FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Automatické zpracování tištěné papírové účtenky za palivo
Student:	Martin Šafránek
Vedoucí:	Ing. Václav Jirovský, Ph.D.
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

V rámci práce analyzujte metody automatického rozpoznávání znaků a textu použitelné pro analýzu účtenek s jasně definovaným obsahem.
Připravte algoritmus, který bude zpracovávat fotografii účtenky a vyčítat definovaná data (částka, čas, datum).
Účinnost realizovaného algoritmu vyhodnoťte, specifikujte omezení a navrhněte případná zlepšení.
Výsledek implementujte v prostředí studentského projektu sdílení vozidel Uniqway.
Při realizaci cíle využijte týmové spolupráce se studenty realizujícími projekt Uniqway.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Karel Klouda, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 14. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Automatické zpracování tištěné papírové účtenky za palivo

Martin Šafránek

Katedra aplikované matematiky
Vedoucí práce: Ing. Václav Jirovský, Ph.D.

4. června 2020

Poděkování

Děkuji svému vedoucímu panu Ing. Václavu Jirovskému, Ph.D., za pomoc s touto prací a vstřícný přístup. Zejména pak při shromažďování účtenek a jejich anotováním. Dále děkuji své rodině a přátelům za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne . . . . . .

Odkaz na tuto práci

Šafránek, Martin. *Automatické zpracování tištěné papírové účtenky za palivo.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Práce porovnává úspěšnosti několika postupů pro rozpoznání ceny, času a datumu na tištěné účtence za palivo s použitím OCR nástroje Tesseract. V předzpracování účtenky zkoumá vliv Sauvolova binarizačního algoritmu a mediánového filtru. Pro odhad zkosení účtenky srovnává Houghovu transformaci a metodu lineární regrese s K-means. Celé řešení je implementováno ve formě webové aplikace a porovnáno s Google Vision. Nejlepší úspěšnost řešení je pro datum, které na účtence nalezne v 73 % případů v porovnání s 92 % u Google Vision.

Klíčová slova rozpoznávání textu, odhad zkosení, extrakce faktů, tištěný text, předzpracování účtenky, Tesseract, Google Vision, ORC

Abstract

The thesis compares several methods for price, time and date recognition on printed fuel receipt using OCR tool Tesseract. In preprocessing step, Sauvola binarization and median filter impact is studied. For skew estimation, Hough transformation and linear regression with K-means are compared. This process is implemented as a web application and compared with Google Vision. The best method result is 73% for date extraction compared to 92% with Google Vision.

Keywords text recognition, skew estimation, fact extraction, printed text, receipt preprocessing, Tesseract, Google Vision, OCR

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Většinu činností firmy lze do nějaké míry automatizovat. Právě míra a kvalita této automatizace dokáže firmě poskytnout konkurenční výhodu. Příkladem je zpracování tištěných dokumentů. Firma je potřebuje roztřídit do několika kategorií. Extrahovat z nich důležité informace a výsledek uložit do databáze. Tato činnost je pro člověka velice časově náročná a náchylná na chyby. Její automatizací se celý proces výrazně zrychlí, zlevní a bude poskytovat konzistentní výsledky.

Zpracování tištěných dokumentů je komplexní problém a neexistuje jeden správný postup. Jeho nejdůležitější částí je OCR (optical character recognition). OCR je systém, který má na vstupu obrázek s textem a na výstupu rozpoznaný text. Většina OCR systémů hledá v obrázku vzory, které připomínají písmena. Z nich pak sestaví rozpoznaný text. Hledání vzorů je stěžejní částí OCR a jeho úspěšnost výrazně ovlivňuje kvalita vstupního obrázku.

Kvalita vstupního obrázku se dá zlepšit jeho předzpracováním. V případě tištěných dokumentů to může být správné natočení, odstranění šumu, binarizace nebo ořezání. Platí, že každý OCR systém vyžaduje jiné předzpracování. Například jeden OCR systém si poradí s neúplným vzorem, druhý nikoliv. Navíc pro každý typ obrázku je vhodné jiné předzpracování. Toto vede na komplexní problém. Pro danou kombinaci typu obrázku se musí zvolit správný postup předzpracování a OCR nástroj.

Hlavním cílem této práce je rozpoznání textu na tištěné papírové účtence za palivo a extrakce ceny, času a datumu uvedeného na účtence. K dosažení tohoto cíle je použit open-source OCR nástroj Tesseract. Dílčím cílem je celé řešení otestovat a nasadit ve formě jednoduché webové aplikace, kam bude možné nahrát obrázek tištěné účtenky a obdržet uvedenou cenu, čas a datum.

Struktura práce je rozdělena do osmi kapitol. V kapitole 1 je popsán OCR nástroj Tesseract. Nachází se zde popis jeho fungování, možnost použití vlastních trénovacích dat, jeho alternativy, včetně popisu komerčního OCR nástroje Google Vision. Také je zde popis použití nástroje Tesseract v této práci.

V kapitole 2 je popsán shromážděný dataset účtenek. Je zde popsáno rozdělení datasetu na dobré a špatné účtenky, včetně jejich upravení před dalším zpracováním. Také se zde nachází ukázky účtenek společně s jejich nejčastějšími vadami. Kapitola 3 popisuje předzpracování účtenky. Je zde popsáno odstranění šumu pomocí mediánového filtru a dva binarizační algoritmy — Otsův a Sauvolův. Kapitola 4 popisuje dva způsoby, jak lze rotovat účtenku do správného úhlu. První z nich je založen na Houghově transformaci. Druhý na metodě lineární regrese. Ta používá několik kroků, nejzajímavější je však v této práci nově navržená dynamická volba strukturního elementu pomocí algoritmu K-means. V kapitole 5 je popsán postup extrakce ceny, času a datumu z textu. Důležitá je znalost formátu těchto faktů, proto je zde i popis formátu ceny, času a datumu v získaném datasetu. Kapitola 6 popisuje implementaci webové aplikace a její strukturu — business a prezentační vrstvu. Dále je zde postup nasazení webové aplikace na kterémkoliv počítači podporující nástroj Docker. Nejrozsáhlejší je kapitola 7. V ní je popsaná metodika testování a výsledky tří sad testů. První sada obsahuje 8 testů a zkoumá přesnost celého řešení. U jednotlivých testů se liší parametry použitých algoritmů. Druhá sada obsahuje 4 testy a zkoumá dva algoritmy pro rotaci účtenky. Poslední sada testů obsahuje 1 test a zkoumá komerční OCR nástroj Google Vision. Výsledky testů jsou zde shrnuty a porovnány s výsledky ostatních autorů. V poslední kapitole 8 jsou popsány návrhy na zlepšení a rozšíření celého postupu. Součástí práce je i obsáhlejší příloha. V příloze A se nachází tabulky s parametry algoritmů pro jednotlivé sady testů a jejich krátký popis. Také jsou zde tabulky s klíčovými slovy a příklady formátů použitých v extrakci faktů. Příloha B obsahuje grafovou přílohu. Jsou v ní vizualizované výsledky testování, které se nevešly do hlavní části. Příloha C obsahuje seznam použitých zkratek. Příloha D obsahuje popis přiloženého média, kde se mimo jiné nachází použitý dataset a spustitelná aplikace.

Kapitola 1

Tesseract

Tato kapitola popisuje použitý nástroj Tesseract. Stručně vysvětluje jeho fungování, možnost použití trénovacích dat a jeho možné alternativy. Nakonec je popsáno jeho nasazení ve formě API.

Tesseract je OCR nástroj pro rozpoznávání textu. Byl vyvinut firmou Hewlett-Packard v letech 1984 až 1994. Od té doby jeho vývoj stagnoval. V roce 2005 ho HP uvolnila pod Licencí Apache v2.0. Stává se z něj tedy open-source. Vývoj od roku 2006 sponzoruje Google [1]. Díky jeho podpoře se Tesseract dočkal mnoha vylepšení. Verze použitá v této práci [2] vydaná v prosinci 2019 podporuje 131 jazyků, včetně češtiny. Od verze 4.0 vydané ke konci roku 2018 Tesseract používá neuronovou síť typu LTSM [3]. Ta zlepšuje přesnost a nabízí lepší možnosti trénování.

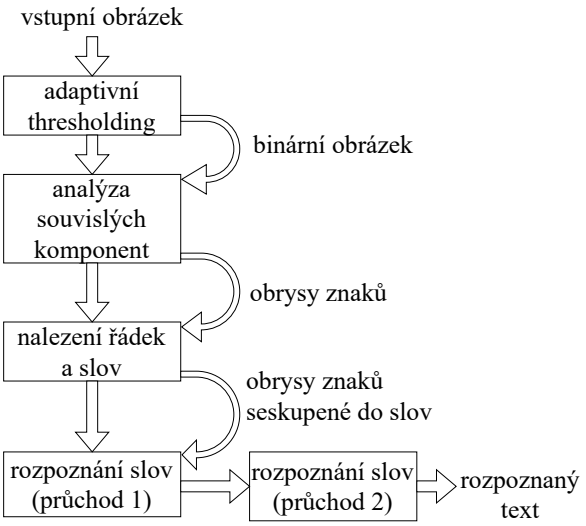
1.1 Princip fungování

Fungování nástroje Tesseract lze rozdělit do několika kroků, které na sebe navazují. Graficky jsou tyto kroky znázorněny na obrázku 1.1. Jejich podrobný popis je v pracích [4, 5], z kterých tato sekce vychází. Obě práce jsou ale staršího data. Neobsahují tedy popis nejnovějších funkcí.

První krok je adaptivní thresholding. Ten převede vstupní obrázek na binární obrázek. Tesseract k tomu podle zdrojového kódu [6] využívá Otsův algoritmus [7]. Ten však není optimální pro některé vstupní obrázky (více o binarizaci a adaptivním thresholdingu viz kapitola 3).

Následuje nalezení a analýza souvislých komponent. V této fázi se uloží obrysy souvislých komponent, s kterými se pracuje dále. Spojením obrysů se vytvoří bloby. Z blobů jsou pak sestaveny řádky textu. Ty jsou dekomponovány do jednotlivých slov.

Posledním krokem je dvoufázové rozpoznání slov. Roli tady hraje adaptivní klasifikátor. Slova s velkou pravděpodobností správného rozpoznání jsou dána na výstup a také do adaptivního klasifikátoru. Ten je použije jako trénovací



Obrázek 1.1: Tesseract architektura. Zdroj: [5, překresleno, přeloženo]

data pro zlepšení klasifikace a porozumění textu. V druhé fázi jsou špatně rozpoznaná slova (s malou pravděpodobností) vstupem do opakovaného procesu rozpoznání. Ten ale tentokrát použije adaptivní klasifikátor pro zlepšení přesnosti.

1.2 Trénovací data

Zmíněná verze nástroje Tesseract [2] podporuje 131 jazyků. Podpora jazyka neznamená ale nic jiného, než dostupnost speciálního souboru pro daný jazyk s názvem `language.traningdata`. Ten Tesseract načte při inicializaci a je schopen rozpoznat znaky specifické pro daný jazyk.

Pro vytvoření vlastních trénovacích dat jsou potřeba všechny znaky, které chceme rozpoznávat a můžou se objevit na vstupu. Tyto znaky anotujeme a spustíme trénovací proces. Výsledkem je `language.traningdata` soubor, který můžeme použít při inicializaci nástroje Tesseract. Přesný postup není na oficiálních stránkách dobře zdokumentovaný, existují ale nástroje, které tento proces automatizují. Například v práci [8] použil autor pro vytvoření trénovacího datasetu pro matematické a chemické rovnice program jTessBoxEditor[1].

[1] `http://vietocr.sourceforge.net/training.html`

1.3 Alternativy

V sekci open-source OCR nástrojů je podle mého názoru Tesseract nejlepší volbou. Důvodem je rozsáhlá dokumentace projektu, aktivní komunita podílející se na vývoji a konzistentní rozpoznávací schopnosti. Existuje i řada jiných volně dostupných OCR nástrojů. Většina z nich je ale na nástroji Tesseract založena a nepřináší tím pádem nic nového z pohledu algoritmu pro rozpoznávání textu. Například FreeOCR, WeOCR nebo OCRFeeder jsou OCR systémy pro rozpoznávání textu, které používají Tesseract. Jejich přidaná hodnota je třeba export rozpoznaného textu do Microsoft Word nebo přívětivé grafické rozhraní. Alternativou, která není založená na nástroji Tesseract, může být GORC [9]. V práci [10] ale autor tvrdí, že Tesseract má ve většině případů lepší přesnost.

V komerční sféře je zajímavé řešení Google Vision [11]. To je dostupné pouze ve formě API a není open-source. Podle dostupných informací používá předtrénovaný model. Ten rychle klasifikuje obrázek do tisíce kategorií. Následně rozpozná jednotlivé objekty, detekuje tváře, rozpozná textová pole a pokusí se určit jejich obsah. Nevýhoda Google Vision je zmíněná dostupnost pouze pomocí API. Nelze ho tedy použít v real-time systémech kvůli vysoké latenci. Podle [12] je jeho nevýhoda v trénovacích datech. Ty podle autora Google Vision pravděpodobně získává z databází Googlu. Testováním také prokázal sníženou přesnost při použití snímků z kamery oproti obrázkům vyhledaným na Googlu. Rozdíl ale nebyl velký. Podle [13] lze přidáním malého šumu do obrázku vynutit špatnou klasifikaci obrázku. Toto může podle autora vést k potencionálním bezpečnostním rizikům.

1.4 Tesseract API

Pokud chceme použít Tesseract pro rozpoznání textu přímo, jediný způsob je použití příkazové řádky. Na té se kromě vstupního souboru specifikují také volitelné parametry, kterých je 638. Tento způsob je pro člověka nepřehledný a pro použití v aplikaci někdy i nemožný. Často se pro přístup k nástroji Tesseract používá wrapper. Wrapper je prostředník mezi aplikací a daným programem. Jeho výhodou je možnost ovládání programu z aplikace bez použití formátu komunikace specifikovaného programem.

V této práci se pro komunikaci s nástrojem Tesseract používá Java wrapper Tess4J [14]. Tesseract lze tedy snadno integrovat do webové aplikace podporující Javu s použitím Tess4J. Jeho nejdůležitější prvky jsou popsány v následujícím textu.

Tess4J je nejprve nutné inicializovat. Příkazy ve výpisu 1 se vytvoří a inicializuje nová Tesseract instance. Nejdříve se specifikuje složka, ve které Tesseract najde trénovací data. Dále se může specifikovat jazyk. Pro každý jazyk ale musí být dostupná trénovací data. Ty jsou dostupná spolu s Tesseract dis-

tribucí. Konkrétně pro volbu jazyka `ces` musí ve složce `./tessdata` existovat soubor `ces.traineddata`.

Dalším parametrem, který není povinný, ale je dobré ho nastavit, je DPI vstupního obrázku. Tato volba může do značné míry ovlivnit rozpoznávání, protože určuje velikost hledaných znaků. Vyčíst se může z metadat vstupního obrázku, pokud jsou dostupná. Není-li tato hodnota nastavena, Tesseract nastaví DPI na hodnotu 70. Z doporučených kroků pro zlepšení kvality [15] je udávaná optimální hodnota 300 DPI, při které Tesseract funguje nejlépe. DPI se nastaví pomocí příkazu ve výpisu 2.

Takto nastavený Tesseract už je možné použít pro rozpoznání textu. Výsledek rozpoznání je vždy jednoho typu, viz tabulka 1.1. Dá se tedy nastavit, zda chceme výsledek jako jednotlivé symboly, slova, řádky, paragrafy nebo bloky.

Příkazy ve výpisu 3 spustí rozpoznávání na specifikovaném obrázku. Výsledek vrátí jako entity typu `Word`, které v tomto případě představují jednotlivé symboly. Následně se s každým symbolem vytiskne jeho pravděpodobnost správného určení a souřadnice, kde se symbol v rámci obrázku nachází.

```
Tesseract tessInstance = new Tesseract();
tessInstance.setDatapath("./tessdata");
tessInstance.setLanguage("ces");
```

Výpis 1: Tesseract – inicializace

```
tessInstance.setTessVariable("user_defined_dpi", "300");
```

Výpis 2: Tesseract – nastavení DPI

```
List<Word> wordLst = tessInstance.
        getWords(image, TessPageIteratorLevel.RIL_SYMBOL);
for (Word word : wordLst) {
    System.out.println(word.getText());
    System.out.println(word.getConfidence());
    System.out.println(word.getBoundingBox());
}
```

Výpis 3: Tesseract – rozpoznání textu

typ	**význam**
RIL_SYMBOL	znak ve slově
RIL_WORD	slovo v řádce
RIL_TEXTLINE	řádka uvnitř paragrafu
RIL_PARA	paragraf uvnitř bloku
RIL_BLOCK	blok textu

Tabulka 1.1: Hodnoty TessPageIteratorLevel

Kapitola 2

Dataset účtenek

Tato kapitola popisuje dataset účtenek použitých pro testování a jeho rozdělení do kategorií. Tento dataset byl získán výhradně pro tuto práci a je dostupný pouze na přiloženém médiu.

Dataset obsahuje 233 obrázků účtenek od 43 společností. Tento dataset byl získán postupně z několika zdrojů, především však dobrovolným přispíváním jednotlivců. Takto posbíraný dataset splňuje pouze pár podmínek. Ty jsou: český nebo slovenský jazyk textu na účtenkách, účtenka obsahuje cenu, čas a datum. Naopak obrázky účtenek mají různé rozlišení, font, velikost písma, DPI, úhel, světelnost, kontrast s pozadím, obsahují barevná razítka nebo podpis, jsou foceny různým fotoaparátem a některé jsou neúplné nebo pomačkané. Protože je dataset jednou z nejdůležitějších částí, která ovlivňuje výsledky, je potřeba ho nějakým způsobem pročistit nebo rozdělit do kategorií. Jediná úprava obrázků v datasetu spočívala v částečném ořezání pozadí. V aplikaci na rozpoznávání účtenek tento krok může být simulován uživatelem, který účtenky před nahráním ořízne. Posledním krokem je rozdělení datasetu do tří kategorií.

První kategorie je **dataset dobrých účtenek**. Tento dataset obsahuje účtenky s minimem vad, které mohou vést ke špatnému rozpoznání textu OCR systémem. Naopak **dataset špatných účtenek** obsahuje účtenky, které mají vady, které mohou vést ke špatnému rozpoznání OCR systémem. Poslední kategorie je **dataset účtenek bez rotace**. Tam jsou vybrány účtenky z předchozích dvou datasetů, které nemají natočené řádky, jejich skon je tedy 0°. Je také rozdělen na dobré a špatné účtenky.

Definice 2.0.1. *Dobrá účtenka je taková účtenka, která není kontrastní s pozadím, má rovnoměrné osvětlení, neobsahuje odlesky od světla, není pomačkaná, potrhaná nebo rozmazaná a člověk na ní dokáže bez větších problémů přečíst cenu, čas a datum.*

Definice 2.0.2. *Špatná účtenka je taková účtenka, která není dobrá.*

Na obrázku 2.1 jsou vidět příklady dobrých účtenek. Z nich účtenky na obrázcích 2.1a, 2.1b a 2.1d patří do datasetu účtenek bez rotace, protože nemají žádný sklon. Na obrázku 2.2 jsou vidět příklady špatných účtenek a některé jejich vlastnosti. Povšimněte si, že oba datasety mohou obsahovat nakloněné účtenky.

Po tomto rozdělení obsahuje dataset dobrých účtenek 120 obrázků a dataset špatných účtenek 113 obrázků. Dataset účtenek bez rotace obsahuje 109 obrázků. Nejčastějším důvodem klasifikace účtenky jako špatné je rozmazání nebo nerovnoměrné osvětlení.

(a) (b) (c) (d)

Obrázek 2.1: Příklady dobrých účtenek

(a) kontrast s pozadím (b) nerovnoměrné osvětlení (c) pomačkání (d) nečitelná cena

Obrázek 2.2: Příklady špatných účtenek

Kapitola 3

Předzpracování účtenky

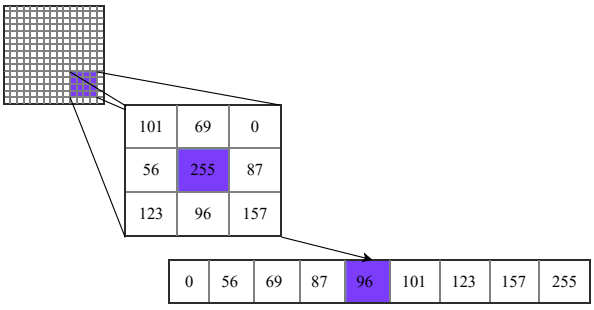
Tato kapitola popisuje předzpracování účtenky. Je zde popsáno odstranění šumu pomocí mediánového filtru a dva binarizační algoritmy — Otsův a Sauvolův.

Jednotlivé kroky jsou pak použity pro účtenku před její rotací a rozpoznáním textu nástrojem Tesseract. Vstupem do této části je libovolný obrázek účtenky převedený do odstínů šedi. Výstupem je pak obrázek, který lze použít pro rotaci nebo rozpoznávání nástrojem Tesseract.

3.1 Odstranění šumu

Odstranění šumu z obrázku účtenky se musí provádět velice opatrně, pokud vůbec. Důvodem je špatná kvalita a nevýraznost některých tištěných znaků, které mohou být považovány za šum.

Použit je pouze mediánový filtr. U něho se nejdříve zvolí fixní okénko. Střed tohoto okénka je postupně každý pixel ve vstupním obrázku. Pro všechny pixely obsažené v tomto okénku se spočítá medián a jeho hodnota se zapíše do výstupního obrázku. Takto se postupuje pro celý vstupní obrázek. Ukázka fungování pro okénko 3×3 je znázorněna na obrázku 3.1.



Obrázek 3.1: Princip fungování mediánového filtru. Zdroj: [16, překresleno]

3.2 Binarizace

Proces binarizace hraje zásadní roli při předzpracování obrázku účtenky. Vstupem je obrázek ve stupních šedi a výstupem je pak binární obrázek. Tedy obrázek, kde je každý pixel buď černý nebo bílý. V ideálním případě by binarizovaný obrázek obsahoval pouze černý text a bílé pozadí. Špatná binarizace obrázku může vést ke snížení kvality do takové míry, že Tesseract na obrázku nerozpozná žádný text.

Neexistuje jeden správný postup binarizace. Vždy je potřeba zvolit takový postup, který se hodí pro vstupní data. Svědčí o tom třeba každoročně pořádaná soutěž v binarizaci [17], která je dobrým zdrojem porovnání kvality binarizačních metod. Dvě z nich jsou tu popsány a použity pro binarizaci obrázku účtenky.

3.2.1 Otsu

Jednoduchý a často používaný algoritmus binarizace je Otsův algoritmus [7]. Je to globální binarizační algoritmus. Pro vstupní obrázek najde jedno fixní číslo k, kterému se říká threshold. Následně hodnotu každého pixelu ve vstupním obrázku porovná s tímto thresholdem k. Pixely s menší hodnotou nastaví ve výstupním obrázku na černé, ostatní na bílé. Hodnota k se určí následovně.

Uvažujme vstupní obrázek g ve stupních šedi. Ten je tvořen pixely s intenzitou $g(x, y) \in [0, 255]$, kde (x, y) je pozice pixelu na obrázku g. Pixely výstupního obrázku o se vypočítají podle rovnice 3.1.

$$o(x,y) = \begin{cases} 0 & \text{pokud } g(x,y) \leq k \\ 255 & \text{jinak} \end{cases} \tag{3.1}$$

(a) grayscale (b) Sauvola (c) Otsu

Obrázek 3.2: Ukázka binarizace

Hodnota k se určí iterativně přes všechny možné hodnoty k, tedy v rozmezí $[0, 255]$. Různé hodnoty k rozdělí pixely vstupního obrázku podle jejich hodnoty na dvě skupiny. Pro každou skupinu se vypočítá rozptyl hodnot pixelů které obsahuje. Následně se vypočítá vážený součet těchto dvou rozptylů. Výsledné k má tuto hodnotu nejmenší možnou, viz rovnice 3.2.

$$k = \underset{m \in [0,255]}{\arg\min} \left(W_{C_1} \sigma^2_{C_1} + W_{C_2} \sigma^2_{C_2} \right) \tag{3.2}$$

kde

$C_1 = \{(x, y) \mid g(x, y) \leq m\}$, $C_2 = \{(x, y) \mid g(x, y) > m\}$

$W_{C_1} = \frac{|C_1|}{|C_1|+|C_2|}$, $W_{C_2} = \frac{|C_2|}{|C_1|+|C_2|}$

$\sigma^2_{C_1}$ je rozptyl hodnot pixelů v C_1

$\sigma^2_{C_2}$ je rozptyl hodnot pixelů v C_2

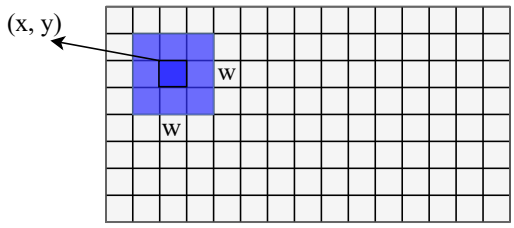
Otsův algoritmus v určité míře používá Tesseract při procesu rozpoznávání textu. Ukázka binarizace je na obrázku 3.2c, kde je vidět jeho hlavní nevýhoda. Díky tmavému osvětlení byla zvolena příliš velká hodnota k, která nedovolila segmentovat špatně osvětlenou část textu. Výhodou tohoto algoritmu je jeho rychlost a jednoduchá implementace.

3.2.2 Sauvola

Sauvolův binarizační algoritmus [18] je zástupce z lokálních binarizačních algoritmů. Stejně jako u Otsova algoritmu se používá pro určení hodnoty výsledného pixelu rovnice 3.1. Threshold se ale počítá pro každý pixel zvlášť v rámci jeho okolí. Okolí pixelu je definováno parametrem W a znamená čtverec $W \times W$ pixelů se středem ve zkoumaném pixelu, viz obrázek 3.3. Threshold pro pixel (x, y) se určí podle rovnice 3.3.

$$t(x, y) = m(x, y) \left[1 + K \left(\frac{s(x, y)}{R} - 1 \right) \right] \tag{3.3}$$

kde K je číslo z intervalu $[0, 2; 0, 5]$, R je maximální hodnota směrodatné odchylky, $m(x, y)$ je průměr okolních pixelů, $s(x, y)$ je směrodatná odchylka okolních pixelů. Ukázka binarizace je vidět na obrázku 3.2b.



Obrázek 3.3: Okolí pixelu (x, y) pro $W = 3$

Na rozdíl od Otsova algoritmu potřebuje Sauvola znát 3 parametry: W, K,R. Poslední R je pro obrázky ve stupních šedi rovno 128. Zbývá určit šířku mřížky W a číslo K. V práci [19] použili $W = 14, K = 0,34$ jako nejlepší volbu pro běžné dokumenty. V [20] zkoumali výkon OCR a došli k výsledku $W = 60, K = 0,4$. Sauvola použil v [18] $W = 15, K = 0,5$.

Složitost takto popsaného Sauvolova algoritmu je pro obrázek $N \times N$ pixelů rovna $\mathcal{O}(|W|\, N^2)$. Použitím integrálního obrazu lze ale složitost zbavit velikosti mřížky a dostat ji na $\mathcal{O}(N^2)$. Cenou je ale vyšší spotřeba operační paměti. Takto implementovaný Sauvolův algoritmus má podobné časové vlastnosti jako rychlejší Otsův algoritmus. [21]

Kapitola 4

Rotace účtenky

Tato kapitola popisuje dva způsoby, jak lze rotovat účtenku do správného úhlu. První způsob popsaný v sekci 4.1 je metoda lineární regrese. Ta používá několik kroků, nejzajímavější je však v této práci nově navržená dynamická volba strukturního elementu pomocí algoritmu K-means. Druhý způsob pro rotaci účtenky popsaný v sekci 4.2 je založen na Houghově transformaci.

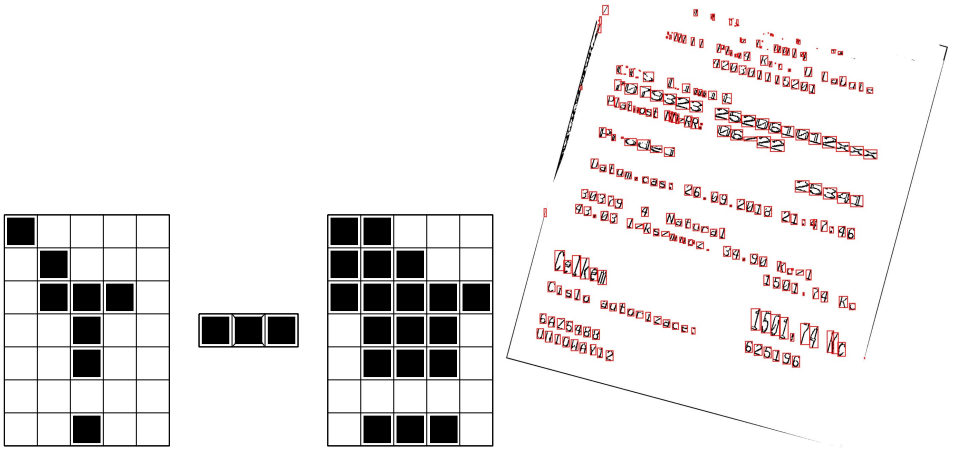
4.1 Metoda lineární regrese

Tato sekce popisuje jeden ze způsobů, jak lze určit sklon obrázku účtenky. Skládá se ze tří po sobě jdoucích kroků aplikovaných na předzpracovanou účtenku. Kroky jsou popořadě — morfologická dilatace, extrakce komponent souvislosti a regresní přímka, podle které se tato sekce jmenuje. Tento postup je popsán v [22]. V této práci je navržený nový způsob dynamické volby strukturního elementu.

4.1.1 Morfologická dilatace

Tato část je nejdůležitější v celém postupu a její špatné provedení má za následek mylné natočení účtenky. Nejdříve je zde popsána morfologická dilatace obecně, pak stručně její použití v práci [22] a nakonec nový způsob založený na algoritmu K-means.

Morfologickou dilataci si můžeme představit jako operaci nad binárním obrázkem reprezentovaným maticí $X \in \{0,1\}^{m,n}$ se strukturním elementem $B \in \{0,1\}^{k,l}$, kde $k \leq m, l \leq n$. Aplikací morfologické dilatace $X \oplus B$ „*přičteme*“ ke každému prvku matice X strukturní element B. Přičtením je při aplikaci na binární obrázek operace OR. Příklad aplikace morfologické dilatace je znázorněn na obrázku 4.1a. Na něm je vlevo binární obrázek X a uprostřed strukturní element B. Výsledek aplikace morfologické dilatace $X \oplus B$ je vpravo.



(a) aplikace strukturního elementu Zdroj: [23] (b) bounding boxy pro znak

Obrázek 4.1: Morfologická dilatace

Morfologickou dilataci použijeme na účtenku. Na obrázku 4.1b jsou červeně vidět bounding boxy okolo rozpoznaných znaků. Každý bounding box si můžeme představit jako prvek matice X nastavený na hodnotu 1. Cílem morfologické dilatace je rozšířit tyto bounding boxy tak, aby vytvořily souvislou komponentu, která reprezentuje co možná nejdelší řádku. Pak už bude jednoduché aproximovat přímku, která prochází danou řádkou a natočit obrázek účtenky o její úhel. Problematická je ale volba strukturního elementu B. Pokud se zvolí příliš malý, nedojde k vytvoření velkých souvislých komponent. Naopak pokud B zvolíme příliš velké, dojde ke spojení řádek a aproximace přímky bude mylná. Dále pokud strukturní element zvolíme jakýkoliv, ale bude fixní, musíme také počítat s velikostí mezer mezi řádky textu. Tam fixně zvolené B může propojit řádky, které jsou blízko u sebe. Pro fixní B je tedy nutná předchozí znalost vstupního textu, který navíc musí mít podobný formát. Toto ale dataset účtenek nesplňuje.

Autor tento problém v [22] řeší zavedením n různých strukturních elementů B_n. Pak pro vstupní obrázek X provede nezávisle n morfologických dilatací, tedy $Y_n = X \oplus B_n$. Výslednou souvislou komponentu (nejdelší řádku) získá tak, že najde nejdelší společnou podposloupnost společnou pro všechny Z_n. Toto řešení mu poskytuje na perfektním obrázku dobré výsledky, používá se ale pořád fixní volba strukturních elementu. V této práci se strukturní element pokusím odhadnout dynamicky.

4.1.2 Dynamická volba strukturního elementu

Tato část popisuje nový algoritmus pro dynamickou volbu strukturního elementu.

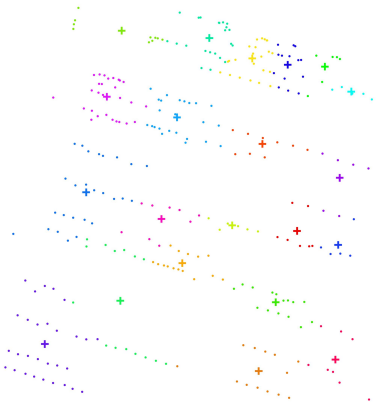
Algoritmus nejprve začíná s bounding boxy okolo znaků, viz obrázek 4.1b. Ty roztřídí do k clusterů, kde k je odhadnutý počet slov na obrázku účtenky. Cluster tedy ideálně reprezentuje jedno slovo a znaky v něm obsažené. V rámci každého clusteru je pak určen strukturní element B následovně: Pro každý cluster se spočítá průměrná vzdálenost bodů od jeho geometrického centra. Tato vzdálenost se označí $\tilde{x}$. Potom se různý násobek $\tilde{x}$ přičte k šířce a výšce bounding boxu. Takto upravený bounding box se nazve dilatovaný bounding box. K roztřídění bodů do clusterů se použije algoritmus K-means, který použije jako body středy bounding boxů. K-means je popsaný zde [24]. Počet slov na obrázku účtenky se v práci odhaduje pomocí nástroje Tesseract.

Tento postup má několik nedostatků. Prvním je správná volba počtu slov k. Pokud se počet slov odhadne špatně, K-means spojí do clusterů více řádek (malé k) nebo naopak jen jednotlivé znaky (velké k). Druhým je zvolení násobku $\tilde{x}$ tak, aby rozšířené bounding boxy nepřesáhly svojí řádku.

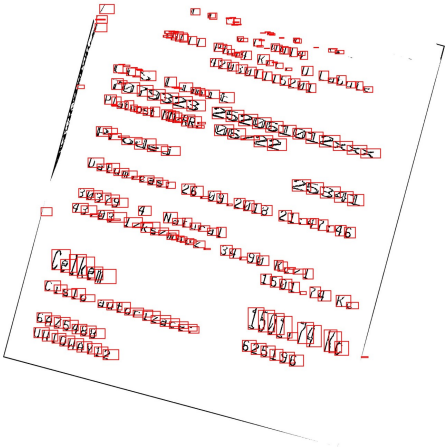
Kroky jsou graficky znázorněny na obrázku 4.2. Nejprve se středy nalezených bounding boxů algoritmem K-means roztřídí do clusterů. Ty jsou na obrázku 4.2a vyobrazeny barevně s křížkem, který značí geometrický střed clusteru. Pak se k šířce bounding boxu v rámci clusteru přičte $0,25 \cdot \tilde{x}$, k výšce bounding boxu se na tomto obrázku nic nepřičetlo. Výsledkem jsou dilatované bounding boxy, viz obrázek 4.2b.

4.1.3 Extrakce komponent souvislosti

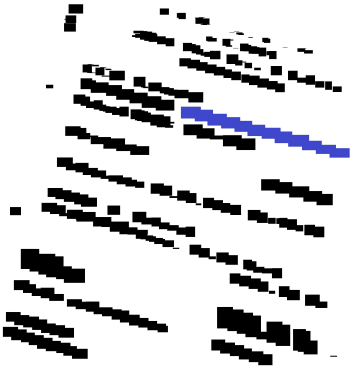
Z dilatovaných bounding boxů z předchozího kroku se vytvoří souvislé komponenty. Souvislá komponenta se skládá z bounding boxů, které dohromady tvoří jeden polygon. Zde jsou možné dva způsoby, jak toho docílit. První je algoritmus se složitostí $\mathcal{O}(n^2)$, kde n je počet bounding boxů. Ten pro každou dvojici bounding boxů zkontroluje, jestli se protínají. Pokud ano, spojí je. Vzhledem k tomu, že bounding boxy jsou obdélníky, které mají dvě strany rovnoběžné s osou x a dvě strany rovnoběžné s osou y, lze použít rychlejší algoritmus se složitostí $\mathcal{O}(n \log n)$. Ten je popsaný v práci [25]. Zrychlení ale bude znát pro větší hodnoty n, které se v případě obrázku účtenky pohybuje okolo 150. Pro obrázek účtenky jsou tedy použitelné oba algoritmy. Příklad souvislých komponent je na obrázku 4.2c.



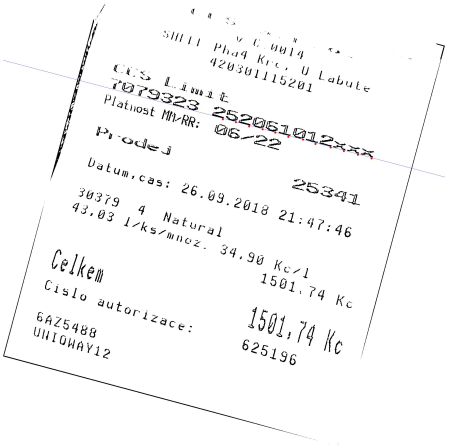
(a) K-means clustery



(b) dilatované bounding boxy



(c) souvislé komponenty



(d) regresní přímka

Obrázek 4.2: Kroky metody lineární regrese

4.1.4 Regresní přímka

Posledním krokem je odhad úhlu, o který je obrázek účtenky natočen. Nejdříve se nalezne souvislá komponenta s největším obsahem. Obsah souvislé komponenty se vypočítá jako obsah polygonu, který reprezentuje. Na obrázku 4.2c je taková komponenta zvýrazněna modrou barvou. Pak se z ní extrahují body následujícím způsobem: Z každého bounding boxu který tvoří největší souvislou komponentu je získán bod jako střed jeho dolní hrany. Ty jsou vybrány proto, že mají největší šanci tvořit přímku. Pokud bychom uvažovali například střed bounding boxu, kvůli rozdílné výšce znaků by jsme nikdy nedostali přesnou přímku.

Z extrahovaných bodů se aproximuje regresní přímka metodou nejmenších čtverců. Metoda je popsána zde [26]. Z přímky se určí její úhel a o ten je obrázek účtenky nakloněn do správné polohy. Příklad regresní přímky je vidět na obrázku 4.2d.

4.2 Houghova transformace

Tato sekce popisuje druhý způsob, jak lze určit sklon obrázku účtenky. Využívá k tomu Houghovu transformaci popsanou v [27, 28], z kterých tato sekce vychází.

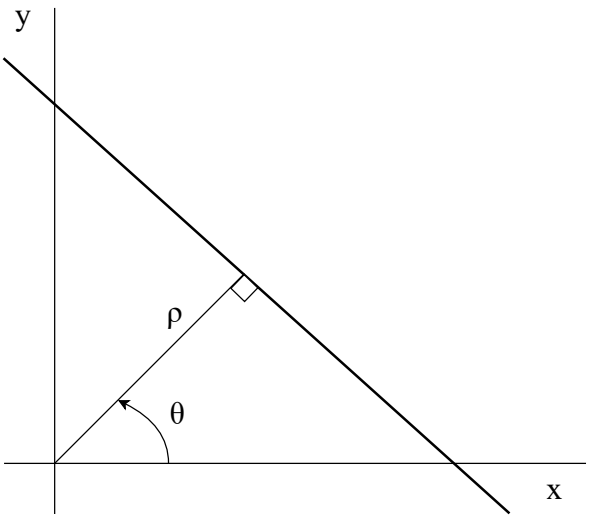
Houghova transformace je metoda, která bod (x, y) v prostoru obrázku převede na sinusoidu v prostoru parametrů (bod (x, y) si můžeme představit jako souřadnici bounding boxu z obrázku 4.1b). Převod se provede pomocí rovnice 4.1.

$$x \cos\theta + y \sin\theta \;=\; \rho \tag{4.1}$$

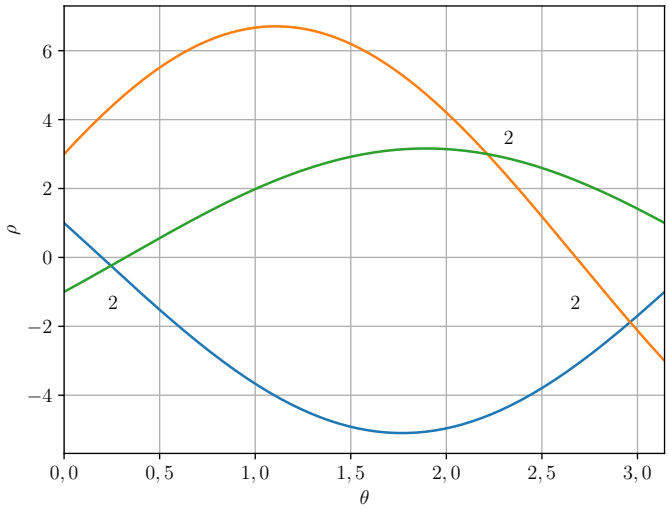
Parametry jsou úhel θ a číslo ρ. Každý bod v prostoru obrázku tedy odpovídá jedné sinusoidě v prostoru parametrů. Jinak si ještě můžeme parametry θ a ρ představit jako přímku procházející bodem (x, y) s úhlem θ a délkou její normály ρ, viz obrázek 4.3.

Pro více bodů v prostoru obrázku můžeme na každý z nich aplikovat Houghovu transformaci. Dostaneme několik sinusoid v prostoru parametrů. Když se nějaké dvě sinusoidy protnou v bodu p, souřadnice tohoto bodu reprezentují přímku procházející oběma body v prostoru obrázku. Pokud bodem p prochází ještě třetí sinusoida, tři body leží na jedné přímce v prostoru obrázku.

Takto popsaný model hledá pouze přímky, které přesně procházejí danými body a nedovoluje žádné odchylky. Tento nedostatek se dá vyřešit pomocí rozdělení grafu se sinusoidami do mřížky, viz obrázek 4.4. Teď každá část mřížky v prostoru parametrů reprezentuje jednu přímku v prostoru obrázku.



Obrázek 4.3: Prostor parametrů. Zdroj: [27, Fig. 1, překresleno]

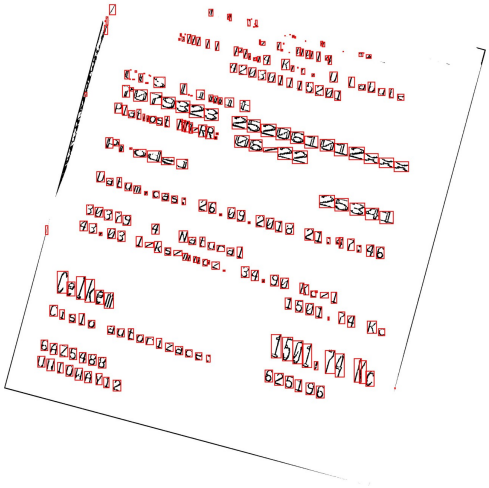


Obrázek 4.4: Houghova transformace s mřížkou

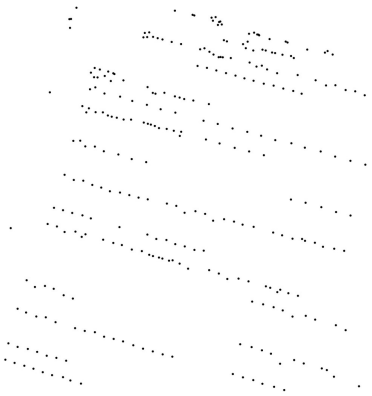
Pro určení správné rotace účtenky se použije transformace s mřížkou. Ta funguje následovně: Nejprve se naleznou bounding boxy v předzpracované účtence, viz obrázek 4.5a. Z každého bounding boxu vznikne bod jako střed jeho dolní hrany, viz obrázek 4.5b. Důvodem výběru středu dolní hrany je zarovnanost textu do řádek. Pokud by byl jako bod vybrán například střed bounding boxu, díky různé výšce znaků by výsledné body měly větší rozptyl. Poslední krok je nalezení části mřížky, kterou prochází nejvíce sinusoid. Na obrázku 4.4 jsou tři sinusoidy v prostoru parametrů, kdy každá z nich reprezentuje jeden bod v prostoru obrázku. Tři části mřížky mají skóre 2. Nejlepší přímky jsou tři, kdy každá prochází právě dvěma body v prostoru obrázku. Výsledkem Houghovy transformace je tedy množina přímek. Každá přímka má přiřazené skóre podle toho, kolik reprezentuje bodů. Na obrázku 4.5c jsou znázorněný tři přímky s nejvyšším skóre.

V tomto postupu je důležitá velikost a rozdělení mřížky. Ta se určuje úhlem θ a parametrem ρ. Úhel θ říká, pro jaké stupně budeme přímky hledat. Pokud například rozdělíme θ po 1°, uvažujeme všechny přímky, které se od sebe liší o 1°. Minimální hodnota θ je 0°, maximální hodnota 180°. Pro větší hodnoty by se přímky opakovaly. Hodnota ρ nabývá hodnot $[-u, u]$, kde u je délka úhlopříčky vstupního obrázku.

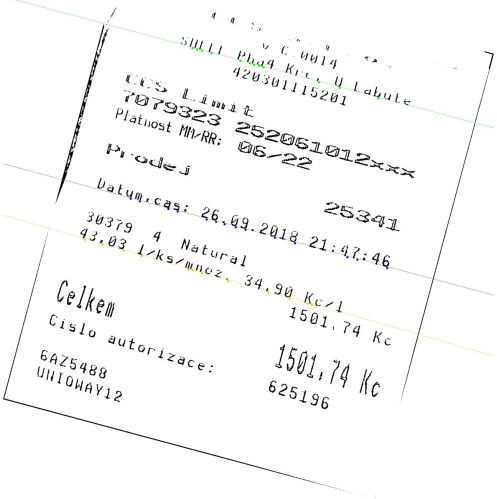
V této práci se parametr $\theta \in [0, 180]$ rozdělí po 1°. Parametr $\rho \in [-u, u]$ se rozdělí na `rho resolution`$\times 2\,u$ stejných hodnot. Rozumná hodnota pro `rho resolution` je například $0,9$.



(a) bounding boxy



(b) středy dolních hran bounding boxů



(c) Houghovy přímky

Obrázek 4.5: Houghova transformace

Kapitola 5

Extrakce faktů

Tato kapitola popisuje poslední krok celého postupu — nalezení ceny, času a datumu v rozpoznaném textu. Nejdříve jsou v sekci 5.1 popsány formáty ceny, času a datumu. V sekci 5.2 je popsán algoritmus pro extrakci faktů.

Nalezení ceny, času a datumu v textu není komplikované vzhledem ke specifickému datasetu. Před hledanými informacemi se totiž nachází vždy různá klíčová slova, podle kterých se v rozpoznaném textu může vyhledávat, viz definice 5.0.1. Nicméně rozpoznaný text nebude vždy ideální, takže se musí počítat s určitou chybou. Využívá se proto editační vzdálenosti popsané v definici 5.0.2.

Definice 5.0.1. *Klíčové slovo v textu je takové slovo, které bezprostředně předchází hledaný fakt nebo bezprostředně následuje za hledaným faktem.*

Definice 5.0.2. *Editační vzdálenost $d(u, v)$ mezi slovy u, v je minimální počet operací vkládání, mazání a substituce takových, aby po jejich provedení byly u,v totožné.*

5.1 Formáty

Úkolem je z účtenky vyčíst cenu, čas a datum. Tyto údaje mají napříč datasetem pevně daný formát. Navzdory 43 rozdílným společnostem vydávajícím tyto účtenky. Navíc se u těchto údajů pravidelně vyskytují klíčová slova předcházející a klíčová slova následující za daným údajem. Například za cenou se téměř vždy vyskytuje měna (*kč*) a před cenou se vyskytují slova *celkem*, *suma* atd. Tento formát je důležitý pro nalezení hledaných údajů v rozpoznaném textu.

Cena má ze všech tří nejmenší variaci ve formátu. Ve valné většině účtenek je cena číslo, které obsahuje buď jednu desetinnou tečku nebo čárku. V několika případech u společnosti SHELL obsahuje cena jak tečku, tak čárku. Cena se na účtence vyskytuje několikrát (celková cena, cena bez DPH, cena jednotlivých

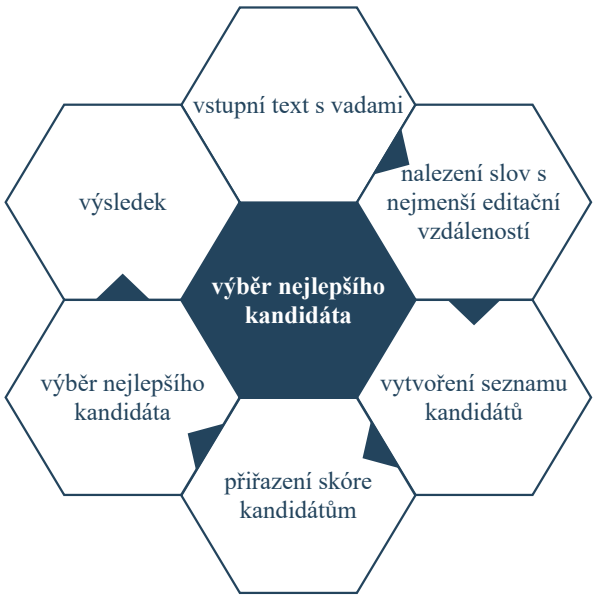
položek, mezisoučet, ...). Tabulka A.3 v příloze popisuje slova předcházející a následující za cenou společně s příklady formátu ceny.

Čas a **datum** mají podobný formát. Jsou to čísla, která obsahují dva oddělovače. U času je to pouze dvojtečka, v několika případech je pouze jedna. U datumu je oddělovačem tečka, spojovník, nebo lomítko. Oba údaje se vyskytují pohromadě. Na účtence se datum nachází většinou pouze jednou. Čas se na účtence nachází většinou dvakrát a je rozdílný (datum vystavení účtenky a datum zaplacení účtenky kreditní kartou). Pro čas i datum jsou společná klíčová slova, která se před nimi vyskytují. Klíčová slova vyskytující se za časem a datumem neexistují. Viz tabulka A.4 v příloze.

5.2 Postup

Klíčová slova jsou rozdělena podle **typu** a **kontextu**. Typy klíčového slova jsou `cena`, `čas` a `datum`. Kontext je `předcházející` nebo `následující`. Klíčové slovo může mít více typů ale jen jeden kontext. Například klíčové slovo *kč* je typu `cena` s kontextem `následující`. Klíčové slovo *datum* je typu `čas` a `datum` s kontextem `předcházející`.

Zjednodušený postup extrakce pro jeden typ klíčových slov je znázorněný na obrázku 5.1. Vstupem je text, který může obsahovat vady. V textu se naleznou slova, která mají nejmenší editační vzdálenost od klíčových slov. Pomocí těchto slov se vytvoří seznam kandidátů pro hledaný fakt. Kandidátům je podle jejich pořadí a podobnosti hledanému faktu přiřazeno skóre. Následně se vybere kandidát s největším skóre, který je vrácen jako hledaný fakt.



Obrázek 5.1: Výběr nejlepšího kandidáta

Algoritmus 5.1: Nejlepší kandidáti

```
1  TEXT := rozpoznany text na uctence
2
3  Pro KEYWORDS jako vsechna klicova slova postupne pro cenu, cas a datum:
4    D := prazdne pole vzdalenosti
5    C := prazdne pole kontextu klicovych slov
6    Pro WORD v TEXT:
7      D[WORD] := minimalni editacni vzdalenost mezi WORD
8                 a vsemi slovy z KEYWORDS
9      C[WORD] := kontext klicoveho slova,
10                od ktereho ma WORD minimalni editacni vzdalenost
11     MIN = nejemensi editacni vzdalenost v D
12
13     CAND := prazdny seznam kandidatu
14     P := prazdne pole se skorem kandidatu
15     Pro vsechny WORD s D[WORD] == MIN:
16       LIST := prazdny list
17       Pokud C[WORS] == "predchazejici":
18         LIST = poporade pet slov v TEXT nasledujicich slovo WORD
19       Pokud C[WORS] == "nasledujici":
20         LIST = poporade pet slov v TEXT predchzejicich slovo WORD
21       Pro vsechny slova X v LIST, ktere nejsou v CAND:
22         pridej X do CAND
23         P[X] = PORADI(X) * PODOBNOST(X)
24
25   BEST := takove X z CAND, ktere ma nejvetsi P[X]
26   Oznac BEST jako nejlepsiho kandidata pro dany typ klicovych slov
```

Algoritmus 5.1 má na vstupu `TEXT`, který představuje rozpoznaný text z účtenky. `TEXT` může obsahovat vady, například vynechané nebo změněné znaky. Algoritmus pak postupně projde (3-26) všechny typy klíčových slov. V `TEXT` najde slova, která mají nejmenší editační vzdálenost od klíčových slov (6-10). Následně pokud je takové slovo kontextu `předcházející`, do seznamu kandidátů přidá 5 následujících slov (17-18). Pokud je slovo kontextu `následující`, do seznamu kandidátů přidá 5 předcházejících slov (19-20). Pak každému kandidátovi přiřadí skóre (23). Kandidát s největším skóre se označí za hledaný fakt (25-26).

Skóre `P[X]` kandidáta `X` je součinem dvou čísel. První je `PORADI`. To se určí podle rovnice 5.1. Pro pět kandidátů budou hodnoty popořadě $1,79$; $1,60$; $1,38$; $1,09$; $0,69$. Tedy bližší kandidáti mají tuto hodnotu vyšší. Druhým je `PODOBNOST`. Ta nabývá hodnot z intervalu $[0;1]$ a vyjadřuje podobnost slova hledanému faktu. Například podobnost k času u slova *15:41:29* je 1, ale u slova *prodej* je podobnost 0. Výpočet `PODOBNOST` je implementován pomocí regulárních výrazů. Konkrétní implementace je ve zdrojových kódech v přiloženém médiu v třídě `DataPointExtractorTool`.

$$\mathcal{R}(X) = \ln(k - i + 2) \tag{5.1}$$

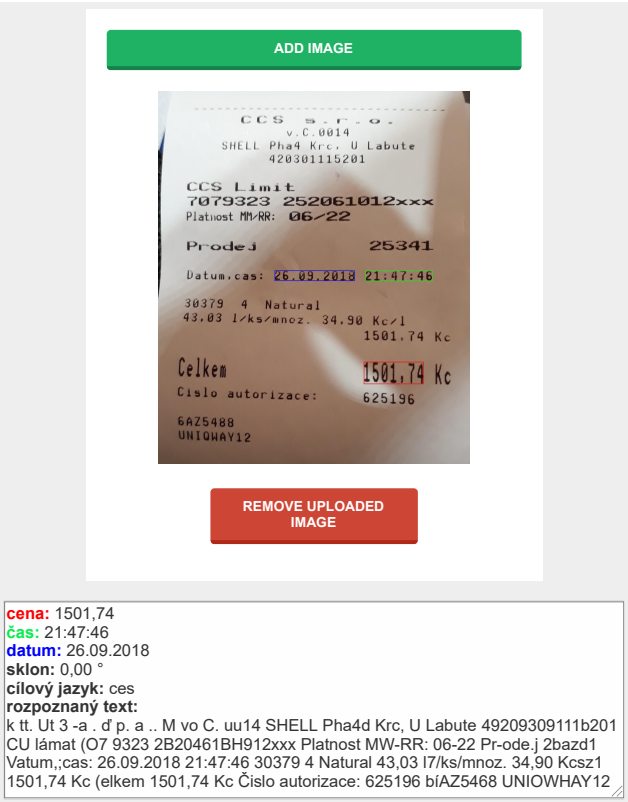
kde

X je označený kandidát
k je počet označených kandidátů
i je pozice označeného kandidáta

Kapitola 6

Implementace webové aplikace

Tato kapitola popisuje implementaci webové aplikace a její strukturu — business a prezentační vrstvu. Dále je zde popsán postup nasazení webové aplikace na kterémkoliv počítači podporující nástroj Docker. Aplikace umožňuje uživateli nahrát vybraný obrázek tištěné účtenky a obdržet uvedenou cenu, čas a datum, viz obrázek 6.1, který zachycuje snímek z webové aplikace.

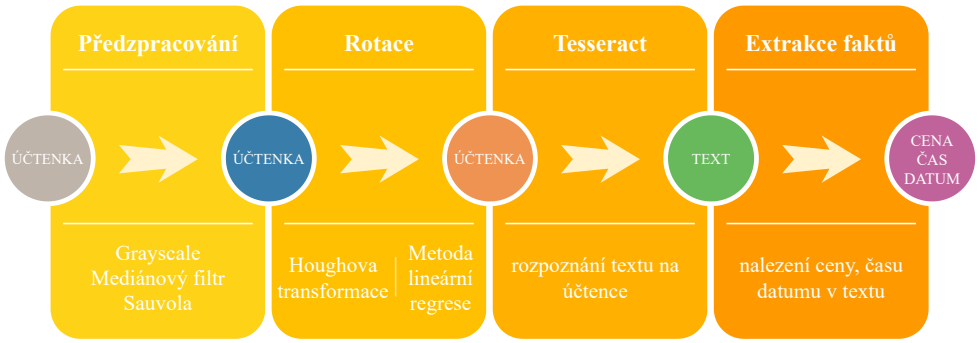


Obrázek 6.1: Ukázka webové aplikace

6.1 Architektura

Celá aplikace je koncipovaná jako dvouvrstvá. Skládá se z prezentační a business vrstvy. Tyto dvě části jsou navzájem oddělené a komunikují mezi sebou prostřednictvím definovaného rozhraní. Lze tedy nezávisle kteroukoliv z nich vyměnit/upravit. Prezentační vrstva je popsaná v sekci 6.1.1, business vrstva je popsaná v sekci 6.1.2.

Workflow aplikace je na obrázku 6.2. Znázorňuje celý postup tak, jak je aplikován na účtenku. Skládá se ze čtyř kroků — předzpracování, rotace, Tesseract, extrakce faktů. Do **předzpracování** je vstupem účtenka od uživatele společně s metadaty (DPI, typ, jazyk, ...). Ta se převede do odstínů šedi. Pokud je povolený, použije se mediánový filtr s parametrem `median k`. Pak se, pokud je povolený, použije Sauvolův binarizační algoritmus s parametry K, W. Výstupem je předzpracovaná účtenka. Ta je vstupem do kroku **rotace**. V té se, pokud je rotace povolená, použije buď Houghova transformace s parametrem `rho resolution`, nebo metoda lineární regrese s parametry `K-means iter` a `width scale`. Výstupem je předzpracovaná a orotovaná účtenka. Ta se použije jako vstup do další části. V té se pomocí nástroje **Tesseract** rozpozná text. Parametry pro Tesseract jsou jazyk rozpoznávání a DPI vstupního obrázku. Rozpoznaný text je vstupem do poslední části. V té se pomocí algoritmu pro **extrakci faktů** ve vstupním textu nalezne cena, čas a datum.



Obrázek 6.2: Workflow aplikace

6.1.1 Prezentační vrstva

Prezentační vrstva má za úkol načíst vybranou účtenku od uživatele a zobrazit výsledek rozpoznání. Před jejím posláním na rozpoznání do business vrstvy probíhají dvě transformace vstupního obrázku.

První z nich je správná rotace obrázku. Ta nemá ale nic společného s rotací popisovanou v předchozích kapitolách. Některé kamery totiž ukládají obrázek vždy „*naležato*“ a informaci o jeho správném natočení uloží do EXIF hlavičky. Důvodem je výpočetní náročnost rotace pořízeného obrázku. Prezentační vrstva EXIF hlavičku pokaždé přečte. Pokud je potřeba pixely rotovat, udělá to. Tento krok je výpočetně náročný.

Druhá transformace je zakódování obrázku pomocí BASE64. Kódování BASE64 je bezztrátové a zvýší velikost obrázku zhruba o 33 %. Důvodem je bezproblémový přenos a interpretace dat zakódovaných pomocí BASE64.

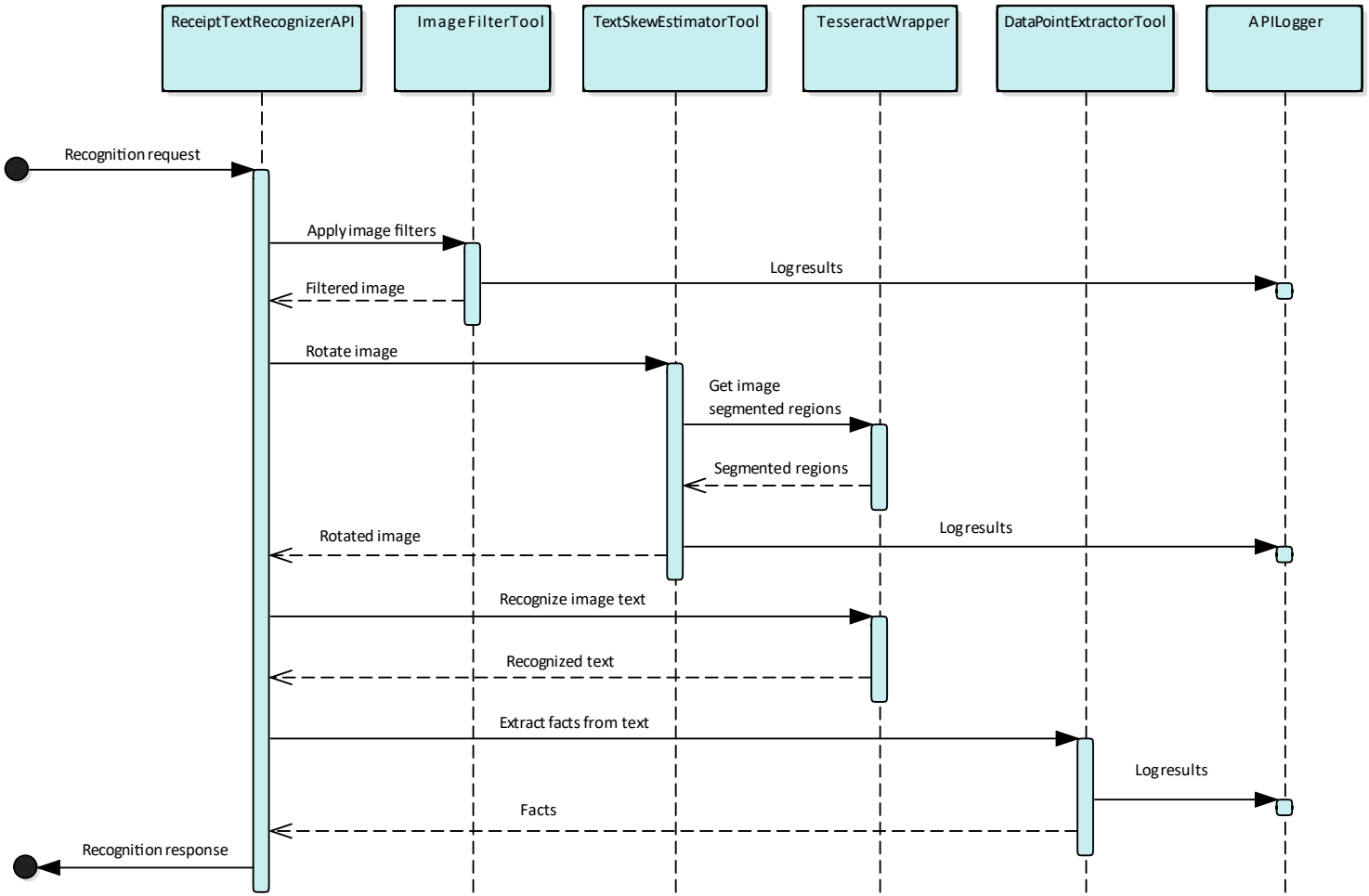
Po těchto dvou transformacích je obrázek odeslán na analýzu do business vrstvy. Společně s obrázkem se také posílají metadata: DPI, požadovaný jazyk pro rozpoznání, název a typ obrázku.

Výsledek rozpoznání je tří typů. První je úspěšný. Na obrázku se podařilo najít a rozpoznat text. V takovém případě aplikace zobrazí uživateli správně natočený obrázek a úhel, o který byl natočen. Barevně pak vyznačí cenu, čas a datum, pokud byly v textu nalezeny. Uživateli je také prezentován celý rozpoznaný text a cílový jazyk. Po poklepání na obrázek si ho může zvětšit. Tento stav je zachycen na obrázku 6.1. Druhý typ je nenalezení textu. Tento výsledek znamená, že business vrstva úspěšně zpracovala obrázek, ale nenašla v něm žádný text. Poslední typ je vnitřní chyba, která signalizuje chybu při zpracování obrázku v business vrstvě.

6.1.2 Business vrstva

Business vrstva je základ celé webové aplikace. V této vrstvě jsou implementovány popisované algoritmy a postupy. Princip fungování business vrstvy je znázorněn na sekvenčním digramu na obrázku 6.3. Zpracování požadavku na rozpoznání začíná u `ReceiptTextRecognizerAPI`, které postupně aplikuje na vstupní obrázek jednotlivé kroky. Každá aplikace kroku je komunikována s `APILogger`, který výsledky daného kroku perzistentně uloží. Je zde znázorněn i `TesseractWrapper`, který je využíván celkem dvakrát. Poprvé je použit pro rychlou segmentaci obrázku, podruhé je použit pro rozpoznání textu na předzpracovaném a orotvaném obrázku.

`ImageFilterTool` odpovídá předzpracování účtenky. `TextSkewEstimatorTool` odpovídá rotaci účtenky. `DataPointExtractorTool` odpovídá extrakci faktů. `TesseractWrapper` odpovídá použití nástroje Tessract.



Obrázek 6.3: Zjednodušená architektura business vrstvy

6.2 Nasazení webové aplikace

Nasazení webové aplikace probíhá pomocí nástroje Docker [29]. Docker je terminálová aplikace dostupná pro většinu operačních systémů. Princip fungování nástroje Docker je takový, že tvůrce aplikace vytvoří Docker Image. To je virtuální operační systém, kam tvůrce nahraje zdrojové kódy, nainstaluje potřebný software a provede nastavení. Tento Docker Image potom jde jednoduše distribuovat a spustit kdekoliv pomocí nástroje Docker bez nutnosti dalšího nastavování a konfigurace na hostitelském počítači.

Spuštění a stažení aplikace se provede příkazem ve výpisu 4. Tento příkaz se pokusí najít Docker Image s názvem `slarty/tessapi` na lokálním počítači. Pokud ho nenajde, stáhne ho ze serveru DockerHub[2]. Potom použije port 8080 v Docker Image a prováže ho s portem 8080 na hostitelském počítači. Nakonec aplikaci spustí na pozadí.

```
$ docker run -p 8080:8080 -d slarty/tessapi
```

Výpis 4: Stažení a spuštění aplikace z DockerHub

Způsob bez stahování aplikace z DockerHub je s použitím přiloženého souboru `tessapi.tar`. Ten obsahuje zabalený Docker Image. Ke spuštění aplikace se v tomto případě použijí příkazy ve výpisu 5.

```
$ docker import tessapi.tar tessapi-imported
$ docker run -p 8080:8080 -d tessapi-imported
        "/opt/tomcat/bin/catalina.sh" "run"
```

Výpis 5: Spuštění aplikace z přiloženého souboru

Po spuštění a úspěšném dokončení příkazu je aplikace dostupná na této[3] adrese. Aplikace byla testována s prohlížečem Firefox [30] a nástrojem Docker verze [29] na testovacím datasetu. Parametry jsou přednastaveny na hodnoty shodné s `testCase1` popsané v příloze A. Bližší informace jsou na přiloženém médiu.

[2] `https://hub.docker.com/`
[3] `http://localhost:8080/receipt_recognition_frontend/`

Kapitola 7

Testování

Tato kapitola popisuje výsledky testů a jednotlivých částí postupu společně s metodikou testování. Výsledky jsou zde porovnány s ostatními autory.

Sekce 7.1 stručně popisuje použité testy a pojmy použité k interpretaci výsledků. Sekce 7.2 popisuje přesnost celého řešení. Diskutuje rozdílné výsledky pro cenu, čas a datum. Tyto výsledky pak srovnává s Google Vision a jinými autory. Sekce 7.3 zkoumá vliv kroku předzpracování. Sekce 7.4 popisuje výsledky dvou algoritmů pro rotaci.

Popis parametrů jednotlivých testů a klíčových slov použitých pro extrakci faktů se nachází v příloze A. V příloze B se nachází grafy, které jsou výstupem testování a nevešly se do této kapitoly.

7.1 Metoda vyhodnocování testů

Testování je rozděleno na tři sady testů. V první sadě se testuje celý algoritmus tak, jak je popsaný v předchozích kapitolách. Tedy předzpracování účtenky, rotace, rozpoznání textu pomocí nástroje Tesseract a extrakce faktů. Výsledky této části jsou rozděleny do osmi testů nazvaných `testCase1-8`. Každý `testCase` má jiné parametry, například různé hodnoty `K, W, R` u Sauvolova algoritmu. V druhé sadě se testují dva algoritmy pro rotaci účtenky. Tedy rotace účtenky pomocí metody lineární regrese a Houghovy transformace. Tyto testy jsou čtyři a jsou nazvány `rotationTest1-4`. V poslední sadě se testuje Google Vision.

Vyhodnocování testů `testCase` a Google Vision probíhalo na stejném datasetu 233 účtenek. Testování `rotationTest` probíhalo na datasetu účtenek bez rotace. Oba datasety jsou blíže popsány v kapitole 2.

Před každým testem byly nastaveny parametry algoritmů popsané v příloze A. U testu s Google Vision byl jediným nastaveným parametrem typ požadavku `TEXT_DETECTION`. Výsledky testů byly uloženy pro pozdější zpracování.

7.1.1 Pojmy

Na účtence **nebyl rozpoznán žádný text** pokud OCR systém žádný text nenašel nebo OCR systém vrátil chybu při zpracování.

Správné označení bounding boxu s údajem znamená, že algoritmus vytvořil bounding box buď pro cenu, čas nebo datum a tento vytvořený bounding box se protínal se skutečným bounding boxem, kde se údaj na účtence vyskytuje.

Přesně rozpoznaný údaj znamená, že algoritmus nejen pro daný údaj správně označil bounding box, ale také správně určil text, který se v daném bounding boxu má nacházet. Tato metrika je velice přísná, protože stačí jeden špatně rozpoznaný znak a údaj není klasifikován jako přesně rozpoznaný.

Editační vzdálenost rozpoznaného údaje od správné hodnoty se vyhodnocovala pouze u správně označených bounding boxů. Pro každý takový bounding box se vypočetla editační vzdálenost mezi jeho rozpoznanou a skutečnou hodnotou. Například na obrázku účtenky se podařilo správně označit bounding box s cenou a rozpoznat v něm text „*10C9.50*“. Pak se vypočetla editační vzdálenost mezi „*10C9.50*“ a skutečnou hodnotou „*1009.50*“, která je 1.

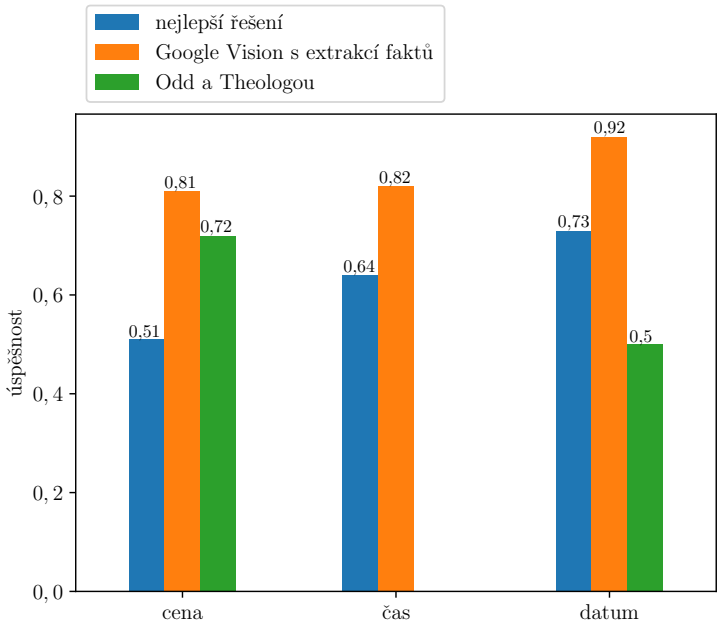
7.2 Výsledky navrženého postupu

Výsledky tří systémů pro extrakci ceny, času a datumu z účtenky jsou vidět na grafu 7.1a. První z nich je *nejlepší řešení*. To odpovídá zde v práci použitému `testCase8`, které dosáhlo nejlepších výsledků ze všech `testCase`. *Google vision s extrakcí faktů* prezentuje výsledky použití komerčního OCR nástroje Google Vision [11] bez jakéhokoliv předzpracování a rotace. Použita byla pouze extrakce faktů. Poslední *Odd a Theologou* představuje výsledky z práce [31], kde testovali nalezení ceny a datumu. K testování použili 556 účtenek ve švédském jazyce, které sami nafotili v kontrolovaném prostředí.

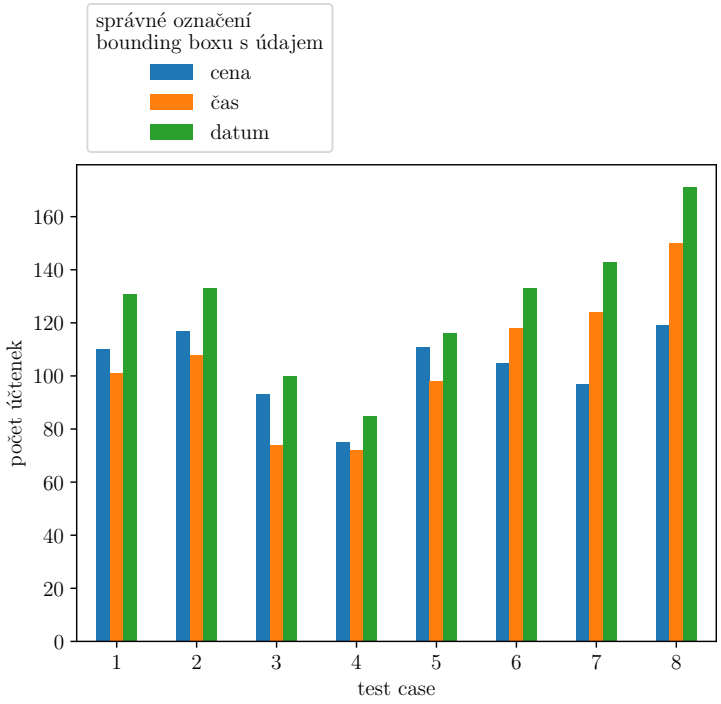
U **ceny** dosáhlo *nejlepší řešení* nejhoršího výsledku. Je to tím, že extrakce faktů hledá v rozpoznaném textu klíčová slova. Čím více se v textu nachází špatně rozpoznané znaky, tím více bude hledání klíčových slov zkreslené. V případě ceny, která má nejkratší klíčová slova, například „*kč*“, může dojít ke zkreslení velice rychle. U *Google Vision* jsou výsledky ceny a času konzistentní, protože docházelo k malému počtu špatně rozpoznaných znaků.

U **času** jsou výsledky *nejlepší řešení* lepší než v porovnání s cenou. Důvodem je specifický formát času a delší klíčová slova pro čas.

Nejlepších výsledků dosáhlo *Google Vision* a *nejlepší řešení* pro **datum**. Je tomu tak díky velice specifickému formátu datumu, kterému se na účtence nic nepodobá. Také se datum na účtence nachází převážně jednou.



(a) porovnání různých řešení



(b) výsledky navrženého postupu

Obrázek 7.1: Správné označení bounding boxu

7.3 Výsledky předzpracování

Tato sekce popisuje dva kroky předzpracování a jejich vliv na celkové rozpoznání.

Je ale zajímavé, že nejlepších výsledků dosáhl `testCase8`, který nepoužíval Sauvolovu binarizaci, mediánový filtr a rotaci. Jednalo se pouze o použití nástroje Tesseract pro rozpoznání textu a extrakce faktů. Důvodem je rozmanitost datasetu, kde je obtížné nalézt parametry Sauvolovy binarizace, vhodné pro všechny účtenky.

7.3.1 Sauvolova binarizace

První tři testy `testCase1-3` se lišily parametry pro Sauvolovu binarizaci. Nepoužívali mediánový filtr a k rotaci použili Houghovu transformaci. Jako parametry Sauvolova algoritmu byly zvoleny v literatuře tři nejčastější kombinace, viz podsekce 3.2.2. Z těchto tří testů byl nejlepší `testCase2`, viz graf 7.1b. Jeho parametry jsou $K = 0,4; W = 60$. Tyto parametry byly původně navržené v práci [20], kde zkoumali výkon OCR a došli právě k těmto parametrům. Tyto testy výsledek autorů potvrdily.

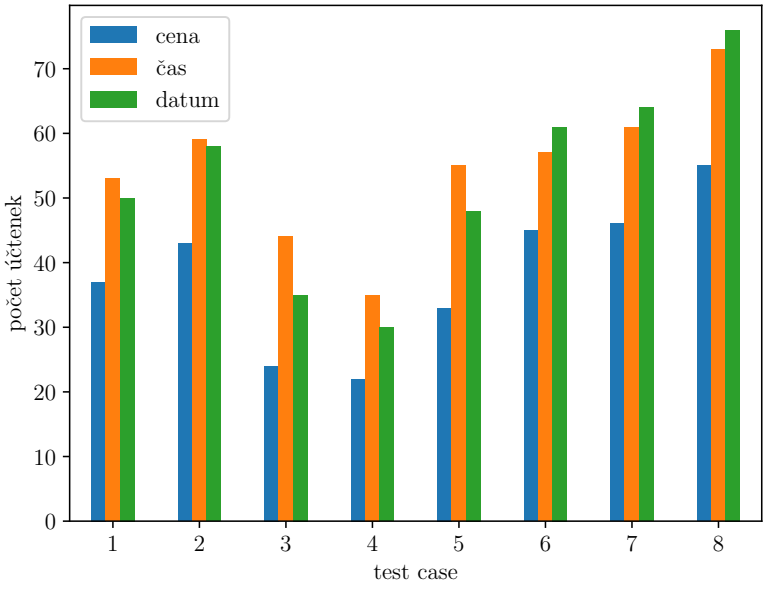
Pokud jde o dopady použití Saovolovy binarizace jako takové, můžeme porovnat testy `testCase6-8`, které nepoužívaly Sauvolovu binarizaci a testy `testCase1-5`, které používaly Sauvolovu binarizaci. Z porovnání testů na grafu 7.1b plyne, že nejlepší výsledek s binarizací je v porovnání s testy bez binarizace průměrný. Důvodem neúspěchu binarizace může být rozmanitý dataset. Sauvolova binarizace pro určitou skupinu účtenek poskytuje dobré výsledky, ale pro jinou už ne. Příklad je na obrázku 7.3, kde se změnou parametrů K, W mění kvalita segmentovaného textu.

7.3.2 Mediánový filtr

Zajímavé je pozorování dopadu mediánového filtru. U `testCase2` se nepoužil mediánový filtr, u `testCase5` se použil mediánový filtr. Jinak se oba testy nelišily. Oba použily Sauvolovu binarizaci se stejnými parametry a rotaci pomocí Houghovy transformace. Ve správném označení bounding boxu se `testCase2` bez mediánového filtru projevil jako lepší, měl v průměru o 10 účtenek lepší skóre než `testCase5` s mediánovým filtrem, viz graf 7.1b. Když se podíváme na přesně rozpoznané údaje, viz graf 7.2, `testCase2` měl také lepší výsledky. Mediánový filtr tedy v kombinaci s binarizací není vhodný. Rozdíl ale není velký.

Další dvojicí je `testCase6`, který použil mediánový filtr a `testCase7`, který nepoužil mediánový filtr. Oba testy nepoužívají binarizaci a používají rotaci pomocí Houghovy transformace. Ve správném označení bounding boxu se ukázal `testCase6` s mediánovým filtrem jako mírně lepší, viz graf 7.1b. Pokud se podíváme na graf 7.2, rozdíl je minimální. Mediánový filtr tedy bez použití

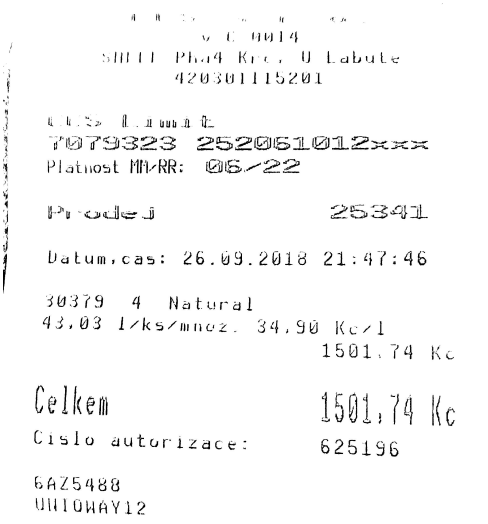
Sauvolovy binarizace mírně zlepší nalezení daného údaje. Nezlepší ale jeho správné rozpoznání.



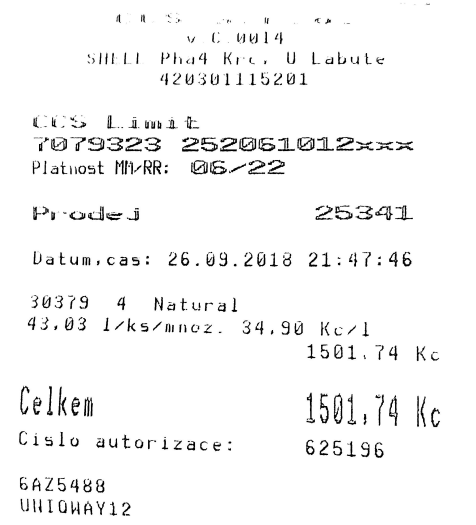
Obrázek 7.2: TestCase – přesně rozpoznané údaje

(a) vstup

(b) $K = 0,4; W = 60$



(c) $K = 0,5; W = 15$



(d) $K = 0,34; W = 14$

Obrázek 7.3: Sauvolova binarizace pro různé parametry

7.4 Výsledky rotace

Tato sekce shrnuje a porovnává výsledky dvou použitých algoritmů pro rotaci účtenky. Tedy Houghovu transformaci a metodu lineární regrese. V podsekci 7.4.1 jsou blíže popsány výsledky Houghovy transformace, v podsekci 7.4.2 jsou blíže popsány výsledky metody lineární regrese.

Pro testování rotace byly použity 4 testy nazvané `rotationTest1-4`. Použit byl dataset účtenek bez rotace, tedy 109 účtenek. Každý vstupní obrázek byl postupně rotován o předem danou hodnotu a pomocí algoritmu pro rotaci účtenky byl odhadnut jeho sklon. Pro měření úspěšnosti se použily chyby popsané v rovnicích 7.1 a 7.2.

Na grafu 7.4 jsou vidět nejlepší výsledky pro Houghovu transformaci a metodu lineární regrese. Pro malý stupeň natočení je lepší metoda lineární regrese, rozdíl ale není zásadní. Pokud se ale podíváme na naměřené hodnoty MAE, nejsou tak rozdílné oproti stupni natočení. To fakticky znamená, že oba algoritmy pro rotaci účtenky nedosahují dobrých výsledků. Za dobré výsledky by se daly považovat hodnoty MAE pod 1, kterých například dosáhl autor v práci [22]. V té použil metodu lineární regrese s jinou morfologickou dilatací. Pro testování používal jeden řádek textu bez jakýchkoli vad.

$$RMSE = \sqrt{\frac{1}{R}\sum_{k=1}^{R}(\beta_{ref} - \beta_{est}^{k})^{2}} \tag{7.1}$$

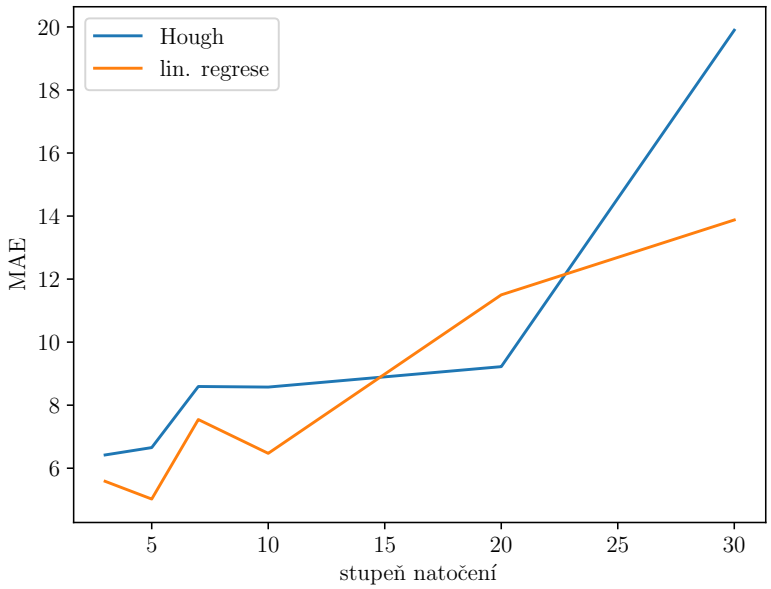
$$MAE = \frac{\sum_{k=1}^{R}|\beta_{ref} - \beta_{est}^{k}|}{R} \tag{7.2}$$

kde

R je počet účtenek v datasetu

β_{ref} je referenční úhel

β_{est}^{k} je odhadnutý úhel pro k-tou účtenku



Obrázek 7.4: Porovnání nejlepších výsledků rotace

(a) Houghova transformace: špatné přímky (b) metoda lineární regrese: spojení více souvislých komponent

Obrázek 7.5: Časté problémy rotace

7.4.1 Houghova transformace

Výsledky Houghovy transformace byly testovány v `rotationTest1-2`. Testy se liší použitím Sauvolovy binarizace. `rotationTest1` použil Sauvolovu binarizaci s parametry $K = 0,34; W = 14$. `rotationTest2` Sauvolovu binarizaci nepoužil.

Pokud se podíváme na vliv Sauvolovy binarizace, z grafu B.3b vidíme, že s použitím binarizace byla hodnota MAE u `rotationTest1` nižší, než u druhého `rotationTest2` zhruba o 10. Z toho plyne, že binarizace pomáhá při Houghově transformaci. Důvodem je, že binarizace segmentuje strukturu účtenky, tedy slova a řádky. Čitelnost výsledného textu při rotaci nehraje roli.

Pokud ještě porovnáme na grafech B.4 a B.5 hodnoty RMSE a MAE pro dobré a špatné účtenky, potvrzuje se opět, že použití Sauvolovy binarizace snížilo obě chyby. Navíc `rotationTest1`, který použil Sauvolovu binarizaci, měl lepší skóre pro dobré účtenky než pro špatné. To neplatí o `rotationTest2`, který Sauvolovu binarizaci nepoužil. Z toho plyne, že Houghova transformace s binarizací může benefitovat z lepší kvality účtenek. Naopak Houghova transformace bez binarizace nebenefituje z kvality účtenek a závisí spíše na použitém OCR nástroji.

Nicméně pro všechny zde zmíněné grafy jsou hodnoty RMSE a MAE špatné. Důvodem je výsledek rotace účtenky převážně dvěma způsoby. První z nich je správné natočení účtenky. Při něm jsou hodnoty RMSE a MAE malé. Druhý je natočení účtenky „*naležato*“, kde jsou hodnoty RMSE a MAE velké. Příklad je vidět na obrázku 7.5a, kde jsou vidět tři přímky vytvořené Houghovou transformací. Dvě přímky vedou k rotaci účtenky „*naležato*“ a jedna z nich vede ke správné rotaci účtenky.

Houghova transformace tedy lépe pracuje s použitím Sauvolovy binarizace. Kvalita účtenky hraje roli pouze při použití Sauvolovy binarizace. Bez jejího použití kvalita závisí na použitém OCR nástroji. Hlavním problémem Houghovy transformace je výběr správné přímky.

7.4.2 Metoda lineární regrese

Výsledky metody lineární regrese byly testovány v `rotationTest3-4`. Testy se liší použitím Sauvolovy binarizace. `rotationTest3` použil Sauvolovu binarizaci s parametry $K = 0,34; W = 14$. `rotationTest4` Sauvolovu binarizaci nepoužil.

Pokud se podíváme na vliv Sauvolovy binarizace, z grafu B.3b vidíme, že podobně jako u Houghovy transformace byly lepší výsledky s použitím binarizace. Důvodem je shodně lepší segmentace účtenky při použití binarizace.

Porovnání výsledků na špatných a dobrých účtenkách se u `rotationTest3`, který použil Sauvolovu binarizaci, výrazně neliší. Zajímavé jsou ale výsledky `rotationTest4`, který Sauvolovu binarizace nepoužil. Ten měl výrazně lepší výsledky pro špatné účtenky, viz grafy B.4 a B.5. Důvodem může být, že

zhoršená kvalita účtenek vede k horší segmentaci OCR nástrojem. Tím pádem je menší pravděpodobnost, že se spojí dva řádky při vytváření souvislých komponent a odhad přímky bude mylný.

Podobně jako u Houghovy transformace celkové výsledky nejsou dobré. Důvodem je v předchozím odstavci zmíněné spojování více řádku. To je znázorněno na obrázku 7.5b. Zde je modře znázorněna nejdelší souvislá komponenta. Aproximace přímky je v tomto případě náhodná. Důvodem tohoto spojení byla morfologická dilatace, která přeškálovala nalezené bounding boxy. Řešením tohoto problému by byla analýza šířky mezer mezi řádky a nastavení škálovacího parametru $\tilde{x}$ tak, aby ji nepřesáhl. Tato analýza je znemožněna různorodým datasetem, kde mají účtenky různou šířku mezi řádky.

Metoda lineární regrese tedy stejně jako u Houghovy transformace pracuje lépe s použitím Sauvolovy binarizace. Kvalita účtenky s použitím Sauvolovy binarizace nehraje velkou roli. Naopak pokud se Sauvolova binarizace nepoužije, tak metoda lineární regrese poskytuje výrazně lepší výsledky pro špatné účtenky.

Kapitola 8

Návrhy na zlepšení a rozšíření

Tato kapitola popisuje možné zlepšení kroků postupu. Také jsou zde návrhy na jeho rozšíření. Kapitola je logicky členěná podle pořadí kroků, které jsou aplikovány na účtenku — předzpracování, rotace, rozpoznání znaků a extrakce faktů. Poslední sekce obsahuje návrhy na zlepšení implementační části webové aplikace.

8.1 Předzpracování

První část předzpracování je aplikace mediánového filtru. Ta se používá kvůli odstranění šumu. Existují ale i další metody, které zde nebyly otestovány. Například by bylo zajímavé zkoumat použití Gaussova filtru nebo klouzavého průměru.

Následuje binarizace pomocí Sauvolova algoritmu. Ten byl testován pro tři sady parametrů K, W, které se v literatuře osvědčily jako nejlepší. Pro specifický dataset účtenek by se mohly otestovat i další sady nebo zkoumat použití jiného binarizačního algoritmu, který by dokázal lépe oddělit text od pozadí.

Na některých účtenkách byly znaky slabé a poškozené. Jako další část předzpracování by se mohl implementovat postup jejich opravy, popsaný v práci [32]. Tento postup by také bylo možné použít pouze ke zlepšení přesnosti rozpoznání. Stává se totiž, že se hledaný fakt podaří správně na účtence najít, ale rozpoznání přesných znaků se už nepodaří. Oprava znaků by mohla být využita právě na vyříznutý údaj, kde je přesné rozpoznání znaků nejdůležitější.

Technika vyřezávání části účtenky by mohla být dále rozvíjena. Místo celé účtenky by se v celém postupu mohlo pracovat například s řádky.

8.2 Rotace

Oba popsané postupy pro rotaci účtenky neposkytly dobré výsledky a mohou být zlepšeny.

Výsledek Houghovy transformace je množina přímek. Z té můžeme vyřadit ty špatné například podle znalosti, že účtenka má větší délku než šířku. Můžeme tedy rovnou vyřadit ty přímky, jejichž délka přesahuje větší stranu. Například na obrázku 7.5a by byly vyřazeny dvě špatné přímky a zbyla by pouze ta správná. Dále by bylo možné zavést formu hlasování. Vybralo by se n přímek, které mají nejvíce bodů. Následně by se přímky rozdělily na skupin podle úhlů, které odhadují. Vyhrála by nejpočetnější skupina. Tento postup by ale například na zmíněném obrázku 7.5a nefungoval. Rozdělil by přímky do dvou skupin, kde vítězná by obsahovala obě špatné přímky.

U metody lineární regrese jakožto nově navržené metody by bylo vhodné ji více otestovat. Například ladění škálovacího parametru $\tilde{x}$ na užším datasetu účtenek, kde bude možné předem odhadnout počet slov nebo délku mezer mezi řádky.

8.3 Rozpoznání znaků

Dalším zajímavý krokem by bylo použití jiného nástroje pro rozpoznávání znaků, než je Tesseract. Jeden takový byl už v práci testován v podobě komerčního Google Vision. Dalším, který má blíže k nástroji Tesseract a nese také označení open-source, je GOCR [9]. I když v práci [10] autor tvrdí, že Tesseract poskytuje lepší výsledky, bylo by zajímavé vidět porovnání těchto dvou nástrojů.

Samotný Tesseract ale prošel velkou změnou při přechodu z verze 3.X na v práci použitou verzi 4.X. Největší změna je v použití LTSM. Bylo by zajímavé otestovat obě verze a zjistit dopad použití LTSM.

Poslední možností by bylo navržení nového postupu pro rozpoznávání textu. Ten by mohl využít formátu účtenky. Tedy toho, že text je především černý na bílém pozadí. Je přesně zarovnán do rovnoběžných řádek a je strojově tištěný. Pro užší dataset účtenek (například pouze od jednoho prodejce) by se dal navíc analyzovat i font, délka mezer mezi řádky, počet slov a pozice hledaných faktů.

8.4 Extrakce faktů

V této práci se z účtenky za palivo vyčítá cena, čas a datum. Extrakce těchto faktů by mohla být rozšířena o další údaje, které se nacházejí na účtence. Například o typ čerpací stanice, typ paliva a počet natankovaných litrů. Pro úspěšné rozpoznání dalších faktů je nutné rozšířit extrakci faktů. To spočívá v analýze nových faktů, jejich formátu a klíčových slov, které se okolo nich nacházejí.

Dalším možným rozšířením je použití postupu na jiné dokumenty, než je tištěná účtenka za palivo. Například pro fotku občanského nebo řidičského průkazu. Pro tento případ by se musel otestovat vliv binarizace a mediánového filtru. Postup rotace závisí na struktuře textu, která musí obsahovat rovnoběžné řádky. V tomto případě by se rotace dala použít. Dále by se stejně jako je popsáno v předchozím odstavci musela rozšířit extrakce faktů.

Posledním možným rozšířením je použití celého postupu pro jiný jazyk. To spočívá ve dvou krocích. První je nastavení nástroje Tesseract pro daný jazyk. To se provede v procesu inicializace, kdy se načítají data pro jazyk. Druhým krokem je úprava extrakce faktů pro daný jazyk. V případě účtenky v jiném jazyce by stačilo analýzou dostatečného množství účtenek zjistit klíčová slova v cizím jazyce a těmi nahradit stávající.

Alternativou k v práci popsané extrakci faktů může být N-gram. Ten je například popsaný v práci [31]. V ní se autoři také věnují zpracování účtenek. N-gram je ale model založený na pravděpodobnosti, že se nějaká skupina slov nachází pohromadě. Tato pravděpodobnost se určí jednoduše analýzou textu. Pro účtenky to znamená přesný přepis jejich obsahu a určení této pravděpodobnosti. N-gram také nepředpokládá poškozené znaky, musel by být tedy upraven.

8.5 Implementační část

V prezentační části by uživatel měl možnost ořezat vstupní obrázek. Mohl by například vyříznout pouze účtenku, nebo rovnou část s cenou, časem nebo datumem. Mohl by také poskytnout zpětnou vazbu. V ní by uživatel označil údaje, které systém správně rozpoznal. Tím by mohl spustit další rozpoznávání špatně rozpoznaných údajů.

V business vrstvě je možným zlepšením snížení velikosti účtenky pomocí komprese. Tento fakt nijak nepomůže rozpoznávání znaků, ale zrychlí se výpočty a sníží se přenos dat mezi uživatelem a aplikací.

Závěr

V úvodu byly definovány dva cíle. Prvním z nich bylo vytvoření postupu pro rozpoznání textu na tištěné papírové účtence za palivo a extrakce ceny, času a datumu uvedeného na účtence. Druhým cílem bylo implementovat a otestovat tento postup ve formě jednoduché webové aplikace, kam bude možné nahrát obrázek tištěné účtenky a obdržet uvedené údaje. Oba cíle byly splněny.

První cíl byl splněn aplikací několika kroků na obrázek účtenky. První krok je její předzpracování. V něm se účtenka převede do odstínů šedi, aplikuje se mediánový filtr a Sauvolův binarizační algoritmus. V druhém kroku je účtenka správně natočena. Jsou pro to navrženy dva algoritmy. První používá Houghovu transformaci, druhý lineární regresi s K-means. Z předzpracované a natočené účtenky se pomocí OCR nástroje Tesseract rozpozná text. Z textu se extrakcí faktů získá cena, čas a datum. Využije se přitom jejich pevně daného formátu a pozici na účtence.

Navržený postup byl otestován a z výsledku plyne, že Tesseract poskytuje dobré výsledky i bez předzpracování a rotace účtenky. Dále, že mediánový filtr mírně zlepší výsledky pouze bez použití binarizace. Celkově ale nehraje velkou roli. Naopak se prokázalo, že Sauvolova binarizace není vhodná pro segmentaci textu u rozmanitého datasetu. Výsledky obou algoritmů pro rotaci účtenky neposkytly dobré výsledky, mají ale velký potenciál pro zlepšení.

Nejlepších výsledků dosáhla kombinace čistého rozpoznání textu pomocí nástroje Tesseract bez předzpracování účtenky a její rotace. V tomto případě navržený postup našel cenu na obrázku účtenky v 0,51 % případů, z toho v 0,46 % přesně. Čas našel v 0,64 % případů, z toho 0,48 % přesně. Datum našel v 0,73 % případů, z toho 0,4 % přesně.

Dalším testem bylo použití Google Vision. Tento komerční OCR nástroj nahradí všechny popisované kroky, zbyde pouze extrakce faktů. S použitím Google Vision se podařilo najít cenu na obrázku účtenky v 0,81 % případů, z toho v 0,64 % přesně. Čas v 0,82 % případů, z toho 0,48 % přesně. Datum v 0,92 % případů, z toho 0,55 % přesně.

Druhý cíl byl splněn naprogramováním aplikace, která se skládá ze dvou vrstev — prezentační a business. Prezentační vrstva pomocí jednoduchého tlačítka umožňuje uživateli nahrát obrázek účtenky k rozpoznání textu a extrakci faktů. Řeší přitom základní problémy jako jsou správné načtení obrázku od uživatele, jeho zakódování a odeslání na rozpoznání do business vrstvy. V business vrstvě jsou implementovány a odladěny všechny zmiňované algoritmy a postupy. Business vrstva předá výsledek rozpoznání prezentační vrstvě a ta ho zobrazí uživateli.

Celý postup může být rozšířen o nalezení dalších faktů. Například o typ čerpací stanice, typ paliva a počet natankovaných litrů. Postup se také může použít pro dokumenty s podobnou strukturou jako je účtenka. Například pro občanský nebo řidičský průkaz. V obou případech by stačilo upravit pouze extrakci faktů.

Bibliografie

1. VINCENT, Luc. Announcing Tesseract OCR. In: *googlecode.blogspot.com* [online]. 2006 [cit. 2020-04-27]. Dostupné z: `http://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html`.
2. SMITH, Ray; HEWLETT-PACKARD. *Tesseract. Version 4.1.1* [software]. 2005 [cit. 2020-04-24]. Dostupné z: `https://github.com/tesseract-ocr/tesseract`.
3. MATHUR, Natasha. Tesseract version 4.0 releases with new LSTM based engine, and an updated build system. In: *hub.packtpub.com* [online]. 2018 [cit. 2020-04-27]. Dostupné z: `https://hub.packtpub.com/tesseract-version-4-0-releases-with-new-lstm-based-engine-and-an-updated-build-system/`.
4. SMITH, Ray. An Overview of the Tesseract OCR Engine Ray. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. IEEE, 2007, s. 629–633. Dostupné také z: `https://research.google/pubs/pub33418/`.
5. PATEL, Chirag; PATEL, Atul; PATEL, Dharmendra. Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications* [online]. 2012, roč. 55, č. 10, s. 50–56 [cit. 2020-04-07]. Dostupné z DOI: `10.5120/8794-2784`.
6. SMITH, Ray; HEWLETT-PACKARD. *Tesseract Thresholder* [software]. 2019 [cit. 2020-04-24]. Dostupné z: `https://github.com/tesseract-ocr/tesseract/blob/master/src/ccmain/thresholder.cpp`. [funkce OtsuThresholdRectToPix].
7. OTSU, Nobuyuki. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*. 1979, roč. 9, č. 1, s. 62–66. ISSN 0018-9472. Dostupné z DOI: `10.1109/TSMC.1979.4310076`.

8. FILIP, Csaba. *Rozpoznávání matematických a chemických vzorců ve strukturovaných dokumentech*. Praha, 2019. Dostupné také z: https://dspace.cvut.cz/handle/10467/82368. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů. Vedoucí práce: Daniel Průša.
9. SCHULENBURG, Jörg. *GOCR* [software]. 2000 [cit. 2020-04-28]. Dostupné z: https://www-e.ovgu.de/jschulen/ocr/.
10. DHIMAN, Shivani; SINGH, A J. Tesseract Vs Gocr A Comparative Study. *International Journal of Recent Technology and Engineering* [online]. 2013, vol. 2, no. 4, s. 4 [cit. 2020-04-08]. ISSN 2277-3878. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.9685&rep=rep1&type=pdf.
11. GOOGLE. *Google Vision* [software]. 2017 [cit. 2020-04-25]. Dostupné z: https://cloud.google.com/vision.
12. NEVES, António J. R; LOPES, Daniel. A practical study about the Google Vision API. In: *RECPAD*. Aveiro, 2016, sv. 22. Dostupné také z: https://www.researchgate.net/publication/309642837.
13. HOSSEINI, Hossein; XIAO, Baicen; POOVENDRAN, Radha. Google's Cloud Vision API is Not Robust to Noise. In: *International Conference on Machine Learning and Applications*. IEEE, 2017, sv. 16, s. 101–105. ISBN 978-1-5386-1418-1. Dostupné z DOI: 10.1109/ICMLA.2017.0-172.
14. NGUYEN, Quan. *Tess4J. Version 4.5.1* [software]. 2010 [cit. 2020-04-24]. Dostupné z: http://tess4j.sourceforge.net/.
15. Improving the quality of the output. In: *tesseract-ocr.github.io* [online]. 2020 [cit. 2020-05-03]. Dostupné z: https://tesseract-ocr.github.io/tessdoc/ImproveQuality.
16. Median Filtering. In: *www.southampton.ac.uk* [online obrázek]. © 2005 University of Southampton [cit. 2020-04-03]. Dostupné z: https://www.southampton.ac.uk/~msn/book/new_demo/median/.
17. PRATIKAKIS, Ioannis; ZAGORIS, Konstantinos; KARAGIANNIS, Xenofon; TSOCHATZIDIS, Lazaros; MARTHOT-SANTANIELLO, Isabelle; MONDAL, Tanmoy. ICDAR 2019 Competition on Document Image Binarization (DIBCO 2019). In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, s. 1547–1556. ISBN 978-1-72813-014-9. ISSN 2379-2140. Dostupné z DOI: 10.1109/ICDAR.2019.00249.
18. SAUVOLA, Jaakko; PIETIKÄINEN, Matti. Adaptive document image binarization. *Pattern Recognition*. 2000, vol. 33, no. 2, s. 225–236. ISSN 00313203. Dostupné z DOI: 10.1016/S0031-3203(99)00055-2.

19. BADEKAS, Efthimios; PAPAMARKOS, Nikos. Automatic Evaluation of Document Binarization Results. In: *Progress in Pattern Recognition, Image Analysis and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, sv. 3773, s. 1005–1014. Lecture Notes in Computer Science. ISBN 978-3-540-29850-2. Dostupné z DOI: 10.1007/11578079_103.

20. RANGONI, Yves; RANGONI, Yves; BREUEL, Thomas M. OCR Based Thresholding. In: *MVA2009 IAPR Conference on Machine Vision Applications*. Yokohama, 2009, s. 3–18. Dostupné také z: https://www.researchgate.net/publication/228350447.

21. SHAFAIT, Faisal; KEYSERS, Daniel; BREUEL, Thomas M. Efficient implementation of local adaptive thresholding techniques using integral images. In: *Document Recognition and Retrieval XV*. SPIE, 2008, sv. 6815, s. 317–322. Dostupné z DOI: 10.1117/12.767755.

22. BRODIĆ, Darko; MILIVOJEVIĆ, Dragan R. An algorithm for the estimation of the initial text skew. *Information Technology and Control*. 2012, roč. 41, č. 3, s. 211–219. ISSN 1392124X. Dostupné z DOI: 10.5755/j01.itc.41.3.1249.

23. Morfologické operace. In: *www.vutbr.cz* [online obrázek]. © 2010 Karel Horak, Computer Vision Group [cit. 2020-04-27]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ZVS/Exercise10/content_cz.php.

24. KLOUDA, Karel; VAŠATA, Daniel. *Algoritmus k-means* [přednáška]. 2019 [cit. 2020-05-08]. Dostupné z: https://courses.fit.cvut.cz/BI-VZD/lectures/files/BI-VZD-04-cs-handout.pdf. [Soubor přístupný po přihlášení do sítě ČVUT – kopie souboru uložena na přiloženém médiu; str 41].

25. IMAI, Hiroshi; ASANO, Takao. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*. 1983, roč. 4, č. 4, s. 310–323. ISSN 01966774. Dostupné z DOI: 10.1016/0196-6774(83)90012-3.

26. NOVÁK, Petr. *Lineární regrese* [přednáška]. 2019 [cit. 2020-05-08]. Dostupné z: https://courses.fit.cvut.cz/BI-PST/media/lectures/bi-pst-lec12-handout.pdf. [Soubor přístupný po přihlášení do sítě ČVUT – kopie souboru uložena na přiloženém médiu; str 164].

27. DUDA, Richard O.; HART, Peter E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*. 1972, roč. 15, č. 1, s. 11–15. ISSN 15577317. Dostupné z DOI: 10.1145/361237.361242.

28. LIKFORMAN-SULEM, L.; HANIMYAN, A.; FAURE, C. A Hough based algorithm for extracting text lines in handwritten documents. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Montreal, Que., Canada: IEEE Comput. Soc. Press, 1995, sv. 2, s. 774–777. ISBN 978-0-8186-7128-9. Dostupné z DOI: 10.1109/ICDAR.1995.602017.

29. DOCKER INC. *Docker. Version 19.03.8* [software]. 2013 [cit. 2020-04-26]. Dostupné z: https://www.docker.com/.

30. MOZILLA CORPORATION. *Mozilla Firefox. Version 75.0* [software]. 2002 [cit. 2020-04-24]. Dostupné z: https://www.mozilla.org/cs/firefox/new/.

31. ODD, Joel; THEOLOGOU, Emil. *Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms*. Linköping, 2018. Dostupné také z: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1215460&dswid=-7593. Bakalářská práce. Linköpings universitet, Department of Computer and Information Science. Vedoucí práce: Rita Kovordanyi.

32. CAMPS, Claudi Ruiz. *Segmentation, labeling and optical character recognition applied on receipt images*. Barcelona, 2016. Dostupné také z: http://hdl.handle.net/2117/97003. Diplomová práce. Universitat Politècnica de Catalunya. Vedoucí práce: Angulo Bahón.

Příloha A

Parametry

Vysvětlení významu parametrů v tabulkách A.1 a A.2.

- `sauvola`: značí použití Sauvolova binarizačního algoritmu
- `K, W, R`: parametry Sauvolova binarizačního algoritmu
- `median`: značí použití mediánového filtru
- `median k`: velikost okénka mediánového filtru
- `metoda`: použitá metoda pro rotaci účtenky
- `rho resolution`: parametr u Houghovy transformace
- `K-means iter`: počet iterací u algoritmu K-means
- `width scale`: parametr u metody lineární regrese, škálování šířky bounding boxů
- `-`: parametr nebyl použit

	sauvola	K	W	R	median	median k	metoda	rho resolution
testCase1	True	0,34	14	128	False	-	hough	0,9
testCase2	True	0,4	60	128	False	-	hough	0,9
testCase3	True	0,5	15	128	False	-	hough	0,9
testCase4	True	0,5	15	128	True	3	hough	0,9
testCase5	True	0,4	60	128	True	3	hough	0,9
testCase6	False	-	-	-	True	3	hough	0,9
testCase7	False	-	-	-	False	-	hough	0,9
testCase8	False	-	-	-	False	-	-	-

Tabulka A.1: Parametry testCase

	metoda	rho resolution	K-means iter	width scale	sauvola	K	W	R	median
rotationTest1	hough	0,9	-	-	True	0,34	14	128	False
rotationTest2	hough	0,9	-	-	False	-	-	-	False
rotationTest3	lin. regrese	-	3	0,25	True	0,34	14	128	False
rotationTest4	lin. regrese	-	3	0,25	False	-	-	-	False

Tabulka A.2: Parametry rotationTest

	1	2	3	4	5	6	7	8
`předcházející`	cena	cena:	celkem	celkem:	prodej	částka	suma	suma:
`následující`	kc	kč	czk					
příklad ceny	1490,90	1532.48	3.065,10					

Tabulka A.3: Klíčová slova pro cenu, příklady formátů

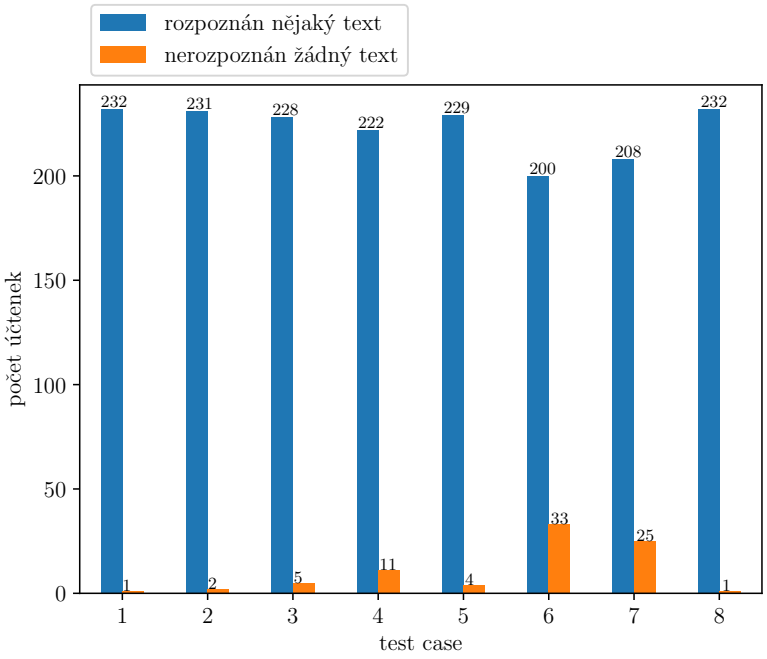
	1	2	3	4	5	6	7	8
`předcházející`	čas	cas	datum	Dat.usk.zd.pl:	plnění	(DUZP)	zdan.plneni	dodání
příklad času	16:06:47	09:04:07	7:08:58	15:38				
příklad datumu	23.02.2017	08.02.2020	23.4.2019	04-12-2017	2018-01-26	06/07/2018	14/09/18	

Tabulka A.4: Klíčová slova pro čas a datum, příklady formátů

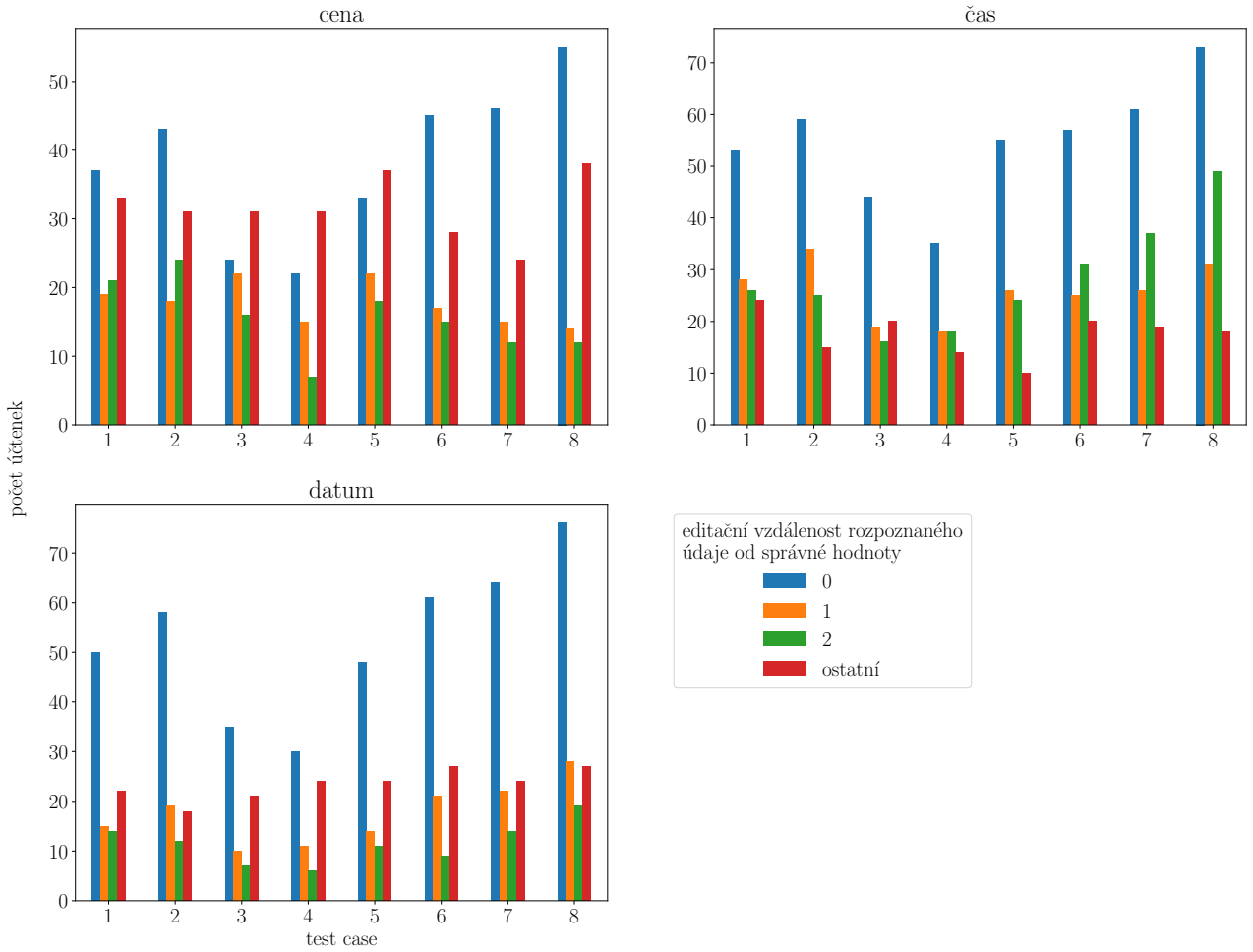
Příloha B

Grafy výsledků testování

B.1 TestCase

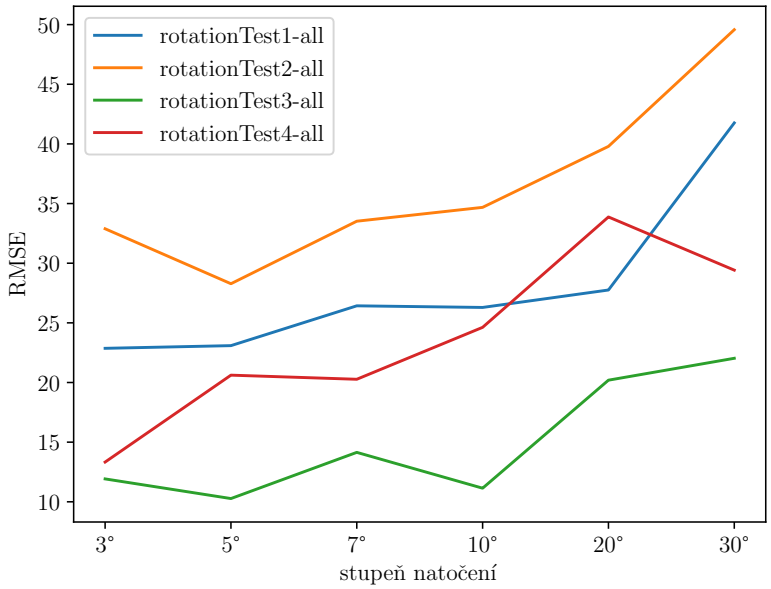


Obrázek B.1: TestCase – rozpoznání nějakého textu

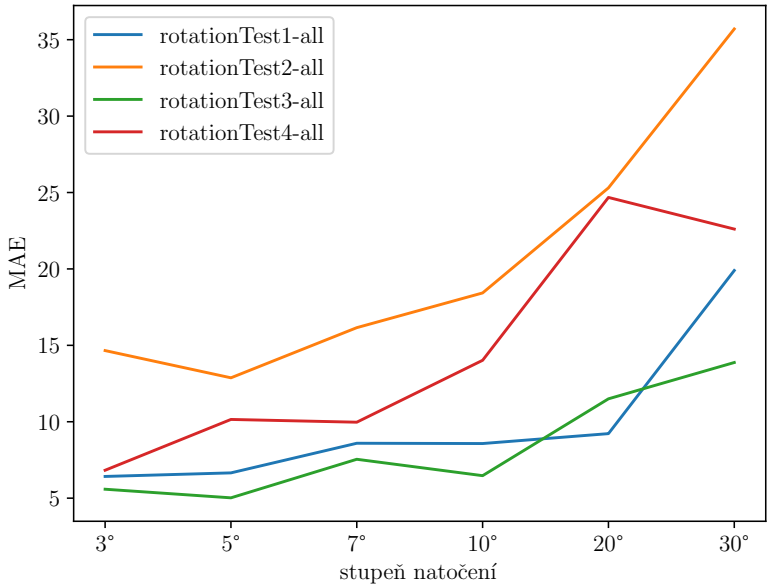


Obrázek B.2: TestCase – editační vzdálenosti od správné hodnoty

B.2 RotationTest

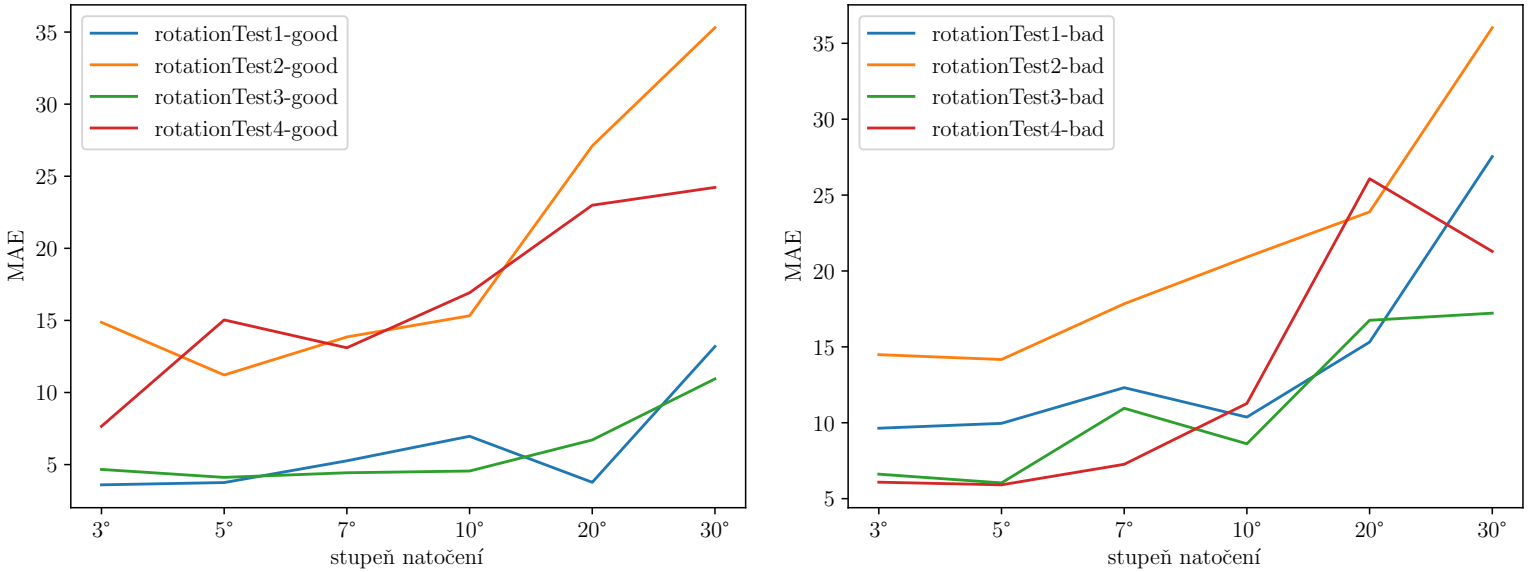


(a) RMSE



(b) MAE

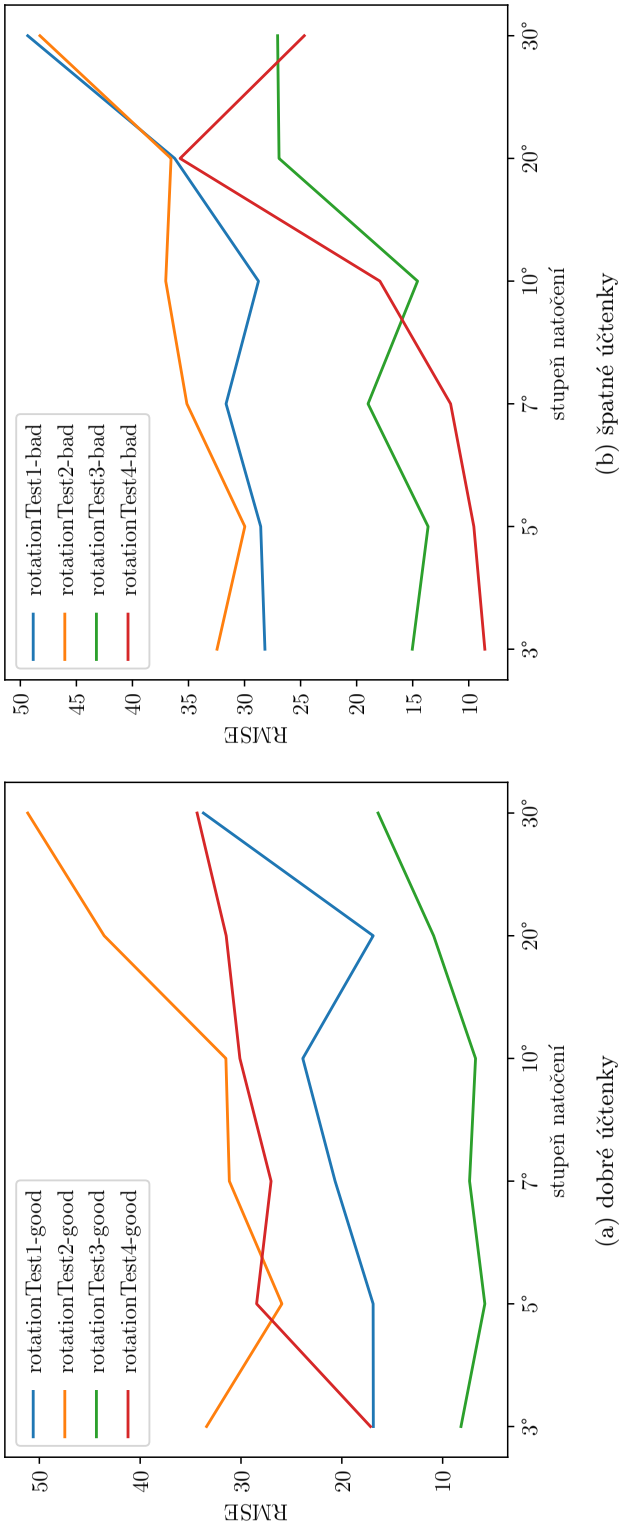
Obrázek B.3: RotationTest – celý dataset



(a) dobré účtenky

(b) špatné účtenky

Obrázek B.4: RotationTest – MAE

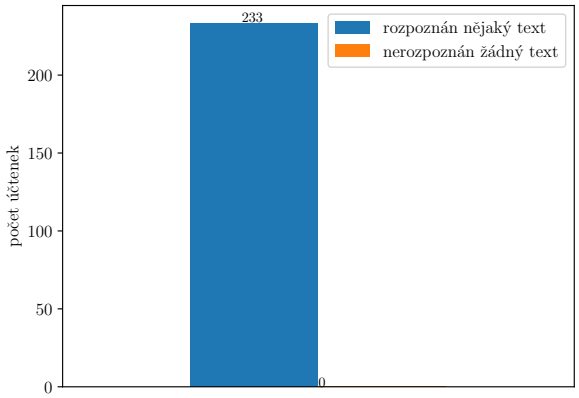


(a) dobré účtenky

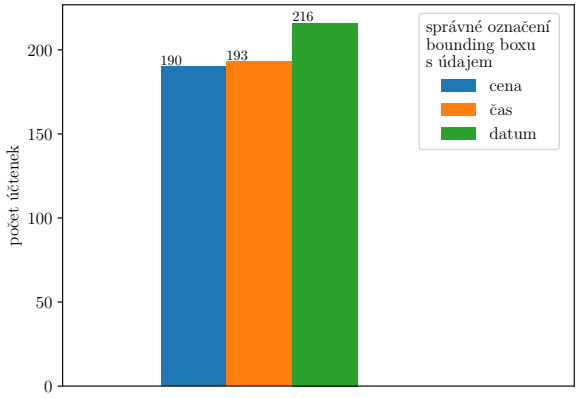
(b) špatné účtenky

Obrázek B.5: RotationTest – RMSE

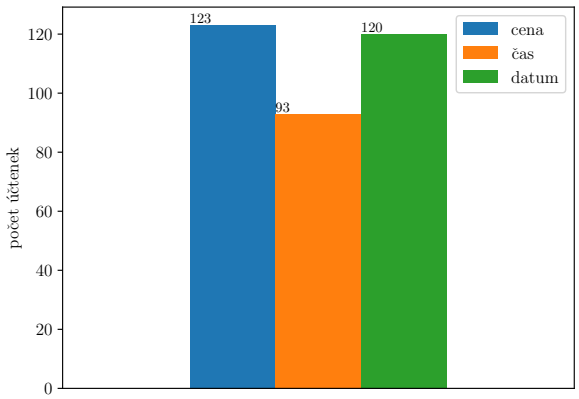
B.3 Google Vision



(a) rozpoznání nějakého textu

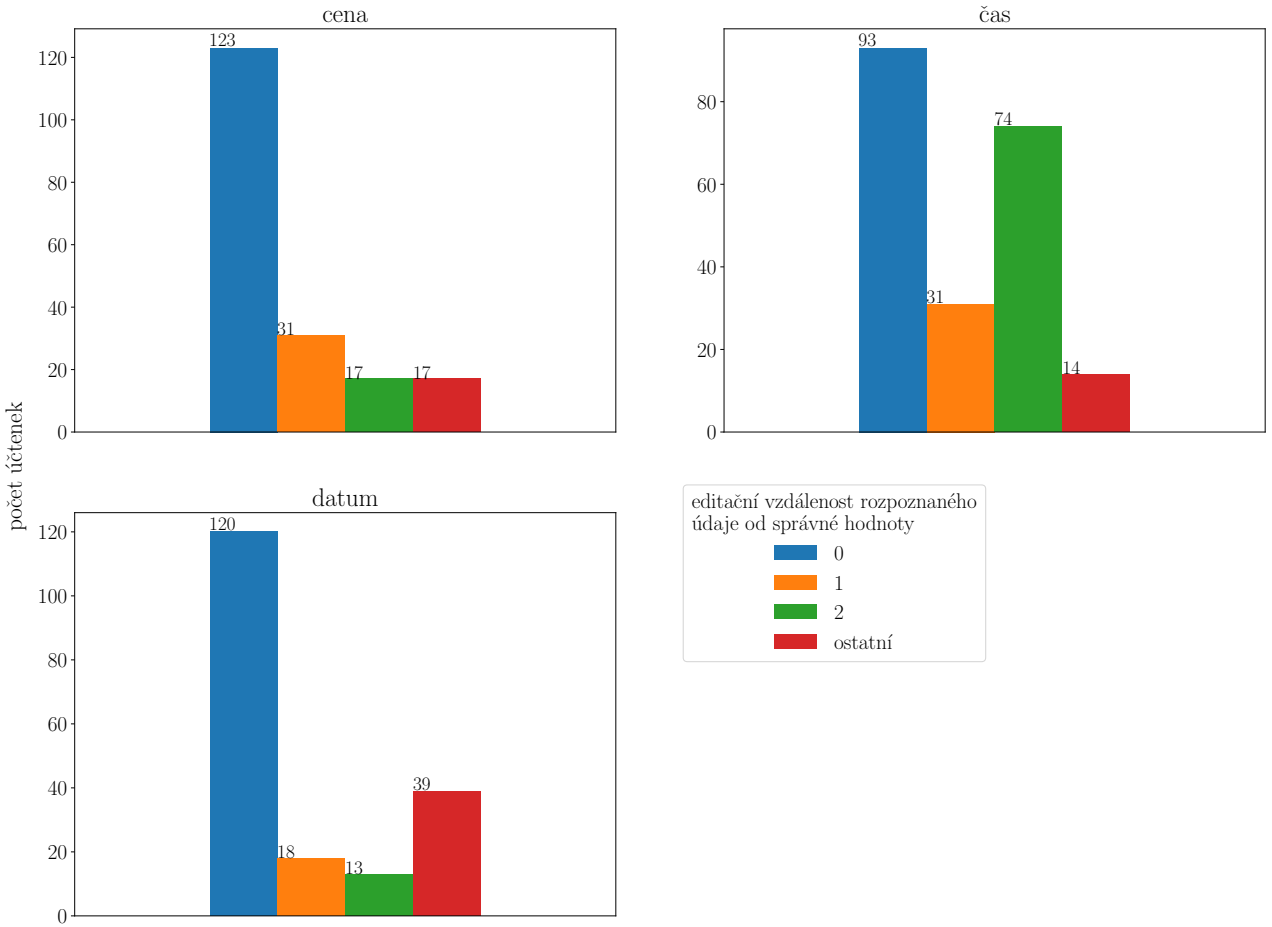


(b) správné označení bounding boxu



(c) přesně rozpoznané údaje

Obrázek B.6: Google Vision



Obrázek B.7: Google Vision – editační vzdálenost od správné hodnoty

Příloha C

Seznam použitých zkratek

API Application programming interface

DPI Dots per inch

EXIF Exchangeable image file format

LTSM Long short-term memory

MAE Mean squared error

OCR Optical character recognition

RMSE Root mean square error

Příloha D

Obsah přiložené SD karty

- `README.txt` ..... stručný popis obsahu SD karty
- `docker`
  - `tessapi.tar` ..... docker image s aplikací
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `tutorials` ..... přednášky s definicemi
- `dataset` ..... obrázky účtenek