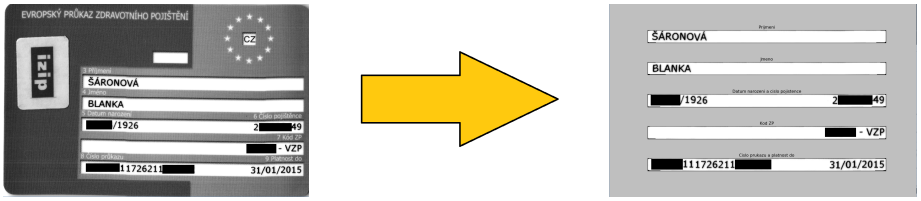
BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Detekce formulářových polí ve skenovaných dokumentech

Michal Moravec



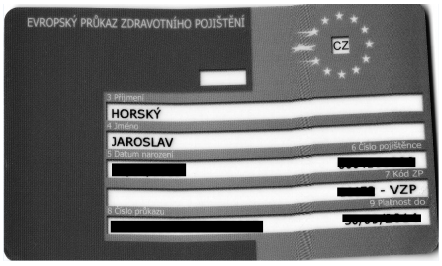
Abstrakt

Cílem práce bylo navrhnout algoritmus, který bude schopen z karty zdravotní pojišťovny vyseparovat pouze textová pole, která se budou dát dále použít v libovolném softwaru na převedení obrázku na text. Program by měl počítat se špatně naskenovanými a libovolně otočenými kartami. Celkový projekt je dělán jako zakázka pro firmu Medingo, která chce algoritmus zakomponovat do svého stávajícího systému, který bude popsán dále. Co se týče výsledků, tak algoritmus dokáže detekovat a vyseparovat textová pole s velmi vysokou pravděpodobností.

Klíčová slova: Zpracování obrazu — Karta pojišťovny — OpenCV

Přiložené materiály: *N/A*

*xmorav28@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké Učení Technické v Brně*



Obrázek 1. Ukázka špatného naskenování.

1. Úvod

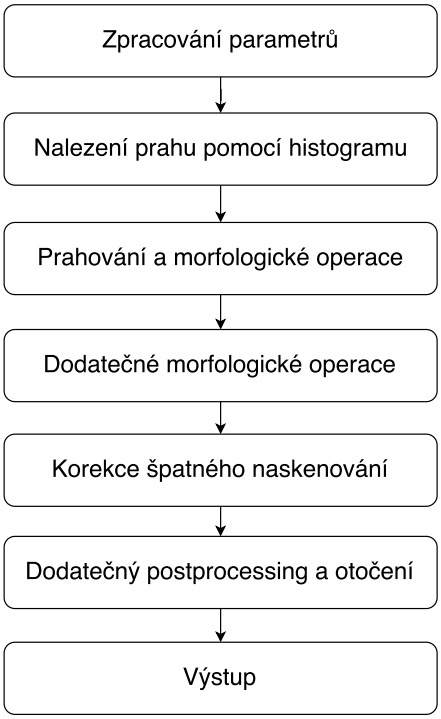
Hlavním důvodem pro vytvoření tohoto projektu je zlevnění pracovní síly a automatizace zpracování skenovaných dokumentů. Úkolem bylo navrhnout algoritmus, kde je vstupem karta zdravotní pojišťovny, která může být libovolně otočená, a výstupem jsou vyfiltrovaná jednotlivá textová pole, které tato karta obsahuje. Vstupní karta je načtena na skenovacím zařízení, ve kterém se může během skenování pohnout, což způsobí poškození, se kterým musí algoritmus také počítat (viz Obrázek 1).

Inspirací je firma Medingo, která má vyvinutý systém Evipa. Tento systém obsahuje čtečku karet zdravotní pojišťovny, která je napojena na počítač sestry u lékaře. Cílem tohoto systému je naskenování karty od pacienta, dále zpracování této karty tak, aby se na počítači sestry objevily konkrétní údaje o pacientovi z databáze nalezené automaticky, čímž se urychlí čekání v čekárně. Projekt řeším, protože firma měla problém s implementací algoritmu pro detekci textových polí, který chtěla značně vylepšit.

Co se týče existujících projektů, které řeší podobný problém, tak stojí za zmínku algoritmus detekce textových polí v obrázcích digitálních dokumentů pomocí texturálních rysů, který tato pole detekuje pomocí Gaborova filtru [1]. Další práce řeší zarovnání otočeného textu v rukou napsaných dokumentech [2].

2. Návrh řešení

V této sekci jsou popsány jednotlivé položky z blokového diagramu 2.



Obrázek 2. Blokový diagram programu.

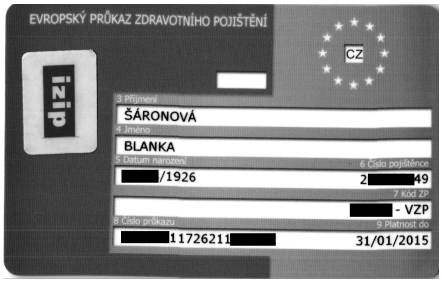
Moje řešení funguje tak, že nejprve se načte naskenovaný obrázek, který se pak pomocí metod z oboru zpracování obrazu přetransformuje na černobílý obrázek, kde jsou zvýrazněna textová pole bílou barvou a zbytek černou barvou. Potom je třeba od sebe oddělit černý text, který je uvnitř bílých textových polí. Po tomto oddělení je vytvořena maska, která když se aplikuje na původní obrázek, tak výsledkem jsou vyfiltrovaná textová pole. Po této filtraci je také nutné textová pole opravit, když je třeba, což se děje spočítáním průměrné výšky textového pole, následného posunutí jeden pixel širokých sloupců do jedné výšky a oříznutí přebytečných částí.

2.1 Zpracování parametrů

Jelikož program měl mít několik různých výstupů, které bylo třeba specifikovat parametry, bylo třeba tyto parametry ošetřit.

První možností je volba, která určí, kolik položek bude obsahovat výstupní vektor. Jelikož jsou všechny karty zdravotní pojišťovny v jednotném formátu, který obsahuje 5 textových polí (viz Obrázek 3), je základní možností těchto 5 polí. Navíc je tu ale možnost vrátit textových polí 7, jelikož jsou na kartě 2 pole, která obsahují více než jednu informaci. Takto se tedy vrátí v každém samostatném výstupu jedna informace.

Druhou možností je parametr, který určí, zda-li výstupní textová pole jsou jen vystřižená z původního obrázku, nebo je na ně dále aplikováno adaptivní prahování pomocí histogramu, které zajistí lepší oddělení černého textu od bílého pozadí. Někdy se totiž může



Obrázek 3. Ukázka karty zdravotní pojišťovny.

Obrázek 4. Ukázka druhé strany karty.

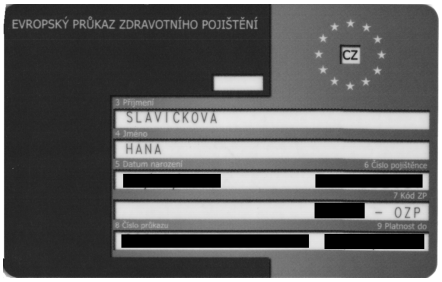
stát, že mají textová pole mírně odlišné barevné rozložení, což znamená, že text je např. šedý. Tato operace tedy zajistí vždy stejné barevné rozložení obsahů textových polí.

Co se týče výstupů z programu, tak je zde již uvedený vektor textových polí, ve kterém je parametricky daný počet prvků. Může se ale také stát, že jako vstup do programu přijde karta VZP, která je naskenována z druhé strany (viz Obrázek 4). Tato možnost je ošetřena tak, že výstupní vektor se vrátí prázdný.

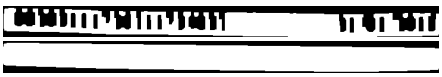
Po ošetření vstupních parametrů je třeba udělat kontrolu otočení karty. Může se stát, že karta je otočená o 90 nebo 270 stupňů, což se dá opravit jednoduchým testem na správný poměr stran a následné otočení zpět. Na funkci programu nemá vliv, jestli je karta otočená o 180 stupňů, což znamená, že tato korekce se provede až později.

2.2 Nalezení prahu pomocí histogramu

Dále následuje nalezení prahu pomocí histogramu, na což jsou implementovány dvě funkce. První funkce zajistí nalezení prahu tak, že prochází histogram zleva (od černé barvy až po bílou), kde hledá lokální minimum. Toto minimum hledá tak, že nejprve najde v první čtvrtině histogramu zleva lokální maximum, a od něj jde směrem vpravo do nejbližšího lokálního minima. Tímto se zachytí správné spektrum barev, které označuje černý text. Toto tedy funguje i v případě, že text není úplně černý a lokální maximum se nachází ne úplně vlevo na histogramu (viz Obrázek 5). Druhá funkce funguje analogicky, jen histogram prochází zprava a hledá lokální minimum pro určení prahu, který ideálně označí bílá textová pole.



Obrázek 5. Ukázka jiné barvy textu.



Obrázek 6. Textové pole s konkávními (nahoře) a bez konkávních nerovností (dole).

2.3 Prahování a morfologické operace

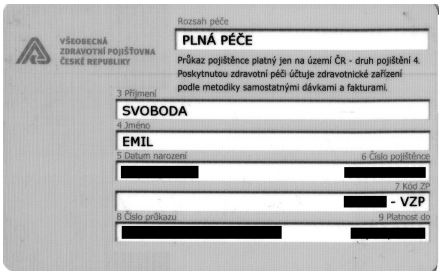
Po načtení vstupního obrázku se tedy provede prahování. Prahovaný obrázek se binárně invertuje a nakonec se provede morfologická operace uzavření. Tyto operace připraví obrázek na vyhledání obrysů (kontur), přičemž do tohoto vyhledání budou spadat i kontury, které nejsou textovými poli. Toto je třeba ošetřit testem, který ověří, jestli obrys splňuje parametry textového pole, jako třeba správný poměr stran a relativní obsah vůči obsahu celé karty. Jelikož po operaci uzavření ale můžou být textová pole rozdělena na více části, je nutné v tomto kroku vyřadit jen ty obrysy, které jsou buď moc blízko okraji obrázku, nebo mají příliš velký obsah. Tyto vyfiltrované obrysy se uloží do pole a aplikuje se na ně operace konvexní obálky, která zajistí, že žádné obrysy nemají konkávní nerovnosti (viz ukázka textového pole s konkávní nerovností a bez nerovnosti (viz Obrázek 6).

2.4 Dodatečné morfologické operace

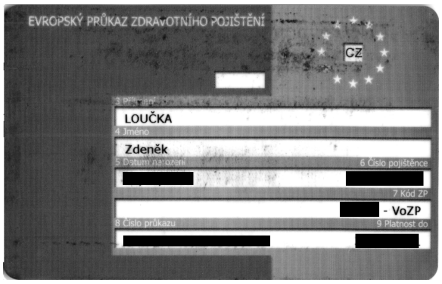
Když je naskenována karta bez nečistot, měla by v tento moment být vyfiltrována textová pole. Když je ale naskenována karta s poškozením (viz Obrázek 8), může se stát, že textová pole nejsou v tento moment propojena. Je tedy třeba aplikovat ještě další sadu operací, které zajistí správnost detekce polí. Aplikuje se tedy další operace uzavření, tentokrát ale s obdélníkovou maticí mající šířku větší než výšku, která tyto pole propojí. Nyní jsou již všechna pole spojena a připravena k výslednému vyhledání obrysů.

2.5 Korekce špatného naskenování

Vhledem k předpokladu, že jsou textová pole celistvá, je lze testovat na správný poměr stran. Po ověření správného poměru stran lze tyto textová pole uložit do kolekce, ve které je můžeme považovat za finální. Nyní je třeba ještě provést korekci posunutých polí, což se může stát při špatném naskenování, např. když se s kartou během skenování pohne, což může způsobit



Obrázek 7. Ukázka starší karty zdravotní pojišťovny.



Obrázek 8. Ukázka karty s poškozením.

zkosení textového pole, nebo protažení, ať už v horizontálním, nebo vertikálním směru. Na tuto korekci existuje funkce, která tyto nedostatky opraví tak, že nejprve spočítá průměrnou výšku pole, a do této jednotné výšky nakopíruje jednotlivé jednopixelové sloupce, které dle potřeby buď doplní o bílé pixely, nebo ořízne. Po této operaci jsou tedy textová pole připravena k výstupu.

2.6 Dodatečný postprocessing a otočení

Nejprve se provedou dodatečné testy, které zjistí, zdali není textových polí odhaleno málo (je naskenována zadní strana karty) nebo příliš mnoho (může se jednat o starý formát karty (Obrázek 7), kde bylo navíc jedno pole). Když je textových polí přesně o jedno více, než má být, zahodí se první pole, které u staršího formátu karet existuje, ale u nového už ne. Tímto se zajistí konzistentní výstup. Nakonec je ještě třeba otočit textová pole, která jsou otočena o 180 stupňů. Toto se dá detekovat tak, že se vezmou souřadnice středů všech textových polí, z nich se udělá vážený průměr, čímž vznikne průměrný střed, který by se měl nacházet vpravo od středu karty. Jestliže je lehce nalevo, znamená to, že je karta otočena o 180 stupňů. Když je zadán parametr pro dodatečný postprocessing, tak se po této korekci se ještě aplikuje prahování, jinak se vezme textové pole z původního obrázku.

2.7 Výstup

Když je požadován výstup jen pěti textových polí, neprovádí se žádné další operace, a výstupem programu jsou textová pole zpracovaná z předchozího odstavce.

Když je zadán parametr, který označuje, že má program vrátit místo pěti textových polí polí sedm, je

třeba ještě udělat některé další operace, které toto zajistí. Z hlediska optimalizace je na začátku programu uvedeno, která pole obsahují více než jednu informaci, a jsou tedy dělitelná. Na tyto pole se aplikuje morfologická operace eroze a dále se testuje, zda-li v rámci jednopixelového sloupce je více než 50% černé barvy. Když je tato podmínka splněna, tento jednopixelový sloupec je vyplněn jen černou barvou, jinak je vyplněn jen bílou. Tato operace zajistí, že místa, která obsahují text, jsou označena černou barvou, a ostatní bílou. Po binární inverzi se dále v takto upraveném textovém poli naleznou obrysy, které vyfiltrují výsledná místa, kde se nachází text. Tato výsledná místa se tedy vrátí vedle těch již dříve nalezených.

3. Experimenty a implementace

3.1 Implementace

Implementace programu byla provedena ve skriptovacím jazyce `Python 2.7` [1], výstupy ve formě obrázků pomocí knihovny `Matplotlib` a samotné funkce pro zpracování obrazu byly použity z knihovny `OpenCV` [2]. Pro ošetření argumentů v Pythonu je použita knihovna `argparse`.

3.2 Data

K testování mi byla poskytnuta data ve formě naskenovaných karet zdravotní pojišťovny přímo v reálné čtečce Evipa. Celkově poskytnutých dat bylo 317, z toho 21 bylo zadních částí karet a 296 bylo předních částí. Z těchto dat jsem vybral 20 karet, ve kterých byly obsaženy správně naskenované karty bez poškození, špatně naskenované karty s posunutím, špinavé karty, reálně poškozené karty a starší karty.

3.3 Experimenty

Na této sadě dvaceti karet jsem si označil textová pole, která by program měl vyfiltrovat a naladil metodu tak, aby ze vstupních dvaceti karet bylo 100 výstupních textových polí. Úspěšnost algoritmu na těchto datech je tedy 100%. Dále jsem algoritmus aplikoval na všechna poskytnutá data, kde algoritmus byl schopen detekovat správně zadní část karty v 90.4% ze 21 případů. U přední části karet, kde musel algoritmus nejen detekovat, že se jedná o přední stranu, ale také vrátit přesně 5 textových polí, měl 96.3% úspěšnost z 296 případů.

4. Závěr

Tato práce vysvětlila čtenáři všechny důležité prvky ohledně programu pro detekci textových polí ze skenovaných dokumentů, jako popis algoritmu, implementace a experimenty.

Co se týče výsledků, tak program funguje na 96.3% předních částech skenovaných karet, které byly poskytnuty jako testovací sada. Tímto byla maximalizována šance pro co nejlepší funkčnost řešení v reálných podmínkách. Byl také kladen důraz na rychlost řešení a optimalizaci, jako např. použití rychlých funkcí z knihovny `OpenCV`.

Program byl vytvořen na zakázku firmě Medingo, která chtěla vylepšit svůj stávající algoritmus pro detekci textových polí.

Do budoucna je algoritmus vlastněný touto firmou a měl by být rozšiřitelný v případě, že to bude potřeba.

[1] https://docs.python.org/2/

[2] http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

Poděkování

Chtěl bych tímto poděkovat svému vedoucímu, Ing. Vítězslavu Beranovi, Ph.D., díky kterému jsem práci úspěšně dokončil.

Literatura

[1] Ilktan Ar and M. Elif Karsligil. Text area detection in digital documents images using textural features. In *Proceedings of the 12th International Conference on Computer Analysis of Images and Patterns*, CAIP'07, pages 555–562, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] E. M. Kornfield, R. Manmatha, and J. Allan. Text alignment with handwritten documents. In *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*, pages 195–209, 2004.