů. Jednotlivé byty proudu je možné načítat postupně, na rozdíl od řetězců, u kterých jsou všechny byty přečteny najednou. Dalším rozdílem je to, že délka proudu je neomezená. Z tohoto důvodu jsou objekty, u kterých se očekává velký obsah dat, ukládány právě jako proudy.

Každý proud se skládá ze slovníku, který mimo jiné obsahuje údaj o počtu bytů tvořících obsah proudu. Za ním následuje samotný obsah proudu, který je vložen do klíčových slov *stream* a *endstream*. Všechny proudy musejí být nepřímé objekty. Na druhou stranu slovník, který proud uvozuje, musí být objektem přímým.

V novějších verzích PDF dokumentů je možné obsah proudu ukládat do externího souboru. Takovýto soubor je pak specifikován ve slovníku proudu. Veškeré byty, které jsou zapsány mezi klíčová slova za slovníkem, jsou potom ignorovány.

Nulové objekty mají typ a hodnotu, kterých nemůže nabývat žádný jiný objekt. Odkaz na nepřímý objekt, který neexistuje, se chová jako nulový objekt. Uvedením nulového objektu na pozici hodnoty záznamu ve slovníku se dosáhne stejného výsledku, jako kdyby slovník záznam daného jména vůbec neobsahoval.

Struktura dokumentu

Všechny dokumenty v tomto formátu mají stejnou stěžejní strukturu. Ta vypadá následovně:

- Hlavička - Specifikuje verzi PDF
- Tělo - Obsahuje jednotlivé objekty definující obsah dokumentu
- Tabulka odkazů - Uchovává informace o nepřímých objektech v souboru
- Závěrečná sekce - Informace o struktuře dokumentu, např. odkaz na tabulku odkazů

Takto struktura vypadá při vytvoření dokumentu, nicméně pozdější úpravy mohou přidat určité elementy na konec souboru.

Jednotlivé datové jednotky v PDF dokumentu se ukládají do řádků, které jsou ukončeny znakem konce řádku. Dokumenty s binárními daty však mohou obsahovat velice dlouhé řádky, což snižuje kompatibilitu s aplikacemi zpracovávajícími PDF. Z tohoto důvodu všechny řádky, které nejsou součástí datového proudu, jsou až na speciální výjimky omezeny na délku maximálně 255 znaků [1].

Hlavička je v každém PDF dokumentu umístěna na prvním řádku souboru. Na tomto místě je specifikována verze PDF, které je dokument přizpůsoben. Všechny verze PDF odpovídají konvencím vyšších verzí. Nicméně v opačném případě to neplatí, protože jsou do nových verzí přidávány nové funkce, které v dřívějších verzích ještě nebyly zavedeny.

Při dodržení určitých podmínek může aplikace podporující pouze starší verze zpracovat i dokument ve verzích novějších. To je možné z toho důvodu, že nové funkce jsou do verzí přidávány s ohledem na to, aby bylo bezpečné jejich ignorování cílovou aplikací, která jim nerozumí.

Tělo souboru se skládá z posloupnosti nepřímých objektů, které představují obsah souboru. Objekty základních typů představují jednotlivé části dokumentu jako jsou například fonty a stránky. Od novějších verzí PDF formátu může tělo dokumentu obsahovat i proudy objektů. Každý takový proud potom představuje sekvenci nepřímých objektů.

Tabulka odkazů obsahuje údaje o umístění jednotlivých nepřímých objektů v souboru. Díky těmto informacím je umožněno přistupování k objektům nesekvenčním způsobem, což snižuje vyhledávací čas. Pro každý nepřímý objekt obsahuje tabulka jeden řádek, který definuje přesné umístění objektu v těle souboru.

Tato část dokumentu jako jediná má přesně specifikovaný formát, který dovoluje náhodný přístup k jednotlivým záznamům. Tabulka odkazů se může skládat z několika sekcí. Při vytvoření dokumentu je celá tabulka obsažena v jedné sekci, ale s každou úpravou dokumentu vzniká sekce nová.

Volné záznamy v tabulce odkazů tvoří spojový seznam, kde každý volný záznam obsahuje číslo objektu následujícího volného záznamu. První záznam v tabulce je vždy volný a představuje začátek spojovaného seznamu. Poslední z volných záznamů vždy odkazuje na první prvek seznamu.

Kromě prvního záznamu tabulky mají všechny objekty v tabulce generační číslo s hodnotou 0. Pokud dojde k vymazání objektu, je jeho záznam označen za volný a přidán do spojovaného seznamu. Generační číslo takového záznamu je potom inkrementováno o 1, což značí, že má být tento záznam použit v případě, že je vytvořen objekt se stejným číslem objektu.

Tabulka odkazů musí povinně obsahovat záznam pro každé číslo objektu od 0 až po maximální číslo objektu, které se v tabulce vyskytuje. Toto pravidlo musí být dodrženo i v případě, že některá čísla objektů se nikde v souboru nevyskytují.

Závěrečná sekce obsahuje umístění tabulky odkazů a dalších speciálních objektů potřebných k rychlému zpracování dokumentu. Proto by aplikace manipulující s PDF dokumenty měly tyto soubory vždy prohlížet od konce.

Jedním ze speciálních objektů, který je obsažen v závěrečné sekci, je závěrečný slovník. Mezi záznamy v tomto slovníku patří například celkový počet položek v tabulce odkazů, odkaz na slovník obsahující informace o dokumentu atd.

Extrakce obrazových dat z formátu PDF

Aplikace, která je výsledkem této bakalářské práce může dostat na vstup jak vzorovou stránku, která je dokumentem skládajícím se z výše zmíněných objektů viz část 2.1, tak skeny, které jsou v podstatě obrázky vložené do dokumentu. Z toho důvodu nástroj použitý

pro získání obrazových dat musí umožnit jak renderování, tak extrakci obrázků z dokumentu. Obě činnosti umožňuje nástroj **MuPDF**.

MuPDF

MuPDF je open-source framework vyvíjený v jazyce C, který se zabývá zpracováním a renderováním dokumentů ve formátech PDF, XPS a EPUB. I přes to, že hlavní použití MuPDF je právě renderování stránek, umožňuje nad dokumenty ve výše zmíněných formátech provádět i jiné operace, mezi které patří například hledání a procházení obsahu a hyperodkazů. V novějších verzích je navíc přidán nástroj pro vytváření a editaci PDF souborů.

MuPDF se zaměřuje na rychlé a kvalitní renderování bez vzniku aliasů v obraze. Při zpracování stránky pracuje nad jejím obsahem a anotacemi, které předává ke zpracování. Jádrem celého zpracování jsou touto knihovnou definovaná zařízení. Proces potom probíhá tak, že stránka je rozdělena na jednotlivé grafické objekty, které jsou postupně předávány příslušnému zařízení. Těch je několik druhů a každý druh zprostředkovává jinou operaci. Knihovna navíc umožňuje vytvoření nových zařízení pro konkrétní účely.

Pro účel této práce je nejdůležitější kreslící zařízení, které zajišťuje právě renderování dokumentu. Každý objekt, který je mu předán ke zpracování, vykreslí do finální mapy pixelů. Tato mapa pixelů musí být vytvořena už před začátkem vykreslování a její rozměry odpovídají rozlišení výsledného obrazu [20].

2.2 Předzpracování obrazu

S rozšiřováním informačních technologií vzniká i potřeba umožnit počítačům využívat a zpracovávat více reprezentací dat. Mezi tyto reprezentace patří i obrazová data. I přesto, že pro člověka je porozumění obrazovým datům intuitivní, je velice složité stejné úrovně porozumění dosáhnout prostřednictvím počítače.

Zpracování obrazu je proces přeměny dat ze stálého obrazu nebo videa na rozhodnutí nebo na jinou reprezentaci těchto dat. Obrazová data se počítači předávají jako matice čísel, případně jako matice vektorů čísel jedná-li se o barevný obraz. Veškeré zpracování se provádí transformacemi nad touto maticí. Někdy se k samotným obrazovým datům mohou přidávat i další informace například o původu obrazu, času pořízení a mnoho dalších.

Samostatné zpracování se dá rozdělit do několika základních na sebe navazujících částí. První z nich je předzpracování obrazu, což je vlastně příprava před samotným zpracováním. To se snaží eliminovat co nejvíce rušivých informací a tím zpřesnit výsledky a omezit chybné vyhodnocování irelevantních dat.

Metody zde uvedené se provádějí na nejnižší úrovni abstrakce dat. Formát dat vstupujících i vystupujících z těchto transformací je shodný.

Tyto metody jsou užívané pro opravu nežádoucích vlastností obrazu nebo na vylepšení vlastností výhodných pro pozdější analýzu. Mezi opravy chyb obrazu patří například následující [18]:

- Opravy chyb senzoru - Někdy mohou být nežádoucí vlastnosti obrazu způsobeny přímo zařízením pořizující obraz. Příkladem těchto chyb může být mrtvý pixel nebo geometrické zkreslení čočkou.

- Opravy osvětlení - Ke špatným výsledkům zpracování obrazu může dojít, pokud podstatnou část obrazu zakrývá stín, nebo pokud jsou části obrazu nerovnoměrně osvětleny.
- Opravy šumu - Velice častým negativním faktorem u zpracování obrazu je právě šum. Ten může být způsoben například kombinací nepříznivých podmínek při snímání s nekvalitním snímacím zařízením, ale i mnoha dalšími vlivy. Metody patřící do této skupiny fungují na principu odstranění malých artefaktů v obraze, tímto však mohou nepříznivě ovlivnit ostatní vlastnosti obrazu.
- Geometrické opravy - Negativně ovlivnit analýzu obrazu může špatné natočení snímku, nebo jeho pořízení z nevhodného úhlu.
- Opravy barev - Metody opravující chyby vzniklé například barevným odrazem. V tomto výčtu jsou však uvedené pouze pro úplnost a pro další text nejsou podstatné.

Je několik pohledů, na základě kterých můžeme metody předzpracování klasifikovat. Jedním z nich je dělení podle okolí pixelu, které je využité k vypočítání jeho nové hodnoty jasu[17]:

- Transformace jasu pixelu - Nová hodnota je vypočítána na základě původní hodnoty pixelu. Tato metoda se může dále dělit na základě toho, zda bere v potaz i pozici pixelu v obraze, nebo zda o nové hodnotě rozhoduje pouze na základě hodnoty původní.
- Geometrická transformace
- Lokální předzpracování - Nová hodnota je určena na základě lokálního okolí počítaného pixelu.
- Obnovování obrazu - K vypočítání nové hodnoty vyžaduje znalost celého obrazu.

Vyhlazení

Metody vyhlazování jsou jednoduché a často používané operace. Mezi mnoho důvodů jejich používání patří zejména redukování šumu v obraze nebo zahlazení artefaktů zapříčiněných pořizovacím zařízením. V některých případech je vyhlazení také použito ke snížení rozlišení obrazu. Úroveň vyhlazení a snížení rozlišení závisí na velikosti okolí, které je použito pro vypočítání nové hodnoty pixelu.

Dále jsou uvedeny některé metody používané pro vyhlazování [2].

Prosté vyhlazení je nejjednodušší způsob vyhlazování. Jedná se o průměr všech okolních hodnot pixelů. Je náchylný na šum v obraze, kdy velký rozdíl okolních hodnot ovlivňuje vypočítaný průměr.

Medianový filtr Nahrazuje současnou hodnotu počítaného pixelu střední hodnotou jeho okolí. Na rozdíl od prostého vyhlazení není tak náchylný na šum v obraze. Vybíráním pouze střední hodnoty zanedbává maxima v okolí, tím pádem není ovlivněn rozdílnými hodnotami způsobenými šumem.

Gaussovské vyhlazování získává novou hodnotu počítaného pixelu aplikováním konvoluce Gaussovou maskou na jeho okolí. Tato maska aproximuje výsledný pixel následujícím vzorcem[17].

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

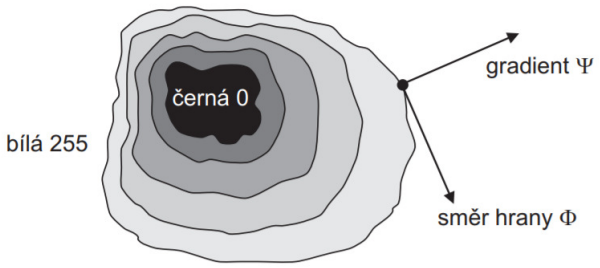
Kde x a y jsou souřadnice v obraze a σ je středněkvadratická odchylka. Výše uvedený vztah pro Gaussův filtr má pouze jeden parametr a tím je právě středněkvadratická odchylka σ, ta je úměrná velikosti okolí, které je použito k vyhodnocení počítaného bodu obrazu. V důsledku tohoto parametru mají body s větší vzdáleností od středu filtru menší váhu a pixely, které jsou dále než 3σ mají váhu zcela zanedbatelnou.

Z uvedených metod dosahuje nejlepších výsledků, a proto je nejpoužívanější metodou. Nicméně co se týká rychlosti, patří mezi pomalejší metody.

Bilaterální filtrování používá stejně jako Gaussovské vyhlazování k vypočítání nové hodnoty vážený průměr. Hodnota váhy v průměru má dvě části, jedna je odvozena od vzdálenosti od počítaného pixelu stejně jako u Gaussova filtru, druhá je odvozena od rozdílu intenzity od hodnoty počítaného pixelu - čím menší tento rozdíl je, tím vyšší je hodnota váhy. Tato metoda patří mezi metody vyhlazování zachovávající hrany.

2.3 Detekce hran

Detektory hran patří mezi metody lokálního předzpracování, které sledují změny ve funkcích intenzity v obraze. Detekované hrany jsou pak taková místa, na kterých se hodnoty funkce strmě mění. Změny ve funkcích intenzit se dají reprezentovat gradientem, který ukazuje ve směru nejstrmějšího růstu obrazové funkce. Samotná hrana je spojená s konkrétním pixelem a je reprezentována vektorem definujícím její směr a velikost. Velikost hrany se rovná velikosti gradientu a směr hrany je kolmý ke směru gradientu [17].



Obrázek 2.1: Gradient, jeho směr a směr hrany

Velikost gradientu $|gradg(x,y)|$ a směr gradientu ψ získáme z následujících vzorců:

$$|gradg(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{2.2}$$

$$\psi = \arg(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}) \tag{2.3}$$

V samotném zpracování obrazu se pro hledání hran používají hranové operátory. Ty podle principu, který využívají k detekci hran, můžeme rozdělit do 3 hlavních skupin [12]

- Aproximace derivace pomocí diferencí
- Průchody 2. derivace obrazové funkce nulou
- Lokální aproximace obrazové funkce jednoduchým parametrickým modelem

Operátory aproximující derivace pomocí diferencí

Výsledky těchto operátorů získáme konvolucí obrazu s maskami operátorů. V případě operátorů, které jsou invariantní ke směru hrany (např. Laplaceův operátor), probíhá konvoluce jen s jednou maskou. U ostatních operátorů, které jsou ovlivněny směrem hran, se provede konvoluce několika maskami. V takovém případě masky odpovídají různým orientacím hrany a na závěr se vybere takový výsledek, který nejlépe lokálně aproximuje obrazovou funkci. Příkladem druhé skupiny je například Sobelův nebo Prewittův operátor.

Laplaceův operátor

Je velice oblíbený operátor, který aproximuje druhou derivaci. Tím z obrazu extrahuje pouze velikost gradientu bez ohledu na jeho směr. Je tedy odolný vůči natočení obrazu. Jeho diskrétní definice vypadá následovně [17]:

$$\nabla^2 g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{2.4}$$

V praxi je tento vzorec aproximován konvolucí, ke které se nejčastěji používá 3x3. Výsledná maska se liší podle toho, zda hledáme hrany ve 4-okolí nebo 8-okolí obrazu.

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.5}$$

Někdy je bodům okolo pixelu přiřazeno rozdílné ohodnocení. V takovém případě je potom ztracena odolnost vůči natočení a maska konvoluce vypadá například takto:

$$h = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \qquad h = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -8 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{2.6}$$

Průchody 2. derivace obrazové funkce nulou

Hlavní otázkou u těchto metod je, jak robustně vypočítat 2. derivaci obrazové funkce. Jedním z možných řešení této otázky je před odhadem 2. derivace použít na obraz některý z filtrů na vyhlazení obrazu pro redukci šumu. Na takový filtr ovšem vznikají dva protichůdné požadavky:

- Pro omezení možných frekvencí, které mohou způsobit průchod nulou 2. derivace, by měl být filtr hladký a přibližně odpovídající pásmové propusti.
- Pro přesnou lokalizaci hran by měl filtr reagovat pouze v blízkém okolí hrany.

Jak už bylo řečeno výše, tyto dva požadavky si vzájemně odporují. Může být však nalezeno optimální řešení s využitím Gaussova rozložení 2.2.

Po použití filtru je potřeba získat 2. derivaci vyhlazené obrazové funkce. S tím nám může pomoci Laplaceův operátor 2.3, který operuje nad 2. derivací. Takovému použití Laplaceova operátoru na obraz vyhlazený Gaussovým operátorem se říká *Laplacián Gaussiána* a označuje se tímto vztahem 2.7, kde $\star$ značí konvoluci Gaussova filtru G s obrazovou funkcí f.

$$\nabla^2 \left[G\left(x, y, \sigma\right) \star f\left(x, y\right) \right] \tag{2.7}$$

Cannyho hranový detektor

přináší nový přístup k detekci hran, který je optimální pro hrany poničené bílým šumem. Tento přístup se řídí třemi kritérii [3].

1. Kritérium detekce vyjadřuje to, že důležité hrany by neměly být přehlédnuty a zároveň by ve výsledku neměly být falešné hrany.
2. Kritérium lokalizace vyznačuje, že vzdálenost mezi původní a detekovanou hranou by měla být minimální
3. Kritérium jednoznačné odezvy minimalizuje několik odpovědí do jediné hrany. Toto kritérium je částečně pokryto kritériem detekce, protože pokud zde jsou dvě detekce jedné hrany, tak jedna z nich by měla být považována za falešnou.

Mezi základní kroky Cannyho hranového detektoru patří [12]:

- Vyhlazení obrazu - odstranění šumu ve vstupním obrazu. Většinou je tento krok realizován konvolucí s Gaussovým fitrem. Tímto krokem je zajištěno první kritérium, tedy eliminace detekce falešných hran.
- Aproximace první derivace - výpočet gradientu většinou pomocí konvoluce se Sobelovým operátorem. V tomto kroku je vypočten směr hrany. Výběrem vhodného operátoru je zajištěno kritérium lokalizace.
- Nalezení lokálních maxim - z obrazu vzniklého po předchozím kroku jsou potlačeny hodnoty, které nejsou lokální maxima. Tímto krokem je nalezená hrana ztenčena a je tím dosaženo kritéria jednoznačné odezvy.
- Odstranění nevýznamných hran - je omezen počet hodnot, kterých mohou nabývat body v obraze. Většinou je v tomto kroku provedena binarizace obrazu pomocí prahování.

2.4 Detekce čar

Houghova transformace

Houghova transformace je metoda, která umožňuje detekci předem známých objektů.

Tento problém může být také vyřešen pomocí posouvání vzoru objektu po obraze a následným zkoumáním, zda se vzor s obrazem dostatečně shoduje. Nicméně tento postup je

velice náchylný na šum a dává špatné výsledky v případě rozdílné velikosti nebo natočení objektu oproti vzoru. Tyto problémy řeší právě Houghova transformace.

Původní Houghova transformace byla navržena k detekci čar a křivek v rovině a může být použita k detekci objektů, pokud je známo analytické vyjádření jejich hranic. Nicméně ne vždy je možné vyjádřit obrys objektu analytickou rovnicí. Později byla proto metoda Houghovy transformace upravena tak, že není potřeba předem znát analytické vyjádření [17].

Standartní Houghova transformace

Principem této metody je transformace z Kartézského souřadnicového systému do parametrického. Pokud tedy v obraze hledáme přímku podle jejího směrnicového vyjádření $y = kx + q$ a známe dva body $A\,[x_1, y_1]$ a $B\,[x_2, y_2]$, kterými přímka prochází, převedeme oba body do systému, kde jedna osa reprezentuje k a druhá q - bod A bude tedy reprezentován přímkou $q = -kx_1 + y_1$ a bod B přímkou $q = -kx_2 + y_2$. Potom přímka, která spojuje oba tyto body, bude charakterizována parametry k a q bodu, ve kterém se výše uvedené přímky protínají.

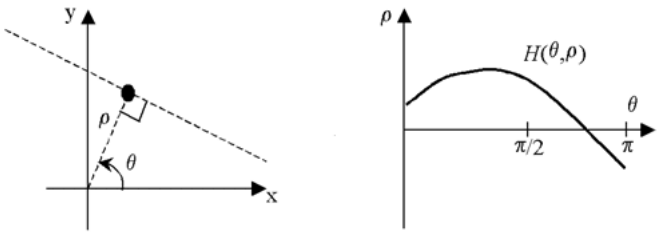
Každá přímka v obraze tedy může být reprezentována jedním bodem v parametrickém souřadnicovém systému.

Hlavní myšlenkou detekce čar je potom nalezení všech možných bodů v obraze, které mohou reprezentovat přímku a nalezení jejich příslušných bodů v parametrickém prostoru. Výsledkem transformace potom budou takové přímky, jejichž body v parametrickém prostoru byly často nalezeny jako průsečíky přímek v tomto prostoru, který se někdy označuje jako *akumulační prostor*, protože se sčítají průchody jednotlivých bodů v tomto prostoru.

V praxi se však ukázalo, že používat pro parametrický prostor směrnicové vyjádření není vhodné. To z toho důvodu, že směrnice přímky může nabývat hodnot od- $-\infty$ do ∞. Proto se v praxi jednotlivé přímky v obraze ukládají pomocí souřadnic (ρ, θ) do systému polárního [17]. Křivka v polárním systému, která reprezentuje bod v původním obraze, je vyjádřena rovnicí:

$$\rho = x \cos\theta + y \sin\theta \tag{2.8}$$

Vyjádření bodu v polárním systému lze vidět na obrázku 2.2.



Obrázek 2.2: Převod bodu do polárního systému: vlevo - bod v kartouzském systému, vpravo - reprezentace bodu v systému polárním [6]

Progresivní probabilistická Houghova transformace

Progresivní probabilistická Houghova transformace se od standartní varianty liší tím, že navíc zjišťuje i hranice nalezených čar. Nedetekuje tedy přímky ale ohraničené úsečky.

Probabilistická se tato metoda označuje proto, že při její aplikaci se upustilo od prozkoumání všech bodů, které by mohly představovat čáru, ale počítá se pouze zlomek z nich. Hlavní myšlenkou, která vedla k zavedení tohoto přístupu, je předpoklad, že pokud vrchol v součtu procházejících přímek bodem v akumulačním prostoru je vysoký, bude oproti ostatním bodům dostatečně vyšší i v případě, že se omezí počet vstupů. Vlivem této obměny je dosaženo částečného snížení výpočetního času.

Algoritmus výpočtu progresivní probabilistické Houghovy transformace je navržen tak, aby minimalizoval nároky na výpočet. V každém kroku algoritmu je vybrán náhodný bod, podle něhož jsou přičteny hodnoty příslušným bodům v akumulačním prostoru. Po každém tomto přičtení následuje otázka, zda sečtené hodnoty v bodech polárního systému mohou ještě být způsobeny šumem, tedy zda nějaký bod naakumuloval m průchodů z N vypočtených bodů, a tím překročil práh s, který se očekává od náhodného šumu. Tento test je proveden porovnáním s prahem s, který se však s počtem počítaných kroků N mění.

Pokud je tímto testem nalezena čára, potom jsou z akumulačního prostoru odečteny body, které tvořily nalezenou čáru. Zároveň jsou ze skupiny bodů, pro které měly být vypočteny hodnoty v akumulačním prostoru, odstraněny i body, které byly součástí detekované čáry. Následně pokračuje výpočet pro další náhodný bod [11].

Výhody Výhody tohoto algoritmu zahrnují schopnost nalezení objektu okamžitě po přesáhnutí součtu bodů v akumulačním prostoru. Tím je zaručena schopnost algoritmu vracet validní výsledky i v případě, kdy je v jakémkoli čase přerušen.

Další významnou předností je, že algoritmus nepotřebuje žádnou konečnou podmínku. Výpočet je sám přerušen v okamžiku, kdy pro každý bod vstupního obrazu, který je potencionálně součástí přímky, buď byly vypočteny příslušné hodnoty a přičteny v akumulačním prostoru, nebo byl bod přiřazen již detekované přímce a v důsledku toho byl vyřazen z výpočtu.

Nevýhody Nevýhodou v porovnání se standardní Houghovou transformací je možnost detekování falešných čar. Pokud porovnáme standardní Houghovu transformaci s její progresivní probabilistickou variantou, je to vlastně jediná možná nevýhoda oproti původní metodě. Ta se navíc projeví jen v případech, kdy je přiřazení bodu k přímce nejednoznačné, tedy v případě, kdy jsou body v okolí několika různých čar.

Přehlédnutí nějaké křivky nepředstavuje problém, protože po dokončení algoritmů jak standartní tak progresivní probabilistické Houghovy transformace jsou výsledky obou metod shodné. Pokud tedy byla křivka detekována standartní variantou, bude ve výsledku detekována i progresivně probabilistickou úpravou metody.

Matematická morfologie

Jednou z dalších možností detekce čar je použití operací matematické morfologie.

Matematická morfologie využívá nelineárních nástrojů, které operují nad sadami bodů, jejich okolí a tvary. Metody spadající do matematické morfologie mohou být použity v různých částech zpracování obrazu, kde mohou mít různé využití. Příklady použití mohou být následující [17]:

- Předzpracování obrazu - filtrování šumu, zjednodušení obrazu
- Úprava struktury objektu - skeletonizace, zúžení, rozšíření

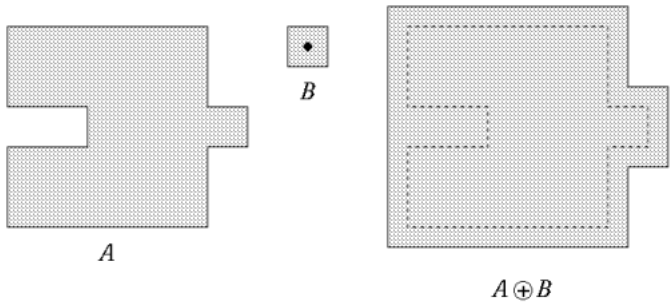
- Segmentace objektu od pozadí
- Detekce objektů

Samotná morfologická transformace je relací mezi dvěma sadami bodů, kde jedna sada reprezentuje obraz a druhá potom strukturní element. Aplikace morfologické transformace potom probíhá posouváním strukturního elementu po celém obraze. Každý strukturní element obsahuje *reprezentativní bod*, při aplikaci morfologické transformace je bodu obrazu, který pozicí odpovídá reprezentativnímu bodu, přiřazena hodnota v závislosti na použité morfologické operaci.

Následně jsou přiblíženy základní morfologické transformace.

Dilatace

Výsledek dilatace je získán kombinací dvou setů bodů využitím vektorového součtu. Ukázku použité této operace lze vidět na obrázku 2.3.

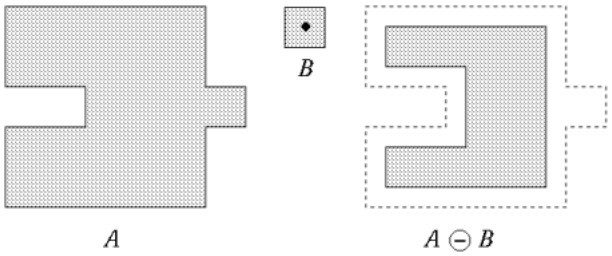


Obrázek 2.3: Aplikování dilatace [14]

Použitím je dosaženo zaplnění malých děr a spojení blízkých objektů. Tím je zvětšena velikost objektů v obraze. Pokud je třeba velikost objektů zachovat je dilatace použita v kombinaci s erozí.

Eroze

Podobně jako dilatace i eroze kombinuje dva sety bodů, k ustanovení výsledného setu však používá operaci vektorového rozdílu. Výsledek aplikování eroze lze vidět na obrázku 2.4.



Obrázek 2.4: Aplikování eroze [14]

Při použití jsou z obrazu odstraněny takové body, jejichž okolí se přesně neshoduje se strukturním elementem v pozici, kdy se překrývá reprezentativní bod s tímto bodem.

V důsledku tedy dochází k zmenšení obrazu, kdy jsou okrajové části vymazány a úzké mezery jsou rozšířeny.

2.5 Segmentace

Účelem segmentace je rozdělit obraz na části, které co nejvíce odpovídají dílčím prvkům reálného světa zachyceného v obraze, nebo takové, které sdílí určitou společnou vlastnost. Výsledné segmentované části se budou lišit podle toho, jakou metodu segmentace použijeme. Právě podle výsledku segmentace můžeme používané metody rozdělit do dvou skupin[17]:

- Úplná segmentace
- Částečná segmentace

Úplná segmentace většinou vyžaduje spolupráci s vyšší úrovní zpracování, která alespoň částečně rozumí zpracovávanému obrazu a objektům na něm. V některých případech však můžeme dosáhnout úplné segmentace pouze s pomocí nižší úrovně zpracování. Takové případy tvoří hlavně obrazy se souvislým pozadím a jasně ohraničenými objekty. Výsledkem úplné segmentace potom jsou části, které přesně odpovídají prvkům zachycených na zpracovávaném obraze.

Částečná segmentace neústí v samotné objekty, ale pouze v části, které jsou si podobné s ohledem na použitou vlastnost. Příkladem takových vlastností může být jas, barva, textura a další. K této segmentaci není potřeba žádná znalost o vlastním obsahu obrazu. Částečně segmentovaný obraz neposkytuje dostatečné výsledky pro zpracování, to i z toho důvodu, že se rozpoznané části mohou překrývat. Proto je následně nutné předat segmentovaný obraz k dalšímu zpracování například s využitím zpracování vyšší úrovně.

Další dělení, které můžeme na segmentační metody uplatnit, závisí na vlastnostech obrazu použitých k oddělení jednotlivých částí. Podle tohoto kritéria se metody dělí na tři části[15]:

- Metody využívající obecné znalosti o zpracovávaném obrazu nebo jeho části, které bývají reprezentovány histogramem vlastností obrazu
- Metody segmentující na základě hran v obraze
- Metody segmentující na základě oblastí v obraze

Druhá a třetí skupina takto rozdělených metod řeší podobné problémy. Platí totiž, že každá hrana ohraničuje určitou oblast a zároveň každá oblast je nějak ohraničena. Nicméně výsledky těchto dvou skupin nejsou vždy ekvivalentní. Samotná segmentace je totiž úzce spjata s metodou, která je pro ni použita. Z toho důvodu jsou v následující části shrnuty často používané segmentační metody.

Prahování

Tato metoda se používá v případech, kdy je potřeba se rozhodnout mezi dvěma hodnotami. Nejčastěji se používá k binarizaci obrazu, kdy jsou hodnoty stupně šedé převáděny pouze na bílé nebo černé. K tomuto účelu je určen práh, který dělí množinu možných hodnot na

dvě části a nová hodnota je přidělena na základě toho, zda původní hodnota byla nižší nebo vyšší než stanovený práh.

Mezi další účely použití prahování patří omezení rozsahu intervalu nabývaných hodnot shora respektive zdola, kdy hodnotám vyšším respektive nižším než práh je přiřazena hodnota definovaná prahem. Posledním zmíněným použitím prahu je přiřazení předem dané hodnoty, nejčastěji se používá hodnota 0, hodnotám nacházejícím se nad respektive pod daným prahem.

Existují však případy, kdy je určení pevně daného prahu pro všechny hodnoty příliš hrubé a může zapříčinit zkreslení obrazu a následné chybné výsledky zpracování. Tuto možnost si můžeme uvést na názorném příkladu: uvažujme následující situaci, kdy potřebujeme provést binarizaci obrazu, avšak část zachycené scény je osvětlená, nebo část scény naopak pokrývá stín. V takovém případě může chybně zvolená hodnota prahu způsobit velké jednolité plochy a tím pádem zkreslení obrazu. Určení pevného prahu v takových případech může být velice obtížné a někdy žádná hodnota prahu nepřinese optimální výsledky.

Adaptivní prahování Ve výše zmíněných případech je vhodné použít metodu adaptivního prahování. Tato metoda na rozdíl od běžného prahování, kde je práh odvozen od obrazu jako celku, odvozuje práh na základě právě zpracovávané pozice. Jednou z možností, jak takový práh odvozovat, je rozdělit zpracovávaný obraz na několik částí a v každé části určit vlastní práh nezávisle na ostatních dílech obrazu.

Segmentace na základě hran

Segmentace na základě hran je jedním z nejstarších přístupů k segmentaci vůbec a stále se často používá. Obraz se u metod spadajících do této skupiny dělí podle informací o hranách v obraze. Informace o hranách získávají pomocí některé z metod detekce hran. Ty v obraze hledají výrazné skoky například v hodnotách jasu pixelu nebo barvě. Po získání hran pokračuje segmentace jejich zřetězením, aby lépe kopírovaly hranice v obraze. Cílem je, aby se jednotlivé hrany spojily do řetězců tak, aby ve výsledku zůstaly jen řetězce, které odpovídají objektům v obraze nebo alespoň jejich částem. Tím je dosaženo částečné segmentace.

Segmentace na základě oblastí

Na rozdíl od segmentace na základě hran, která hledá ohraničení výsledných částí, se segmentace na základě oblastí zaměřuje přímo na nalezení dílčích částí obrazu. Jak už bylo výše řečeno, tento způsob je segmentaci na základě hran dosti podobný, protože nachází oblasti, které jsou hranami ohraničené. Ne vždy oba přístupy vrací stejné výsledky a v mnoha případech se ukázalo výhodné oba přístupy navzájem kombinovat.

Při segmentaci na základě oblastí se může postupovat třemi různými postupy [17].

Metoda šíření oblasti pracuje na principu rozdělení obrazu na malé části. Tyto části jsou následně procházeny a v případě, že je u sousedních oblastí zaznamenána dostatečná podobnost, dojde ke spojení těchto částí. Opakovaným spojováním až do okamžiku, kdy už žádné dvě oblasti nesplňují podmínky ke sloučení, se dosáhne získání výsledných částí segmentace.

Metoda dělení oblasti je přímým opakem metody šíření oblasti. Na začátku metoda pracuje nad celistvým obrazem, který na základě rozdílných vlastností dělí na drobnější

části. V případě, kdy už žádná část nemůže být dále rozdělena, se dosáhlo finálního výsledku segmentace.

Metoda dělení a spojování oblastí je kombinací předchozích dvou metod. Metoda prochází jednotlivé regiony obrazu, a pokud je nějaký region nehomogenní, je rozdělen na 4 části. Naopak jsou-li si dvě sousední oblasti dostatečně podobné, jsou sloučeny do jednoho regionu.

2.6 Homografie

Homografie je projektivní namapování jedné roviny do druhé. Pokud máme bod a v rovině Q a bod a' v rovině Q' takové, že:

$$a = [X\,Y\,1]^T \tag{2.9}$$

$$a' = [X'\,Y'\,1]^T \tag{2.10}$$

potom homografie reprezentuje perspektivní transformaci mezi rovinami Q a Q'. Tuto perspektivní transformaci můžeme popsat vztahem:

$$a' = Ha$$

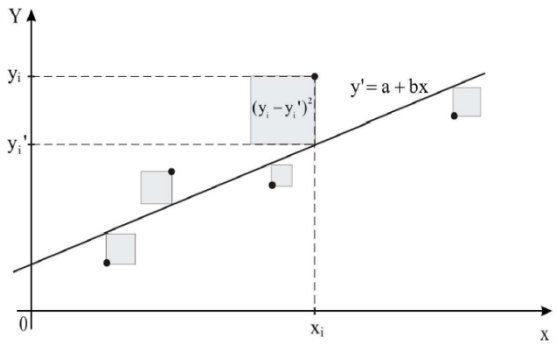
kde H představuje transformační matici 3x3, která reprezentuje převod jednoho bodu v rovině, na jeho obraz v rovině druhé.

Jakmile je nalezena transformační matice, jakýkoliv bod jedné roviny může být převeden na odpovídající bod v rovině druhé [16].

Pro získání homografie mezi dvěma obrazy, je nejprve nutné nalézt v obou obrazech klíčové body. Ty jsou si pak vzájemně přiřazeny jednou z následujících metod.

Metoda nejmenších čtverců

Při použití pouze metody nejmenších čtverců se k výpočtu výsledné transformační matice použijí všechny body z obou sad klíčových bodů. Pro tyto body je hledáno takové řešení, kdy průměr obsahů čtverců, jejichž hrana je definována vzdáleností přiřazených bodů viz obrázek 2.5, je co nejmenší.



Obrázek 2.5: Výpočet nejmenších čtverců

RANSAC

RANSAC (Random Sample Consensus) je iterativní algoritmus pro odhad parametrů matematického modelu ze skupiny vzorových dat. Tento algoritmus předpokládá, že vzorová vstupní data se skládají ze dvou skupin. Jednou skupinou jsou inliers, které náleží hledanému modelu a mohou být použity pro jeho odhad. Druhou skupinou jsou tzv. outliers, které nenáleží modelu, a proto by jeho výpočet neměly ovlivňovat [7].

Jedna iterace tohoto algoritmu je popsána Algoritmem 1.

Algorithm 1 RANSAC iterace

1: Ze vstupních dat je náhodně vybrána skupina
2: Jsou určeny parametry modelu na základě této skupiny
3: Jsou určeny vzorky s chybou nepřesahující hodnotu tolerance
4: Takto určené vzorky jsou považovány za *inliers*
5: Pokud je dostatek *inliers* jsou použity k výpočtu finálního modelu

Po provedení n iterací uvedeného algoritmu je jako výsledek vrácen ten model, který má nejmenší průměrnou chybu ze všech generovaných modelů. Modely jsou generovány metodou nejmenších čtverců.

Tento algoritmus je robustní, protože vykazuje dobré výsledky i s velkým poměrem outliers ve skupině vzorových dat. Nicméně je nutné vhodně stanovit práh tolerance odchylky.

LMeDS

LMeDs (Least Median of Squares) je stejně jako RANSAC metoda, která počítá s tím, že skupina vzorových dat obsahuje outliers. Na rozdíl od RANSAC metody nemusí být určen práh tolerance chyby, ale je vyžadováno minimálně 50% inliers ve skupině vzorových dat [8].

Tato metoda je iterativní, v každém kroku náhodně vybere skupinu prvků ze vzorových dat o rozsahu p a pomocí této skupiny vypočítá model. Nejlepší výsledný model vybere na základě nejmenšího mediánu čtverců vzniklých odchylkou od původního modelu.

Díky tomu, že na rozdíl od metody nejmenších čtverců neuvažuje průměr velikosti čtverců, ale jejich medián, zanedbává extrémně odchýlené body (outliers).

2.7 OCR

OCR je zkratka pro anglický výraz *Optical character recognition*. Účelem OCR je pomocí zpracování převést ručně psaný text uložený jako součást obrazu na text zakódovaný tak, aby mu stroj mohl porozumět.

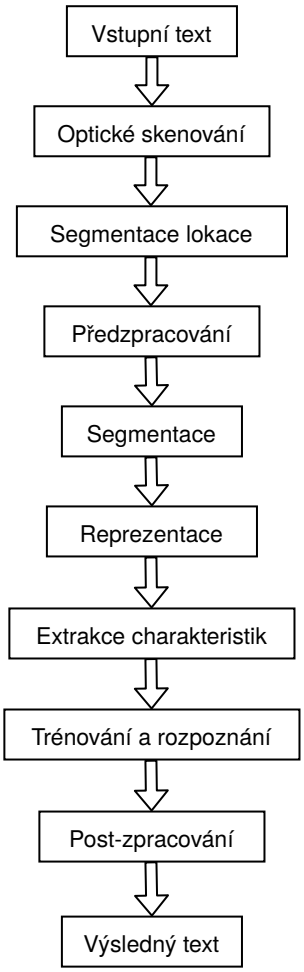
Historie OCR sahá až do roku 1929, kdy byl na toto téma přidělen první patent. V té době se jednalo o mechanický systém, který pomocí šablon zkoumal shodu znaku. Pokud světlo neproniklo mechanismem, potom byla nalezena dokonalá shoda a znak byl rozpoznán. Od té doby byl v tomto oboru učiněn obrovský pokrok, kdy už není žádoucí jen pouhé rozpoznání znaku, ale především jeho digitalizace ve formátu, kterému porozumí počítač. Jedná se tedy po klávesnici a myši o další prostředek pro komunikaci člověka s počítačem.

Ačkoliv cíl byl mírně pozměněn, idea rozpoznávání znaku podle porovnávání se šablonami zůstala. V současné době se tedy rozpoznávání textu ubírá dvěma hlavními proudy. Jedním je už zmíněný *template matching* a druhým proudem je *structure analysis* [13].

V průběhu vývoje těchto dvou proudů se stále více oba proudy ubíraly stejným směrem. To až do takové míry, že se zjistilo, že přístupy obou principů jsou si stále více podobné.

V dnešní době už je hranice mezi přístupem *template matching* a *structure analysis* velice úzká a očekává se, že v krátké době se obě metody postupně sloučí úplně.

OCR funguje na principu *pattern recogniton*. Jedná se o postup, kdy je rozpoznávaný objekt přiřazen k jedné z již existujících tříd. Celkový postup zpracování podle [4] je zobrazen na obrázku 2.6.



Obrázek 2.6: Postup rozpoznání textu

Předzpracování a segmentace

Tato část závisí z velké části na tom, z jakého zdroje pochází vstupní obraz.

Aplikace detekce textu na skenovaný obraz nevyžaduje přílišné úpravy. Pozadí skenovaného dokumentu je většinou téměř bílé bez značného šumu. Zároveň strana není nijak výrazně nakloněná. Takové zpracování potom vyžaduje pouze jednoduché předzpracování.

Základní metody zlepšení vlastností obrazu a segmentace částí byly uvedeny v předchozí kapitole, proto se zde jimi už dále nebudeme zabývat.

Ve chvíli, kdy už jsou vlastnosti obrazu co možná nejlépe upraveny a je známa pozice textu, který má být rozpoznán, následuje část, ve které musí dojít k rozdělení bloku textu na části, které je počítač schopen rozpoznat. Proto po předzpracování a lokalizaci textu následují tato jednotlivá dělení [19]:

1. Rozdělení na linky
2. Rozdělení na slova
3. Rozdělení na jednotlivé znaky

Dělení na linky

V této části je blok textu rozdělen na jednotlivé řádky. V praxi je toho běžně dosahováno poměrně intuitivním způsobem, kdy algoritmus postupně zkoumá jednotlivé pixely v řádcích obrazu. Jako oddělovač jednotlivých řádků textu se potom považují taková místa, kde počet pixelů s hodnotou rozdílnou od pozadí textu se rovná 0, nebo dosahuje velice nízké hodnoty.

Nicméně existuje několik situací, které tento postup mohou značně ztížit.

Jako příklad takové situace může být dotýkání řádků. K tomu dochází, když je volný prostor mezi řádky malý a znaky z jednoho řádku se dotýkají znaků v řádku následujícím. Tento problém je patrný při dělení tištěného textu a v případech, kdy je rozpoznáván ručně psaný text, je ještě umocněn, protože při psaní textu není stanovena žádná pevná mezera mezi řádky a v celém textu se může měnit.

V takovém případě musí metoda dělení řádků správně detekovat spodní části písmen jako je "j", "q", "p", nebo například "g" a přiřadit je odpovídajícímu řádku. Vzniklý problém tohoto typu může být způsoben i háčky a čárkami, které se nacházejí v prostoru mezi jednotlivými řádky[19].

Dělení na slova a znaky

Podobně jako při dělení na řádky je problém segmentace textu na jednotlivá slova a znaky často realizován pomocí součtu pixelů nepatřící pozadí obrazu, tentokrát však ve vertikálním směru. Nicméně dvojí dělení přináší další problém - jak poznat, zda se jedná o mezeru pouze mezi jednotlivými znaky ve stejném slově, nebo zda už mezera odděluje 2 jednotlivá slova. Při ručně psaném textu a v některých případech i v textu tištěném (zarovnání textu do bloku) dochází k případům, kdy mezera mezi znaky je širší než mezera oddělující slova, respektive mezera mezi slovy je užší než mezera mezi dvěma znaky v jednom slově.

Stejně jako u řádků může dojít ke slití znaků dohromady.

V případě ručně psaného textu, který je psán písmem vázaným, je možné dělit text maximálně na úroveň jednotlivých slov. Není totiž možné spolehlivě rozpoznat, kde v jednom slově znak začíná a kde končí. Systémy, které takový text rozpoznávají potom detekují jednotlivá naučená slova. Rozpoznávaných slov je v takových systémech pouze omezené množství, proto je zatím velice obtížné detekovat nestrukturovaný vázaně psaný text[19].

Extrakce charakteristik

Rozpoznávaný znak je vždy k příslušné třídě přiřazován na základě nějakého jeho popisu. Proces získávání takového popisu se nazývá **extrakce charakteristik**. Tento proces se snaží co nejdetailněji popsat určený objekt, aby bylo snazší přiřazení k příslušné třídě.

Nejjednodušším přístupem k popisu znaku je pomocí jeho obrazových dat ve formě hodnot jasu jeho příslušných pixelů.

Další možnosti už nejsou tak přímočaré, ty se snaží co nejlépe zachytit charakteristiky znaku užitečné pro rozpoznání a zanedbat ty méně důležité. Techniky extrakce charakteristik znaku jsou například následující [4]:

- Distribuce bodů
- Transformace
- Analýza struktury

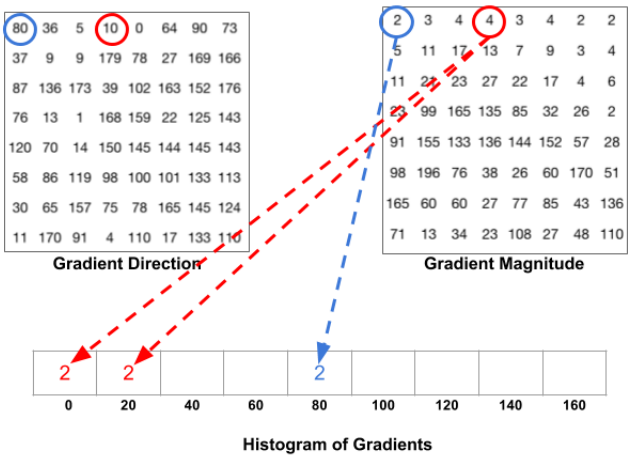
Na základě takto nalezených charakteristik následně probíhá klasifikace jednotlivých znaků do tříd. Klasifikace probíhá hledáním podobnosti v jednotlivých vstupních vzorcích a následně určí, které ze tříd znak nejvíce odpovídá.

Histogram orientovaných gradientů

Jedním z možných popisů charakteristik obrazu je histogram orientovaných gradientů.

Dále je uveden výpočet histogramu orientovaných gradientů. [10]

1. Nejprve je třeba nalézt jednotlivé gradienty v obraze.
2. U nalezených gradientů je následně vypočtena velikost a směr.
3. Poté je obraz rozdělen na menší části, pro které je vytvořen histogram, který udává součet velikostí gradientů v jednotlivých směrech viz obrázek 2.7.



Obrázek 2.7: Přičítání velikostí gradientů podle jejich směru [10]

Výsledný popis obrazu je tedy set histogramů, kdy vždy jeden histogram popisuje jednu část obrazu.

Momenty

Histogram orientovaných gradientů popisuje obraz součtem gradientů v jednotlivých směrech, proto se jeho výsledná podoba pro stejný obraz s jiným natočením může značně lišit.

Tento problém lze řešit pomocí momentů obrazu, které umožňují zjistit natočení objektu v obraze. Poté lze obraz narovnat, což zvyšuje možnost, že příslušný objekt v obraze bude mít podobný histogram orientovaných gradientů jako jeho vzor.

Moment řádu $p + q$ pro digitalizovaný obraz můžeme vyjádřit pomocí vzorce

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i,j) \tag{2.11}$$

kde i, j jsou souřadnice pixelů obrazu. Výpočtem momentu nultého řádu m_{00} u binarizovaného obrazu získáme celkovou oblast objektu v obraze. Takto získané hodnoty momentů závisí na posunutí, velikosti, rotaci a transformaci. Existují však různé obměny momentů, které tyto závislosti odstraňují [17].

Jednou z těchto obměn jsou centrální momenty.

Pomocí momentů vypočítaných vzorcem 2.11 můžeme následujícími rovnicemi získat polohu centroidu v obraze.

$$x_c = \frac{m_{10}}{m_{00}} \qquad y_c = \frac{m_{01}}{m_{00}} \tag{2.12}$$

Pokud je při výpočtu momentů pozice každého pixelu vztažena k vypočtenému centroidu, je získán centrální moment, který je invariantní vůči posunutí v obraze. Výpočet centrálního momentu řádu $p + q$ je uveden na vzorci 2.13 [17].

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i,j) \tag{2.13}$$

Pomocí centrálních momentů lze získat i požadované natočení objektu. To je vypočteno pomocí rovnice 2.14 [5].

$$\alpha = \frac{1}{2} \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{2.14}$$

Trénování a rozpoznání

Aby bylo možné správně provést klasifikaci, je potřeba nástroj, který nalezne podobnosti k jednotlivým třídám. Takový nástroj nejprve musí sám být vytrénovaný na trénovací sadě a na základě znalostí získaných z této sady je potom schopen rozpoznat podobnost s jednotlivými třídami. K tomuto účelu se často používá technik strojového učení. Často používaným nástrojem pro klasifikaci do tříd se také stávají neuronové sítě.

Dále jsou popsané některé techniky klasifikace využívající strojové učení.

Support vector machine

Support vector machine je N-třídní klasifikátor, který vstupní data promítá do prostoru s více dimenzemi. Toho je dosaženo tím, že vytváří nové dimenze kombinováním vlastností dat. V nově vzniklém prostoru je jednodušší najít lineární oddělovač mezi jednotlivými třídami, na druhou stranu v původním prostoru dat nemusí být možné najít lineární separátor. Při dostatečném množství dalších dimenzí je nalezení vhodného oddělovače jednotlivých tříd možné prakticky vždy.

Jiné metody klasifikátorů mohou udávat lepší výsledky, ale k jejich spolehlivosti je nutná dostatečně rozsáhlá trénovací sada. Na rozdíl od takových metod klasifikátor Support vector machine dosahuje spolehlivosti i s omezenými trénovacími daty [2].

K-Nearest Neighbors

Stejně jako Support vector machine je K-Nearest Neighbors N-třídní klasifikátor. K-Nearest neighbors pracuje tak, že ukládá trénovací data. Při přidělování nového bodu do příslušné skupiny, je nejprve z trénovacích dat vyhledáno K nejbližších bodů v uložených datech. Následně je nový bod přidělen do té třídy, která obsahuje největší počet z jeho K sousedů.

Přes jeho poměrně snadný přístup k výpočtu vrací velice dobré výsledky. Nicméně je nutné mít v paměti uloženou celou trénovací datovou sadu. Z toho důvodu je při větším objemu trénovacích dat možné, že bude klasifikátor pomalý [2].

Kapitola 3

Navržená metoda pro hromadné zpracování testů

Tato kapitola popisuje postup návrhu zpracování naskenovaných dat a to včetně slepých větví vývoje. Jsou zde popsány úvahy nad řešením a důvody, které vedly k vytvoření finálního řešení.

Jak už bylo v této práci dříve nastíněno, očekávaným vstupem je dokument ve formátu PDF. Konkrétně jsou ke zpracování zkušební sady testů vyžadovány dva soubory PDF. Prvním dokumentem je vzorový nevyplněný formulář, který byl použit pro tisk zadání testů. V nejlepším případě je tento vzor před použitím přímo vygenerován programem pro tvorbu PDF, takže není zatížen zpracováním žádným vstupním zařízením - například skenerem, nebo dokonce pouze zachycen na fotografii. Druhým souborem jsou potom samotné naskenované vyplněné testy. Pro jeden běh zpracování aplikace jsou všechny formuláře určené k extrakci dat obsaženy pouze v jednom vícestránkovém dokumentu PDF. Rozdělení tohoto souboru na jednotlivé strany je provedeno v rámci samotného procesu zpracování.

Výstupem aplikace je textový soubor ve formátu CSV. Nicméně možnost výstupu v jiném formátu, nebo případně zapojení vytvořené aplikace na vstup dalšího systému tvoří prostor k dalšímu vývoji.

Výchozím předpokladem pro zpracování je podmínka, že data ve zpracovávaném formuláři jsou strukturovaná do jedné nebo více tabulek. Pro extrakci dat je jako jedna datová jednotka považována jedna buňka tabulky.

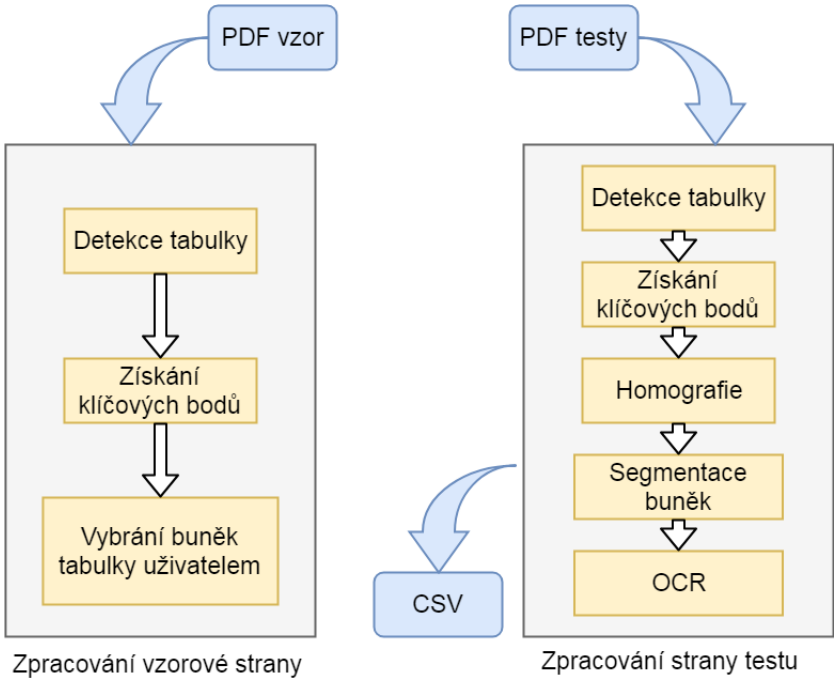
3.1 Postup zpracování

Od celkového zpracování všech vložených testů se předpokládá velká časová náročnost. Proto byla při návrhu snaha co nejrychleji získat vše potřebné od uživatele, aby dále mohla aplikace běžet na pozadí bez jakékoli potřeby obsluhy.

Celkové schéma zpracování je zobrazeno na obrázku 3.1.

Nejvíce časově konzumující bod celého zpracování je převod PDF dokumentu na obraz. Tento proces zabírá i několik vteřin, proto je v první řadě převeden pouze vzor testu, který je potřebný k získání dat od uživatele.

Po dokončení převodu vzorové strany na obraz, jsou jeho obrazová data využita pro detekci tabulky na stránce. Průsečíky této tabulky jsou následně vyznačeny do obrazu, který je zobrazen uživateli, aby mohl ověřit, že tabulka byla správně detekována.



Obrázek 3.1: Schéma postupu zpracování testů

Na vzorové straně jsou uživatelem identifikovány buňky tabulky, které si přeje vysegmentovat. Dále zadá uživatel cestu k druhému vstupnímu dokumentu, který obsahuje samotné testy. Po těchto krocích začíná samotné zpracování.

Protože se od aplikace očekává potencionálně neomezené množství zpracovaných testů, převést všechny testy na obraz a jejich uložení by bylo velice paměťově náročné. Z toho důvodu je v každém okamžiku v paměti pouze právě zpracovávaná strana.

U každé zpracovávané strany se provede vyhledání průsečíků tabulek. Protože by bylo nepraktické muset před vlastním skenováním oddělovat testy od případných papírů s poznámkami studenta, proběhne vyhodnocení , zda nalezené klíčové body odpovídají klíčovým bodům ze vzorové stránky. Pokud je podobnost nalezena, je pomocí takovýchto klíčových bodů nalezena homografie.

Po aplikování homografií nalezené transformační matice získá zpracovávaný formulář takovou podobu, že naklonění a umístění tabulky na stránce přesně odpovídá vzorové straně. Díky tomu mohou být z testu vysegmentovány pouze buňky tabulky ze stejného umístění jako na vzoru formuláře, ve kterém byly uživatelem označené pro zpracování.

Každá takováto buňka je potom předána k rozpoznání textu. Metody a postup použitý v jednotlivých úsecích běhu aplikace jsou popsány v následujících částech kapitoly.

3.2 Zpracování strany

V této části je popsán postup návrhu samotného zpracování obrazu. Jeho cílem je detekovat tabulky nejvyšší úrovně a vypočítat jejich průsečíky čar. Tyto průsečíky jsou posléze použity jako klíčové body při porovnávání se vzorovou stránkou i pro finální segmentaci buněk tabulky.

Před samotným návrhem výsledného řešení byly nejprve experimentálně testovány jednotlivé metody pro detekci čar, detekci tabulky a následné porovnání testu se vzorovým

zadáním. Tyto experimenty vedly k objevení problémů, které by výsledný návrh měl řešit. Z toho důvodu jsou tyto experimenty popsány u jednotlivých částí návrhu a je vysvětlen jejich dopad na výsledné řešení.

V následujícím textu je popsán výsledný postup zpracování obrazu, jehož průběh znázorňuje schéma na obrázku 3.2.

Obrázek 3.2: Postup detekce tabulky na straně

Předzpracování

Vzhledem k tomu, že očekávaným vstupním zařízením je skener, tak se při zpracování nepředpokládá přílišné natočení strany ani nedostatečné osvícení. Z tohoto důvodu v této fázi dochází spíše než k opravám poškození obrazu k upravení dat pro pozdější jednodušší zpracování.

Nejdříve je provedeno mírné vyhlazování použitím Gaussovské matice o rozměrech 3x3.

Barvy v obraze nejsou nijak potřebné ke zpracování, proto je možné je zanedbat a redukovat tím množinu možných hodnot jasu v obraze. Pro samotné zpracování je v podstatě potřeba pouze odlišit pozadí strany od popředí. Z tohoto důvodu bylo v dalším kroku aplikováno adaptivní prahování.

V důsledku skenování a následné rasterizace pdf může dojít k rozpojení slabších původně souvislých čar. Na závěr předzpracování byla proto aplikována dilatace. V našem zájmu je nejvíce žádoucí spojit čáry v horizontálním a vertikálním směru, proto byl pro dilataci

použit strukturovací element ve tvaru kříže:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.1}$$

Detekce tabulky

Definice problémů detekce tabulky

Před samotným vytvořením návrhu byly prozkoumány rozdílné přístupy. Jedním z nich bylo přímé hledání kontur v obraze pomocí algoritmu sledování hranice. V tomto případě však nebyly vždy spolehlivě detekovány buňky tabulky. Navíc byly zachyceny i kontury vzniklé hustým textem ve skenu nebo náčrtky přidané při vyplňování. Pomocí tohoto přístupu byla objevena potřeba odstranit objekty v dokumentu, které nebyly součástí struktury vzorové tabulky.

Jedním z dalších testovaných přístupů, bylo hledání čar pomocí standardní Houghovy transformace. Ta nalezla veškeré linie v obraze. Nicméně také tento přístup byl zamítnut, protože neumožňoval variantu přítomnosti více tabulek na straně. Při více výskytech tabulky na jedné straně totiž nebylo možné nijak rozeznat, které z nalezených čar patří k příslušné tabulce.

V reakci na výše zmíněný problém bylo vyzkoušeno hledání čar pomocí progresivní probabilistické Houghovy transformace, která nevrací jen naklonění a posunutí nalezené linky v obraze, ale především její koncové body. Tento přístup už vykazoval větší potenciál než výše zmíněný, protože bylo jasné, k jaké tabulce příslušný nalezený segment patří. Nicméně v detekovaných čarách byly zahrnuty veškeré čáry v obraze. Nekonzistence potom vznikaly v případech, kdy byly do zadání dokresleny další čáry nebo i celé tabulky. Při tomto experimentu bylo dosaženo závěru, že pro robustnější zpracování je výhodné detekovat pouze tabulky nejvyšší úrovně. Do detekovaných částí by tedy neměly být zahrnuty vnořené tabulky.

Pro vynechání vnořených tabulek z výpočtu byl vyzkoušen algoritmus semínkového vyplňování. Idea při jeho použití byla taková, že při postupném procházení obrazem od nějakého z krajů se každý pixel, který nemá hodnotu pozadí strany, použije jako semínko pro vyplňování. Pokud je poté oblast zasažena vyplňováním větší než minimální hranice očekávaná od tabulky, je taková oblast vynechána z dalšího vyplňování. V důsledku toho veškerý další obsah takové oblasti není vyplněn a tím pádem je vynechán z dalšího zpracování. Zároveň je logicky nemožné při postupném procházení takto nalézt jinou tabulku než tabulku první úrovně. Problémem použití této metody na celý obraz bylo časté pronikání vyplnění i do vnitřních částí tabulky, které byly způsobeny objekty na straně vzniklé při vyplňování nebo opravování dokumentu.

Další pokus se tedy zaměřoval na vynechání ze zpracování právě takových částí obrazu. V případě čar načrtnutých rukou se na rozdíl od tištěných čar patřících vyplňované tabulce dá předpokládat, že většina z nich nebude přímo vertikální nebo horizontální. Proto bylo účelem tohoto experimentu vybrat takovou metodu, která v obraze najde jen vertikální a horizontální linie a až ty budou později předány k výše zmíněnému semínkovému vyplňování. Nicméně ačkoliv jsou dokumenty skenované a nedochází u nich k výraznému natočení strany, tak aplikace musí poskytovat jistou toleranci vůči mírně nakloněným skenům.

Výsledný návrh detekce tabulky

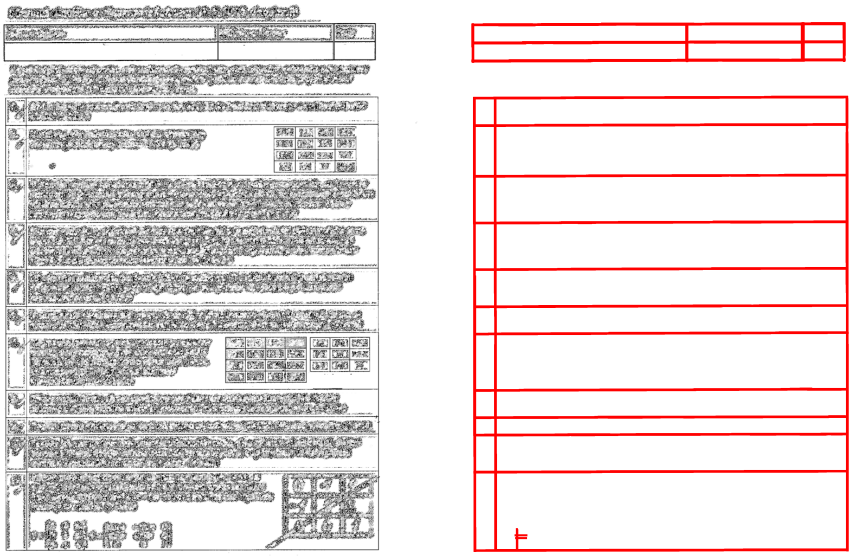
Jak už bylo výše zmíněno, cílová aplikace má rozpoznat tabulku pouze na nejvyšší úrovni. Proto by při zpracování měly být zanedbány čáry patřící vnořeným tabulkám i objekty vytvořené při vyplňování testu a posléze při jeho opravování.

Návrh aplikace předpokládá, že vnořené tabulky s hlavní tabulkou nesdílí žádnou z čar, ani je žádná tištěná okolní čára spolu nespojuje.

Dále návrh předpokládá, že drtivá většina čar přidaných při vyplňování a následujícím opravování není striktně vertikální nebo striktně horizontální. Pro eliminaci takovýchto objektů je navržen následující postup.

Metodou, která umožňovala detekci tabulky i u mírně nakloněné strany, se ukázalo použití morfologických operací. Pro ty musí být nejprve vytvořeny strukturovací elementy, v tomto kroku jsou vytvořeny dva - jeden představující horizontální linii a druhý představující linii vertikální. Následně je každý z těchto dvou elementů použit na jednu kopii obrazu nejdříve pro dilataci, aby se odpovídající struktura zvýraznila. A následně je na takto vzniklých datech použit stejný element pro erozi za účelem eliminovat části obrazu, které mu neodpovídají.

Tímto krokem vzniknou dva obrazy o stejné velikosti jako původní, jeden obsahuje vertikální a druhý horizontální strukturu. Spojením těchto struktur vznikne obraz obsahující linie tabulek, nicméně obsahuje i čáry tabulek vnořených a objekty kopírující řádky v místech s hustým písmem. Tyto části je pro další zpracování také nutné eliminovat. K tomuto účelu je využito semínkové vyplňování. Pokud se postupně prochází od jednoho rohu a na každý nenulový pixel se uplatní semínkové vyplňování, tak první nalezený objekt, který vyplní nezanedbatelnou část obrazu, bude tabulka nejvyšší úrovně. A pokud žádná horizontální čára přímo nespojuje tuto tabulku s vnořenými tabulkami, potom po přeskočení mimo takto nalezenou oblast budou vnořené tabulky úspěšně zanedbány.



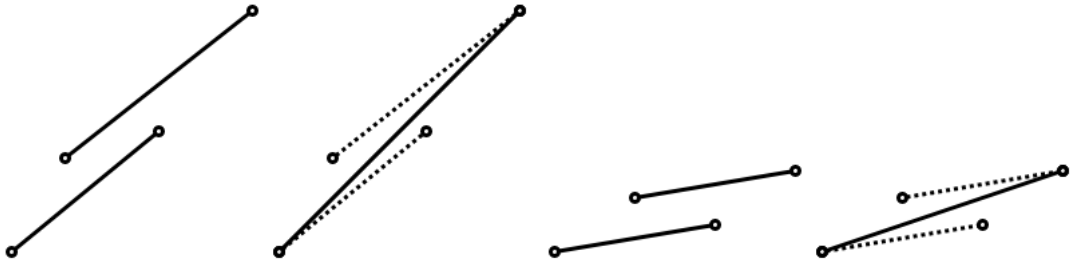
Obrázek 3.3: Zanedbání vnořených tabulek: vlevo - zpracovávaná strana, vpravo - výsledná struktura po semínkovém vyplňování

Pro vyhledání čar byla použita metoda progresivní probabilistické Houghovy transformace pro detekci čar. Tato metoda detekuje jednotlivé segmenty čar. Standartní Houghova detekce čar nemohla být použita, protože neuvažuje konce čar a jakoukoli nalezenou úsečku považuje za přímku táhnoucí se od jednoho okraje až ke druhému.

Nalezené segmenty nemusejí pokrývat celou délku úsečky. Úplná čára potom vznikne pospojováním blízkých segmentů.

Spojování segmentů čar Při spojování segmentů čar je nutné zajistit, aby bylo možné spojit pouze takové segmenty, které směřují stejným směrem. Z toho důvodu je v první řadě nejdříve zkontrolována směrnice obou přímek , u kterých se zjišťuje možnost spojení. Pokud mají přímky stejný směr, je následně vypočítána vzdálenost obou přímek. Vzdálenost je počítána pomocí vzorce pro výpočet vzdálenosti bodu od přímky. V takovém případě je nutné zkontrolovat vzdálenost 3 ze 4 koncových bodů segmentů čar od druhého segmentu.

V případě, že vzdálenost je menší než stanovená maximální vzdálenost pro vyhodnocení, jedná se o segmenty stejné linie a oba segmenty jsou spojeny. U tohoto procesu také závisí na směrnici úsečky. Ta totiž určuje, zda se pro ustanovení koncových bodů nového segmentu použije osa x nebo osa y. Pokud je úhel úsečky svírající s osou x v rozmezí 45° až 135° je jedním koncovým bodem nově vzniklé úsečky zvolen bod z původní čtveřice, který má nejvyšší hodnotu souřadnice y, druhým koncovým bodem je potom ten, který má hodnotu y minimální. Ve zbylých hodnotách, kterých může nabývat úhel úsečky se k ustanovení nově vzniklých koncových bodů, využije stejného přístupu pouze s tou obměnou, že je k výběru minimální a maximální hodnoty použita souřadnice x.



Obrázek 3.4: Spojování segmentů čar: vlevo je k ustanovení přímky použita souřadnice y, vpravo souřadnice x

V ověřování potřeby spojení dvou segmentů se pokračuje do té doby, než bude ověřeno, že žádné další dva segmenty s podobným úhlem naklonění nelze sloučit do jednoho. Poté, co výpočet dospěje do tohoto bodu, jsou zbylé segmenty považovány za výsledné čáry v obraze a zpracování pokračuje výpočtem průsečíků těchto čar.

3.3 Výpočet homografie

Definice problémů výpočtu homografie

Za účelem hromadného zpracování testů je také nutné zpracovávanou stránku porovnat se vzorovým formulářem a tím zjistit, jaké buňky se mají z tabulky vysegmentovat a jejich obsah předat k rozpoznání textu. V praxi je tento problém často řešen nalezením klíčových bodů v obou obrazech a následně jejich vzájemným přiřazením pomocí deskriptorů. Deskriptory svou hodnotou určitým způsobem popisují hodnoty okolí nalezeného klíčového bodu. Tento přístup má dobré výsledky, pokud hledáme přesnou kopii vzorového obrazu. Nicméně v problému řešeném touto bakalářskou prací je jisté, že zpracovávaný obraz přesně neodpovídá vzorovému dokumentu.

Idea nalezení a přiřazení klíčových bodů však zůstala. S jistotou lze tvrdit pouze to, že zpracovávaná stránka bude mít stejnou strukturu tabulky, ale obsahem a okolím této tabulky se může test od testu lišit. Proto nalezení klíčového bodu musí být právě v čarách detekované tabulky.

Jak už bylo řečeno, k vzájemnému přiřazení klíčových bodů nemůžeme použít deskriptory, které jsou odvozeny pouze od hodnot okolních pixelů. Protože však pracujeme nad naskenovanými dokumenty a otočení strany tak bude minimální, můžeme vzájemně přiřazovat klíčové body na základě kritéria vzdálenosti.

U rozpoznaných bodů tedy musí být zaručeno, že budou v čarách strukturujících detekované tabulky a zároveň, že si použité klíčové body budou v jednotlivých obrazech vzájemně odpovídat pozicí v tabulce.

Body, které splňují tyto požadované vlastnosti, jsou průsečíky čar v tabulce. Můžeme u nich vždy zaručit, že jsou obsaženy v čarách tabulky. Zároveň je jisté, že v tabulce budou vždy na stejné pozici. Další pozitivní vlastností průsečíků v tabulce je to, že na základě jejich pozice můžeme segmentovat jednotlivé buňky tabulky. Nastává tedy problém jak detekovat jednotlivé průsečíky v tabulce. Při navrhování aplikace byly vyzkoušeny dva přístupy.

Prvním otestovaným přístupem bylo využití morfologických operací stejně jako při detekci struktury tabulky. V obraze byly pomocí stejných strukturovacích elementů a morfologických operací dilatace a eroze zvlášť rozpoznány horizontální a vertikální linie. Nicméně

tento proces byl použit až nad obrazem obsahujícím pouze detekovanou tabulku nejvyšší úrovně bez dalšího obsahu. Vznikly tedy dva jednotlivé obrazy, jeden obsahoval vertikální čáry tabulky a druhý obsahoval čáry horizontální. Pokud byly poté oba tyto obrazy sloučeny do jediného s použitím operace *and*, zůstaly v obraze pouze objekty na místech, kde se obě struktury překrývají. Protože jen těžko lze předpokládat, že detekované čáry budou definovány jen o tloušťce jednoho pixelu, takže ani místo překryvu takových čar není jen jeden jediný pixel. Proto z této skupiny pixelů potřebujeme vybrat jeden konkrétní bod, který bude reprezentovat jeden průsečík tabulky. Nejdříve je třeba získat pozici a velikost objektů, z toho důvodu byly nalezeny kontury takových objektů pomocí algoritmu sledování hranice. Tyto kontury jsou poté využity k nalezení momentů objektu, ze kterých lze odvodit střed celého útvaru. Tento postup spolehlivě vracel průsečíky v tabulce. Bohužel postup ale neumožňuje při výpočtu kontrolu, zda všechny detekované průsečíky přísluší pouze čarám tvořícím tabulku a nejsou tedy způsobeny jinou čárou, která nebyla odstraněna při detekci tabulky.

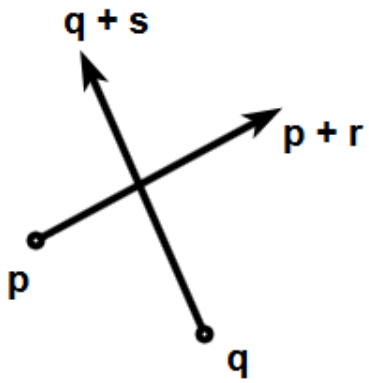
Větší kontrolu nad detekovanými průsečíky umožňoval přístup, ve kterém nad obrazem obsahujícím detekovanou tabulku byl použit algoritmus pro nalezení čar použitím progresivní probabilistické Houghovy transformace. Tímto postupem získáme koncové body detekovaných čar v obraze, od kterých lze odvodit rovnice nalezené čáry. Pomocí těch lze poté vypočítat jednotlivé průsečíky a zároveň tento postup umožňuje kontrolu počtu nalezených průsečíků pro každou z čar.

Výpočet průsečíků

Při výpočtu průsečíků se testuje existence společného bodu na čarách rozdělených na vertikální a horizontální. Tím je omezen výpočet pro čáry, které se nemohou protnout.

K výpočtu průsečíků je použit vektorový součin. Tato metoda je 2D úprava metody hledání průsečíků ve 3D uvedené v [9].

Metoda oba segmenty definuje jako počáteční bod a vektor, definující směr a velikost segmentu.3.5. Každý bod segmentu p potom můžeme vyjádřit jako $p + tr$, kde $t \in \langle 0, 1 \rangle$, podobně každý bod segmentu q můžeme vyjádřit jako $q + us$.



Obrázek 3.5: Vyjádření segmentů přímek

Průsečík je potom vyjádřen vztahem

$$p + tr = q + us \tag{3.2}$$

Po vyjádření tak vzniknou dvě následující rovnice:

$$t = (q - p) \times s / (r \times s) \tag{3.3}$$

$$u = (q - p) \times r / (r \times s) \tag{3.4}$$

Vzhledem k tomu, že je výpočet realizován vždy jen mezi jedním segmentem vertikálním a jedním horizontálním, může dojít jen ke dvěma možným výsledkům těchto rovnic.

1. $t, u \in \langle 0, 1 \rangle$ Potom se oba segmenty vzájemně protínají v bodě $p + tr = q + us$
2. V případě že $t, u \notin \langle 0, 1 \rangle$ Se segmenty ve své délce neprotnou.

Přiřazení klíčových bodů

Ve chvíli, kdy je detekována tabulka a její průsečíky, je třeba ověřit, že je strana opravdu kopií zadaného vzoru formuláře. To lze zjistit porovnáním počtu a pozic nalezených klíčových bodů s klíčovými body vzorového dokumentu.

Protože jsou zpracovávané stránky zatíženy objekty přidanými při vyplňování, tak i přes robustnost detekce tabulky může dojít k detekci jiné struktury načrtnuté vyplňujícím nebo k zahrnutí některé z možných vnořených tabulek do výběru. Z tohoto důvodu bylo před další krok do návrhu aplikace přidáno přiřazování klíčových bodů. U nalezených klíčových bodů je známa pouze jejich pozice a vzhledem k tomu, že od zpracování se očekává robustnost vůči natočení, není garance, že na porovnávaných stránkách si body s nejmenším rozdílem v pozici v obraze odpovídají umístěním v detekované tabulce. Informací, která je jistě zachována v obou sadách nalezených bodů, je jejich vzájemná vzdálenost. Vzdálenost jednotlivých průsečíků v tabulce není nijak výrazně ovlivněna nakloněním stránky, proto je podle ní možné odstranit ze sady průsečíků zpracovávané stránky ty, které byly detekovány navíc, nebo pro další zpracování strany zanedbat průsečíky vzorové stránky, pro které nebyly nalezeny příslušné kopie.

Segmentace buněk tabulky

Dále je třeba příslušná místa tabulky vysegmentovat a předat ke zpracování textu. Pozice a velikost položek, které si uživatel přeje vysegmentovat, jsou známé pouze pro vzorový formulář. Proto je v tomto kroku použit výpočet homografie. Výsledkem operace je poté transformační matice. Po použití této nalezené matice na zpracovávanou stránku se průsečíky vzorové strany a zpracovávané strany nacházejí na stejných pozicích. Díky tomu může být provedena segmentace buňky z přesně stejné pozice jako u vzorového formuláře.

3.4 Rozpoznání ručně psaného textu

V prvním návrhu aplikace bylo plánováno pro rozpoznání ručně psaného textu použít externího nástroje. Tímto nástrojem byla knihovna Tesseract[1], která se sice specializuje na tištěné písmo, ale je i poměrně robustní. Nicméně výsledky vracené touto knihovnou nebyly dostačující, proto se od tohoto nástroje ustoupilo.

Pro další postup bylo navrženo použití knihovny Lipi[2], která se na rozdíl od Tesseractu specializuje přímo na ručně psané písmo. Po bližším prozkoumání knihovny se ukázalo její

[1] https://github.com/tesseract-ocr/
[2] http://lipitk.sourceforge.net/

zapojení do aplikace za velice obtížné, protože její vývoj je už několik let ukončen a není pro ni vytvořena dostatečná dokumentace.

Po těchto neúspěšných pokusech bylo od použití externího nástroje upuštěno a do výsledného návrhu bylo přidáno vytrénování vlastního klasifikátoru pro rozpoznání znaků pomocí techniky *template matching*.

V prvním kroku musí být vytrénován klasifikátor. V aplikaci je použit N-třídní klasifikátor *Support vector machine*, kterému byly k natrénování předána data ve formě histogramu orientovaných gradientů vzorových obrázků znaků.

Stejně jako u zpracování strany je obsah rozpoznávané buňky nejdříve vyhlazen pomocí Gaussova vyhlazování. Následně je převeden na binární hodnoty pomocí adaptivního prahování a dilatací pomocí strukturovacího elementu ve tvaru kříže jsou zvýrazněny a znovu spojeny tenké linky.

Prvním krokem u rozpoznání textu je oddělení jednotlivých znaků. Od znaků rozpoznávaných výslednou aplikací se očekává, že budou ručně psané a zároveň že od sebe budou čitelně oddělené. V tomto kroku jsou v buňce určené k rozpoznání obsahujícího textu nalezeny kontury jednotlivých objektů a následně je kolem těchto objektů nalezen obklopující obdélník. Dále jsou jednotlivě zpracovávané takto nalezené obdélníky.

Pro přiřazení znaku k nějaké z šablon však nejsou použita samotná obrazová data, ale *histogram orientovaných gradientů*, který popisuje přechody hodnot jasu v obraze včetně směru, ve kterém probíhají. Nicméně výsledná podoba takového histogramu silně závisí na natočení znaku, z toho důvodu jsou objekty, které byly detekované v předchozím kroku, pomocí momentů narovnány. Až následně je pro jednotlivé znaky vypočítán histogram orientovaných gradientů, který je předán vytrénovanému N-třídnímu klasifikátoru support vector machine. Ten určí, kterému ze znaků tento histogram nejvíce odpovídá.

3.5 Uživatelské rozhraní

V této části kapitoly je popsán postup návrhu uživatelského rozhraní. Výsledné uživatelské rozhraní má umožnit uživateli vložit dokumenty určené ke zpracování. Současně je nutné od uživatele intuitivně získat umístění dat v tabulce, která jsou pro něj důležitá, takže by mu mělo GUI zobrazit možnosti zvolení těchto buněk.

Pro správné zpracování aplikace vyžaduje tři vstupy:

- Prvním vstupem je vzor testu - Jedná se o elektronickou podobu testu ve formátu PDF, která sloužila k tisku formulářů. Vzhledem k tomu, že se jedná o nevyplněný originál, je usnadněno zpracování dokumentu, protože data neobsahují objekty vzniklé při vyplňování. Současně je originál vygenerován v elektronické podobě, takže není zatížen chybami vstupního zařízení.

- Druhým vstupem jsou naskenované opravené testy - Stejně jako u prvního vstupu se jedná o soubory typu PDF. Nicméně se jedná o mnohastránkový dokument, který je nutné před zpracováním rozdělit na jednotlivé stránky. Současně může obsahovat stránky s výpočty, které nejsou určené k segmentaci dat.

- Posledním třetím vstupem je upřesnění, které buňky tabulky jsou určené pro segmentaci a rozpoznání. Na tato data je možné uživatele se dotázat až po zpracování vzorové stránky.

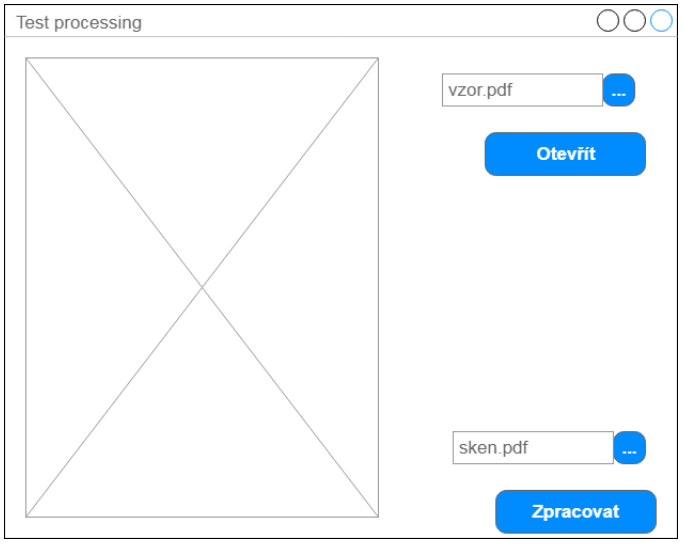
Uživatelské rozhraní bylo navrhováno tak, aby obsahovalo co nejméně ovládacích prvků a bylo pro uživatele co možná nejintuitivnější. Veškeré prvky uživatelského rozhraní jsou

umístěny pouze v jednom okně, aby uživatel nemohl být zmaten přechody mezi jednotlivými obrazovkami.

Většina z očekávané cílové skupiny uživatelů je z Evropy případně ze severní Ameriky. V těchto částech světa je pro drtivou většinu obyvatel přirozené při psaní i čtení postupovat odshora dolů. Proto jsou tímto způsobem za sebe řazené i jednotlivé prvky uživatelského rozhraní v takovém pořadí, které se při zadávání dat očekává. Jako první je tedy zadáván vzor formuláře. Následně je v něm vybráno umístění požadovaných dat a na závěr je vybrán dokument se skeny testů. Ve výše zmíněných oblastech, do kterých spadají cíloví uživatelé, se také běžně píše směrem zleva doprava, z tohoto důvodu je finální ovládací prvek - tlačítko spouštějící samotné zpracování - umístěn na pomyslném konci okna tedy vpravo dole.

V předchozím odstavci je shrnut postup zadávání jednotlivých vstupů. Nicméně pro minimalizování frustrace uživatele při používání aplikace, mohou být vstupy programu libovolně měněny a to až do chvíle, kdy uživatel stiskem tlačítka zahájí proces zpracovávání a tím potvrdí své volby.

První verze návrhu uživatelského rozhraní neobsahovala prvek pro zadání originálu zpracovávaného formuláře. V tomto případě bylo zamýšleno jako vzor pro další zpracování použít první stranu opravených testů. Nicméně v takovém případě chyba při detekci vzorové strany způsobila vysokou chybovost celého výsledku. Z tohoto důvodu byl přidán rušivými vlivy nezatížený originál, který zaručuje, že chybné zpracování jednoho příliš poškozeného skenu neovlivní další strany.



Obrázek 3.6: Schéma navržení výsledného uživatelského rozhraní

V takto navrženém rozhraní bylo plánováno při vybírání zájmových dat právě označenou buňku vysegmentovat a zobrazit náhled jejího obsahu uživateli. V důsledku přidání ke zpracování i originálu sloužícího jako vzorová stránka, by se v tomto kroku zobrazovaly pouze nevyplněné buňky, které by uživateli nijak nepomohly ověřit správnost jeho zadání. Následně tedy bylo od zobrazení buněk upuštěno a uživatelův výběr je vizualizován přímo v obraze vzorové stránky barevným vyznačením. Návrh použitého grafického uživatelského rozhraní je zobrazen na obrázku 3.6.

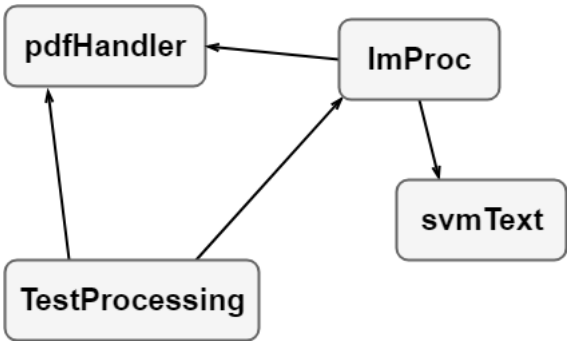
Kapitola 4

Aplikace TestProcessing

Aplikace TestProcessing je implementací výše navržené metody. Aplikace byla implementována v jazyce C/C++. Při implementaci bylo využito možnosti objektově orientovaného programování, které tento jazyk umožňuje.

4.1 Struktura aplikace

Struktura aplikace je zobrazena na obrázku 4.1. Aplikace TestProcessing je rozdělena do čtyř hlavních tříd.



Obrázek 4.1: Struktura aplikace

Třída TestProcessing

Tato třída implementuje grafické uživatelské rozhraní. K jeho vytvoření byl použit multiplatformní aplikační rámec Qt 5[1].

Tato třída obsahuje implementaci jednotlivých metod, které implementují reakce na vstupy od uživatele.

Definice proměnných třídy TestProcessing:

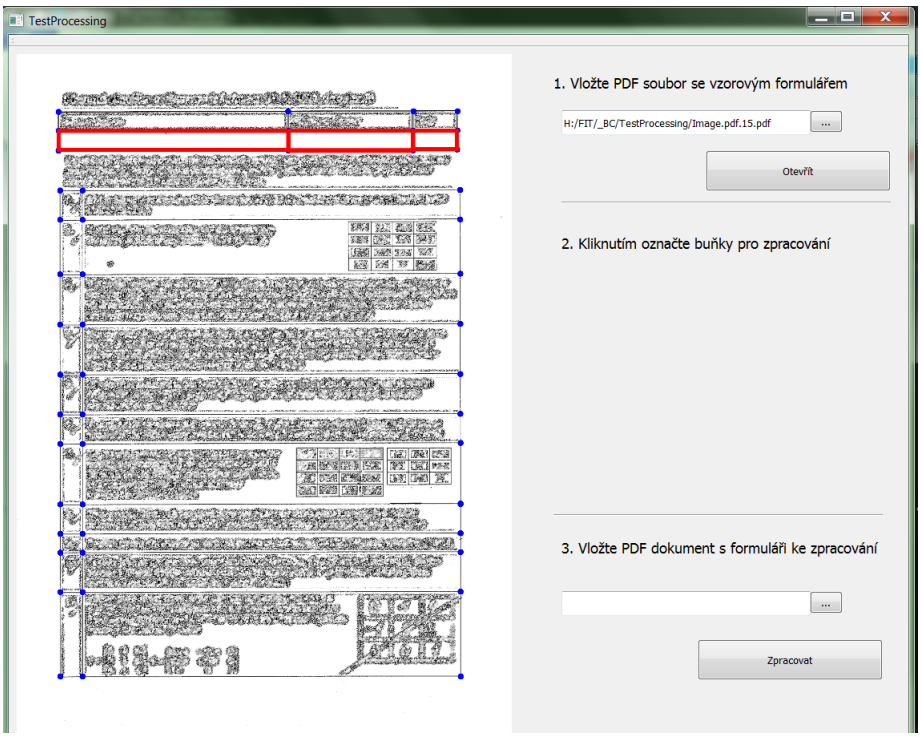
```
QString file1; //cesta k~vzorovemu PDF
QString file2; //cesta k~naskenovanym testum
pdfHandler *p;
ImProc *im;
```

[1] https://www.qt.io/developers/

Jak je vidět výše, jediná data, která třída TestProcessing přímo uchovává, jsou cesty ke vstupním souborům a ukazatele na další třídy. Implementace uživatelského rozhraní je tak zcela oddělena od samotného zpracování v backendu aplikace.

Kromě otevření okna pro výběr souborů zajišťuje tři hlavní činnosti uživatelského rozhraní.

- Po vybrání a potvrzení dokumentu PDF se vzorovou stranou, zašle nejprve pokyn třídě **pdfHandler** k extrakci obrazových dat z tohoto souboru. Následně zavolá metodu třídy **ImProc**, která modelovou stranu zpracuje. Vrácený obraz strany je poté zobrazen uživateli i s výslednými detekovanými klíčovými body, aby uživatel mohl včas zaznamenat špatnou detekci.
- Při následném kliku na zobrazenou vzorovou stranu zašle souřadnice označené polohy v obraze třídě **ImProc**, která podle nich najde příslušnou buňku tabulky. Ta je vyznačena na zobrazené vzorové straně pro kontrolu uživateli. Uživatelské rozhraní s označenými vybranými buňkami lze vidět na obrázku 4.2
- Po vybrání dokumentu se skeny PDF a stisknutí tlačítka zahájení zpracování, kterým uživatel potvrzuje všechny vstupy, je zavolána metoda třídy **ImProc**, která realizuje veškeré další zpracování.



Obrázek 4.2: Okno aplikace po zobrazení vzorové strany a označení buněk ke zpracování

Třída pdfHandler

Tato třída v aplikaci zajišťuje extrakci obrazových dat z dokumentů PDF. K tomu využívá knihovny **MuPDF** viz kapitola 2.1. Současně je zde zajištěno dělení vstupních dokumentů na jednotlivé strany a zjištění počtu stran ke zpracování.

Této třídě je nejprve předána cesta k souboru, který má zpracovat. Následně je volána metoda pro extrakci obrazu, kdy je specifikováno číslo požadované stránky.

Třída ImProc

V této třídě probíhá celé zpracování obrazu. K tomu je využita knihovna **OpenCV**[2] (Open Source Computer Vision Library) ve verzi 3.3.1.

Definice proměnných třídy ImProc

```
vector<Point> model, strana; //Klicove body
vector<Rect> cells;    //Vybrane bunky
Mat modelPage;     //Vzorova strana
svmText *svm;     //trida svmText
int size;      //Pocet stran ke zpracovani
```

Jednotlivé klíčové body modelové a právě zpracovávané strany jsou uchovávány ve *vektorech*, které zaručují konstantní dobu přístupu k jednotlivým prvkům. Průměrná časová složitost operací vkládání a odebírání z konce vektoru je také konstantní. Zároveň podporují řazení, které má garantovanou lineárně logaritmickou časovou náročnost.

Třída pro každou stranu zpracovávaného dokumentu zavolá metodu třídy **pdfHandler** pro extrakci obrazových dat. Následně provede nalezení klíčových bodů. Pokud je nalezena podobnost, je vypočítána homografie se vzorovou stranou. Následně jsou vysegmentovány uživatelem zvolené buňky a obrazová data jejich obsahu jsou zaslány na rozpoznání textu třídě **svmText**.

Třída svmText

Ve třídě svmText je implementována segmentace jednotlivých znaků a jejich rozpoznávání. Stejně jako u předchozí třídy tato část využívá knihovnu **OpenCV**.

K samotnému rozpoznání znaků je použit N-třídní klasifikátor *Support vector machine*. Jako trénovací sada byla využita **NIST**[3] databáze ručně psaných znaků. Ukázka použitých dat je na obrázku 4.3. Z této databáze bylo použito 500 vzorů každého písmene a číslice, kde 450 sloužilo k trénování klasifikátoru a zbylých 50 k výpočtu výsledné přesnosti.

Obrázek 4.3: Ukázka vzorků trénovací sady pro písmeno X

[2] https://opencv.org/
[3] https://www.nist.gov/srd/nist-special-database-19

Jednotlivé vzorky znaků byly nejprve narovnány pomocí momentů a následně popsány pomocí *histogramu orientovaných gradientů*. Až tyto histogramy byly předány klasifikátoru **SVM** pro trénování a rozpoznání.

Při klasifikaci rozpoznávaného znaku potom tento klasifikátor vrací pouze hodnotu třídy, do které znak zařadil.

4.2 Testování

Testování aplikace probíhalo ve dvou krocích. V prvním kroku byla otestována schopnost algoritmu detekovat tabulku pomocí testovací sady **Marmot**[4]. Z té bylo náhodně vybráno 100 dokumentů PDF, které obsahovaly různé tabulky.

V druhém kroku byla otestována celková funkčnost aplikace. Zde byla aplikace spuštěna s reálnými vstupními daty, která jsou tvořena naskenovanými ručně opravovanými formuláři.

Testování probíhalo na stroji se systémem Windows 7 a s procesorem Intel(R) Core(TM) i7-6500U.

Testování detekce tabulky

Úplně prvním krokem při spuštění aplikace je analýza vzorové strany formuláře. Na této straně aplikace musí spolehlivě rozpoznat průsečíky. Pokud je na vzorové stránce nedostatečně detekována tabulka, je touto chybou zatížen celý další chod zpracování. Z tohoto důvodu byla v prvním kroku otestována schopnost aplikace spolehlivě rozpoznat tabulku nejvyšší úrovně na vyplňováním nezatíženém formuláři. Jak už bylo výše uvedeno, k testování byla použita sada **Marmot** pro detekci tabulek. Z této sady bylo použito 100 dokumentů PDF.

Testová sada obsahovala 15 dokumentů, ve kterých se nevyskytovala žádná tabulka, nebo jen taková tabulka, která nebyla čarami ohraničena. Současně se v sadě vyskytovaly dokumenty, které ve svém obsahu měly několik oddělených tabulek.

Při testování byl kladen důraz nejen na schopnost detekce tabulky, ale zároveň i zanedbání hustého textu a mimo tabulku ležících čar. Příklad testovacích dat s detekovanými tabulkami je vidět na obrázku 4.4.

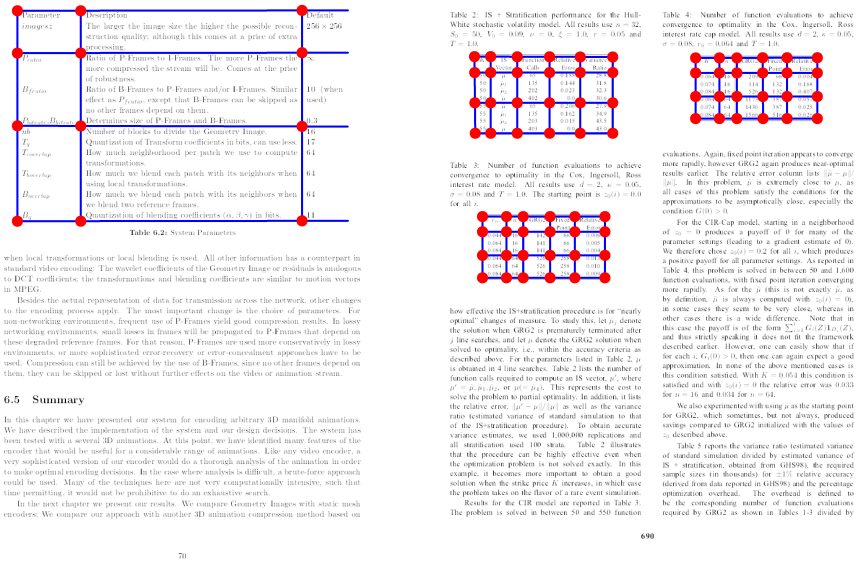
Pro vyhodnocení se počítaly detekované průsečíky a správně detekované tabulky. Za správně detekovanou tabulku se považovala pouze ta, u které byly nalezeny veškeré její průsečíky. Výsledné naměřené hodnoty jsou uvedeny v tabulce 4.1.

	Správně detekované	Celkem testované	Úspěšnost
Průsečíky tabulek	2564	2664	96,25%
Tabulky	81	97	83,5%

Tabulka 4.1: Výsledky detekce tabulky

Ačkoliv statisticky byly detekovány 4/5 tabulek, z úspěšnosti detekce průsečíků je vidět, že se vždy jednalo o malou část tabulky, která byla špatně detekována. Při testování se ukázaly pro detekci jako zásadní šířky čar v rozpoznávané tabulce. Většina případů nenalezení průsečíků byla způsobena slabým ohraničením, jehož čára byla slabší než 1,5 pt.

[4] http://www.icst.pku.edu.cn/cpdp/data/marmot_data.htm



Obrázek 4.4: Ukázka testovacích dat s vyznačenými detekovanými tabulkami a jejich průsečíky

Pokud nebyla dodržena výše zmíněná šířka čar tabulky, záleželo rozpoznání silně na pozici tabulky při renderování stránky. I při nedodržení byla vždy rozpoznána alespoň polovina průsečíků tabulky.

Metoda se ukázala jako spolehlivá pro rozpoznání několika oddělených tabulek na stránce. Zároveň byly z detekce úspěšně vynechány objekty oddělené od tabulky i v případě hustého textu nebo jiných čar.

Reálné použití

V této části testování byla aplikace spuštěna nad reálnými vstupy. Datová sada byla tvořena 20-ti stránkovým PDF dokumentem, který obsahoval 10 testů odpovídajících vzorové stránce a 10 stránek s poznámkami, které obsahovaly různé náčrtky.

Při testování byly pro segmentaci z 26 buněk ve formuláři vybrány 2, které obsahovaly kombinaci jak číslic tak i písmen.

Při testování se sledovala správná detekce tabulky a následně segmentace buněk ze správného umístění. U rozpoznání textu bylo testování zaměřeno na schopnost segmentace jednotlivých znaků a úspěšnost jejich rozpoznání.

Na stejné datové sadě byl také měřen čas načtení vzorové strany a doba kompletního zpracování naskenovaných testů.

	Doba v sekundách
Zobrazení vzorové stránky	7,9 s
Celková doba zpracování	141,6 s

Tabulka 4.2: Čas potřebný pro zpracování

	Úspěšnost
Oddělení testů od poznámek	100%
Přesné nalezení tabulky v testech	80%
Segmentace buněk	100%
Segmentace textu	88%
Rozpoznání textu	40,9%

Tabulka 4.3: Úspěšnost dílčích kroků zpracování

Po vyhodnocení výsledků je ověřena dostatečná robustnost detekce tabulky a její přirovnání k vzorovému formuláři. Nicméně pro možnost použití aplikace v praxi je třeba za zpracování strany přidat spolehlivější systém pro rozpoznání ručně psaného textu.

4.3 Možnosti vylepšení

Hlavní částí systému, která je pro použitelnost třeba vylepšit, je rozpoznání ručně psaného textu. Současná část aplikace zastupující tuto funkci totiž dosahuje velice špatných výsledků. Jedna z možností, jak tuto část vylepšit, je umožnit v aplikaci zadávat u jednotlivých buněk, zda se v nich nachází pouze text, číslice nebo veškeré alfanumerické znaky. Omezením možných hodnot a použitím různých klasifikátorů vytrénovaných na nižším počtu tříd by se mohla v takových buňkách tabulky znatelně zvýšit úspěšnost rozpoznání. Případně u buněk, které nabývají předem dané množiny hodnot jako je login nebo ID studenta, by mohla být přidána možnost vložení této množiny. Výsledná hodnota buňky by poté byla zvolena z této množiny na základě největší podobnosti rozpoznaného textu.

Dalším námětem k vývoji aplikace je použití externího nástroje pro segmentaci jednotlivých znaků textu. Současná podoba segmentace nezvládá rozlišit spojené znaky.

Poslední uvedenou možností vývoje je řešení upravit do podoby webové aplikace. Nynější řešení veškeré výpočty provádí na lokálním počítači. Z toho důvodu je čerpán procesorový čas a zpracování testů může zpomalovat běh ostatních aplikací na počítači, čímž může částečně omezovat uživatele v další činnosti i přes fakt, že celé zpracování probíhá bez jeho obsluhy.

Kapitola 5

Závěr

Cílem této bakalářské práce bylo navrhnout a vytvořit aplikaci, která hromadně zpracuje ručně opravené testy. Výsledná aplikace pro svůj běh vyžaduje dva dokumenty PDF, kde jeden představuje vzor formuláře a druhý samotné zpracovávané opravené testy. Nejprve je zpracována vzorová strana, ve které je detekována a uložena struktura tabulky. Následně jsou načítány jednotlivé testy, ve kterých je hledána vzorová struktura. Pokud dojde ke shodě, je strana transformována tak, aby se její struktura překrývala se vzorem, a následně jsou vysegmentovány buňky tabulky, které uživatel zadal pro vzorovou stranu. Obsah takto získaných buněk je následně předán k rozpoznání textu. Výsledkem běhu programu je textový soubor, ve kterém jsou ve formátu csv uloženy data z tabulky.

Před samotným vytvořením aplikace bylo třeba nastudovat používané techniky zpracování obrazu zajišťující detekci čar v obraze, segmentaci a v neposlední řadě možnosti rozpoznání ručně psaného textu. Při návrhu aplikace byly jednotlivé techniky experimentálně zkoušeny a výsledky těchto experimentů byly použity k finálnímu návrhu zpracování.

Výsledná aplikace načte vzor formuláře a naskenované testy. Použitý návrh zpracování zvládne robustně detekovat tabulku. Zároveň zvládne úspěšně přiřadit zpracovávaný test ke vzorovému formuláři i s nalezením jen 50% bodů vzorové tabulky.

Na základě provedených testů aplikace, jejichž výsledky jsou shrnuty v kapitole 4.2, se ukázala potřeba dalšího vývoje části programu zajišťující zpracování ručně psaného textu, protože současná implementovaná metoda nedosahuje výsledků použitelných v praxi. Nicméně schopnost metody spolehlivě rozpoznat vzorovou strukturu je výraznou motivací v pokračování vývoje.

Literatura

[1] Adobe Systems Incorporated: *PDF Reference Sixth Edition, version 1.7*. URL https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf

[2] Bradski, G.; Kaehler, A.: *Learning OpenCV*. O'Reilly, 2008, ISBN 978-0-596-5161 -0.

[3] Canny, J.: *A computational approach to edge detection*. Elsevier, 1987.

[4] Chaudhuri, A.; Mandaviya, K.; Badelia, P.; aj.: *Optical Character Recognition Systems for English Language*. ročník 352, 12 2017.

[5] Chaumette, F.: *Image moments: a general and useful set of features for visual servoing. IEEE Transactions on Robotics*, ročník 20, č. 4, Aug 2004: s. 713–723, ISSN 1552-3098.

[6] Dartmouth Research Computing: *HOUGH*. [Online; navštíveno 20.04.2018]. URL http://northstar-www.dartmouth.edu/doc/idl/html_6.2/HOUGH.html

[7] Derpanis, K. G.: *Overview of the RANSAC Algorithm. Image Rochester NY*, ročník 4, č. 1, 2010: s. 2–3.

[8] Dubrofsky, E.: *Homography estimation. Diplomová práce. Vancouver: Univerzita Britské Kolumbie*, 2009.

[9] Goldman, R.: Graphics Gems. kapitola *Intersection of Two Lines in Three-space*, San Diego, CA, USA: Academic Press Professional, Inc., 1990, ISBN 0-12-286169-5, s. 304–.
URL http://dl.acm.org/citation.cfm?id=90767.90838

[10] Mallick, S.: *Histogram of Oriented Gradients*. [Online; navštíveno 10.03.2018]. URL https://www.learnopencv.com/histogram-of-oriented-gradients/

[11] Matas, J.; Galambos, C.; Kittler, J.: *Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. Computer Vision and Image Understanding*, ročník 78, č. 1, 2000: s. 119 – 137, ISSN 1077-3142.
URL http://www.sciencedirect.com/science/article/pii/S1077314299908317

[12] Miloš, J.: *Cannyho operátor a další používané hranové detektory*. Bakalářská práce, FIT VUT v Brně, 2008.

[13] Mori, S.; Suen, C. Y.; Yamamoto, K.: *Historical review of OCR research and development. Proceedings of the IEEE*, ročník 80, č. 7, Jul 1992: s. 1029–1058, ISSN 0018-9219.

[14] Peterlin, P.: *Morphological Operations: An Overview*. Červenec 1996, [Online; navštíveno 24.04.2018].
URL http://www.inf.u-szeged.hu/ssip/1996/morpho/morphology.html

[15] Sharpio, L.; Stockman, G.: *Computer Vision*. Prentice-Hall Inc., 2001, ISBN 0-13-030796-3.

[16] Sobotka, L.: *Panorama z Ptačí Perspektivy*. Bakalářská práce, FIT VUT v Brně, 2015.

[17] Sonka, M.; Hlavac, V.; Boyle, R.: *Image processing, analysis, and machine vision*. PWS Publishing, 1998, ISBN 0-534-95393-X.

[18] Verne, J.: *Image Pre-Processing*.

[19] Vynckier, I.: *How OCR works*. [Online; navštíveno 15.04.2018].
URL http://www.how-ocr-works.com/OCR/OCR.html

[20] Watts, R.: *MuPDF Explored*.
URL https://ghostscript.com/~robin/mupdf_explored.pdf

Příloha A

Obsah přiloženého paměťového média

- **doc** - dokumentace vytvořeného technického díla
- **install** - instalační soubor aplikace TestProcessing
- **src** - zdrojové soubory aplikace
- **tex** - technická zpráva ve formátu PDF a její zdrojové soubory
- **media** - video a plakát