VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

IPHONE APLIKACE PRO ROZPOZNÁNÍ SPZ

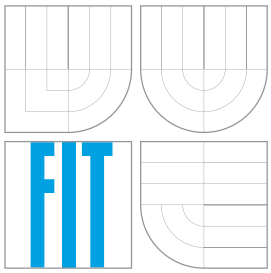
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN SLÁDEČEK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

IPHONE APLIKACE PRO ROZPOZNÁNÍ SPZ

IPHONE APPLICATION FOR NUMBER PLATE RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ROMAN SLÁDEČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. BORIS PROCHÁZKA

BRNO 2011

Abstrakt

Tato práce se zabývá teoretickou a praktickou stránkou tvorby aplikace pro mobilní zařízení Apple iPhone a jeho operační systém iOS. V teoretické části práce jsou obsaženy informace o historii zařízení, o nástrojích k tvorbě aplikací pro tuto mobilní platformu a stručně načrtnutý postup a proces, jakým je aplikace nasazena do samotného zařízení. Další část pojednáva o státních poznávacích značkách, jejich významu a integritních omezeních. Jádrem celé práce je praktická část, a tedy tvorba a implementace mobilní aplikace pro rozpoznávání těchto značek za podpory vestavěného fotoaparátu a její následná textová reprezentace, příp. další manipulace s tímto údajem. V závěrečné, no o nic méně důležité části práce zhodnotim výsledky, které tato aplikace přinese po testování a experimentování v různých podmínkách.

Abstract

This thesis talks about theoretical and practical side of development of application for Apple iPhone mobile cell phone and for its operating system called iOS. In theoretical part are contained informations about history of this device, about tools for creating application for this mobile platform and there is shortly sketched procedure and process how the application is deployed into the phone. Next part deals with the licence number plates, their signification and integrity restrictions. Main core of whole thesis is a practical part about creation and implementation mobile application for recognition of that number plates with supporting of built-in camera and its consecutive text representation or manipulation with that data. In the final part there is a review of results which this application brings after testing and experimenting in different conditions.

Klíčová slova

iPhone, rozpoznávání písma, státní poznávací značka, evidenční číslo vozidla, fotoaparát, mobilní telefon, Apple, iOS, segmentace, lokalizace, vyhledávání vzorů, klasifikace

Keywords

iPhone, text recognition, licence number plate, camera, mobile phone, Apple, iOS, segmentation, localization, template matching, classification

Citace

Roman Sládeček: iPhone aplikace pro rozpoznání SPZ, bakalářská práce, Brno, FIT VUT v Brně, 2011

iPhone aplikace pro rozpoznání SPZ

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Borise Procházky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Sládeček
18. května 2011

Poděkování

Děkuji p. Ing. Borisi Procházkovi za rady a návrhy v dalším postupu, který mi pomohl při práci na tomto projektu. Můj velký dík patří Martinovi Kissovi, mladému iPhone vývojáři, který mi v průběhu celého vývoje mé aplikace, poskytoval cenné rady. Také děkuji rodině, přátelům a všem, kteří mě podporovali a motivovali k práci.

Obsah

Úvod

V tejto práci sa venujem vývoju mobilnej aplikácie, ktorá slúži na rozpoznávanie slovenských štátnych poznávacích značiek (ŠPZ).

V 1. kapitole práce je popísané zariadenie, pre ktoré aplikáciu vytváram, Apple iPhone, konkrétne jeho 4. generácia (iPhone 4). V stručnosti čitateľa oboznámim s jeho parametrami, konštrukciou a tvorbou aplikací (2. kapitola) pre operačný systém iOS, ktorý toto mobilné zariadenie v súčasnosti obsahuje vo verzii 4[1] . Zameral som sa prevažne na popis tých hardwarových súčastí a softwarových funkcionalít, ktoré sa tvorby mojej aplikácie aspoň minimálne dotýkajú. Nakoľko je filozofia spoločnosti Apple, čo sa týka tvorby a následne schvaľovacieho procesu aplikácie mierne špecifická, zmienim sa v krátkosti aj o nej. Nasadenie aplikácie do reálneho zariadenia je podmienené vlastníctvom certifikátu vývojára, preto som pri počiatočnej fáze vývoja aplikáciu testoval na simulátore, ktorý je súčasťou vývojového prostredia pre túto platformu.

V 3. kapitole je popísaná problematika štátnych poznávacích značiek, kategorizácia, rozdielne tvary a ich integritné obmedzenia. Sústredil som sa najmä na slovenské ŠPZ a ich špecifiká. Všeobecne načrtnem aj odlišnosti českých ŠPZ.

Ďalej rozoberiem objektový návrh celej aplikácie a popíšem grafické užívateľské rozhranie (4. kapitola).

Nasleduje gro celej práce (5. kapitola) a tým je rozpoznávanie určitých objektov v obraze, v mojom prípade lokalizácia a rozpoznanie ŠPZ na fotografii automobilu. Popisujem metódy, ktoré som zvolil pri lokalizácii značky, segmentácii, rozpoznaní a klasifikácii jednotlivých znakov, ich návrh a implementáciu v rámci aplikácie.

V poslednej (6.) kapitole a v Prílohe A podrobím hotovú aplikáciu tzv. *test case* - testovacím scenárom pre objektívne posúdenie úspešnosti jednotlivých častí aplikácie.

V úplnom závere zhodnotím celkovú úspešnosť a prínos práce, dosiahnuté výsledky a načrtnem prípadné možné rozšírenia tejto mobilnej aplikácie (7. kapitola).

Pre dostatočné preniknutie do problematiky práce s obrazom mi boli nápomocné tieto knihy [19] [13] [8] [18]. Segmentáciu a klasifikovanie znakov mi bližsie predostreli tieto publikácie [11] [10] a na riešenie niektorých problematických úloh pri programovaní pre iPhone som námety čerpal aj z týchto zdrojov [3] [7] [2] [15].

[1] iOS 4.3.1 ku dňu 18.5.2011.

Kapitola 1

Apple iPhone 4

1.1 História a všeobecné informácie

Prvý telefón predstavila spoločnosť Apple, na čele s výkonným riaditeľom spoločnosti Steve Jobsom, na verejnej výstave Macworld v lete roku 2007. Dostal jednoduchý názov iPhone. V tom čase sa jednalo o telefón s netradičným ovládaním, takmer bez jediného tlačítka na svojom tele, disponujúci obrovským displejom. Postupom času sa ukázalo, že práve týmto impulzom vznikol nový trend v oblasti mobilného priemyslu. S pravidelnou presnosťou, každý rok Apple predstavil nového nástupcu pôvodného iPhonu, konkrétne s označeniami 3G (začala v USA vznikať 3G sieť), 3GS a v roku 2010 sme sa dočkali verzie 4. Tá so sebou priniesla množstvo noviniek a vylepšení, niektoré z nich budú v tejto práci popísané bližsie.

Obrázok 1.1: Mobilný telefón Apple iPhone 4

1.2 Hardware

Zariadenie iPhone je plne dotykový mobilný telefón. Disponuje jediným hardwarovým tlačítkom na prednej strane, dvomi tlačítkami pre ovládanie hlasitosti na bočnej strane, prepínačom tichého režimu a tlačidlom pre vypnutie prístroja na hornej hrane telefónu. Celkové ovládanie spočiva v multidotykovom displeji, ktorý je kapacitný, reaguje teda na vodivosť ľudského tela.

Telefón disponuje pomerne rýchlym Apple A4 procesorom, podtaktovaným z 1 GHz na 800 MHz a grafickou jednotkou PowerVR SGX 535. V zariadení je integrovaná operačná pamäť RAM o veľkosti 512 MB [1]. Do predaja sa telefón dostal v dvoch variantách líšiacich sa v jedinom parametri - flash pamäti pre ukladanie dát. Pri vývoji sme mali k dispozícii zariadenie so 16 GB, existuje ešte dvojnásobne väčšie.

V súvislosti s touto prácou je na mieste spomenúť hlavne parametre vstavaného fotoaparátu. Jeho rozlíšenie 5 Mpx (2596 x 1936 bodov) výrazne napomáha presnejšiemu spracovaniu obrazu a jeho následnej analýze pri rozpoznaní písma. Vedľa objektívu je umiestnená LED dióda, ktorá výrazne uľahčuje prácu s aplikáciou pri nepriaznivých svetelných podmienkach.

1.3 Operačný systém

Mobilné zariadenie iPhone už od svojho vzniku disponuje operačným systémom UNIXového typu. Jeho názov sa postupne minimalizoval z pôvodného iPhone OS na súčasný iOS. Rovnakým operačným systémom je vybavená aj najnovšia verzia hudobného prehrávača iPod Touch, ktorý je v podstate hardwarovo totožný s iPhone až na absenciu telefónnej časti, GPS a pár ďalších detailov[1]. iPad, dotykový tablet, ktorý Apple predstavil na trh nedávno už v 2. verzii, takisto umožňuje v prispôsobenom režime spúšťanie iPhone aplikácii. Pre neho sa primárne aplikácie vytvárajú so špeciálnou úpravou vzhľadom k väčšiemu rozmeru displeja.

Už v iPhone OS verzii 3.2 bola predstavená možnosť systémového zachytávania a rozpoznávania **multidotykových gest**, ktoré ďalej poskytuje aplikácii. V našej aplikácii budeme používať štandardné gestá, akými sú jednoduchý dotyk (*tap*), viacnásobný dotyk (*double tap, tripple tap, ...*), stlačenie, ťahanie, švihnutie (*swipe*), zovretie a rozovretie prstov (*pinch-to-zoom*) a iné.

Od verzie iOS 4 Apple pridal k práci s fotoaparátom funkciu **manuálneho zaostrovania** (*tap-to-focus*). Užívateľ pri snímaní objektov jednoduchým kliknutím zaostrí obraz na ním zvolený objekt. Táto funkcia si našla uplatnenie aj v našej aplikácii.

[1] iPod Touch 4. generácie disponuje TFT Retina displejom, operačnou pamäťou s veľkosťou 256MB a na jeho tele sa nachádza iba jeden fotoaparát, so slabším rozlíšením ako je to u iPhone

Kapitola 2

Vývoj aplikácií pre iPhone

2.1 Výber implementačného prostredia

2.1.1 Konkurenčné mobilné platformy

Google Android

Momentálne asi najväčší konkurent na poli s mobilným priemyslom, čo sa týka vstavaného operačného systému. Google Inc. predstavil niekoľko svojich telefónov, väčšinou ale tento systém nasadzuje do telefónov iných výrobcov. Tu sa črtá prakticky najväčšia nevýhodu vývoja, keďže vývojár dopredu presne nepozná samotný hardware, na ktorom bude jeho aplikácia inštalovaná a spúšťaná. Zariadenia s Android OS majú totiž rôznu veľkosť obrazovky, rôzny výkon hardwaru a v neposlednom rade sú všetky aplikácie vývojármi odosielané do Android Marketu[1] bez výrazného obmedzenia a kontroly zo strany Google. Výrazne tak stúpa riziko stiahnutia aplikácie, ktorá obsahuje nežiadúci a škodlivý zdrojový kód (vírusy). Operačný systém Android je poháňaný linuxovým jadrom, v súčasnosti nesie jeho najnovšia verzia pre mobilné telefóny kódové označenie *Gingerbread* (2.3). Aplikácie sú písané v jazyku Java.

Microsoft Windows Phone 7

Jedna z najnovších platforiem[2] pre "chytré" mobilné telefóny od spoločnosti Microsoft. Na trhu existuje iba niekoľko, dá sa povedať prototypov s týmto operačným systémom a dokumentácia spolu s vývojovými nástrojmi je v súčastnosti na veľmi slabej úrovni. To bol hlavný dôvod, prečo som od vývoja na tejto platforme už na začiatku ustúpil. Microsoft sa u tejto platformy spolieha na niekoľko základných prvkov, medzi inými sú to .NET[3], XNA[4], a štandard Web 2.0 [14].

Blackberry OS

Tento systém je rozšírený hlavne v USA, kam spoločnosť RIM telefóny Blackberry aj primárne cieli a distribuuje. Pre mňa teda nemalo zmysel púšťať sa do tvorby aplikácie pre

[1] Android Market - online obchod s aplikáciami pre OS Google Android.

[2] Microsoft kompletne preprogramoval pôvodný MS Windows Mobile 6.5 a MS Windows Phone 7 voči tejto platforme nie je ani spätne kompatibilný.

[3] .NET - súbor technológii a knižníc k vývoju software pre OS Microsoft Windows.

[4] XNA - vývojové prostredie pre tvorbu hier.

systém, v ktorom by mi nikto z môjho okolia nedokázal pomôcť a poskytnúť cenné rady a skúsenosti s vývojom na týchto zariadeniach.

Symbian OS

Operačný systém Symbian je tu s nami už niekoľko rokov, na vývoji sa podieľa spoločnosť Nokia a stretávame sa s ním už od počiatku éry smartphonov[5]. Momentálne je Symbian dostupný aj vo verzii pre dotykové telefóny. Od vývoja v ňom som upustil hlavne kvôli rýchlosti a značne veľkej a pomalej odozve samotného systému a aplikácii v ňom spúšťaných. Treba ale spomenúť, že táto platforma má prepracovanú dokumentáciu a vývoj v prostredí Qt Toolkit je na vysokej úrovni. Aplikácie sa programujú prevažne v jazykoch C++ alebo Java.

2.1.2 Prečo práve iPhone a iOS?

Rozhodol som sa pre vývoj aplikácie pre iPhone z 2 primárnych dôvodov.

1. Túto platformu poznám dosť dobre ako užívateľ, keďže Apple produkty používam už dlhší čas. S ostatnými platformami som mal možnosť prísť do styku len minimálne.

2. Druhým dôvodom je fakt, že iOS je operačný systém kvalitný, rýchly, čistý, založený na UNIXovom jadre. Vývojár dopredu presne vie, na akom prístroji bude jeho aplikácia spúšťaná a môže tak detailne testovať a debugovať aplikácie bez obáv, že by sa vyskytol hardware, kde by mohol nastať problém s kompatibilitou.

2.2 Spôsob vývoja iPhone aplikácií

Aplikácie pre iPhone je možné tvoriť v dvoch základných rovinách. Jedna z nich je tzv.

- **webová**, kde tvoríme program pomocou HTML 5 kódu a z telefónu k nej pristupujeme iba pomocou internetu cez vstavaný internetový prehliadač. Má to samozrejme svoje výhody aj nevýhody. Za výhodu by sa dalo považovať to, že takúto aplikáciu teoreticky spustíme na akomkoľvek zariadení, ktoré HTML 5 technológiu podporuje a nie je tak strikne viazaná iba na produkty spoločnosti Apple. Naopak, nevýhodou je nutnosť vlastniť dátový tarif, prípadne iný spôsob pripojenia k internetu (napr. cez WiFi).

Druhou možnosťou je vyvíjať aplikáciu ako

- **natívnu** pre danú platformu. Aplikácia je teda nasadená, nahratá a nainštalovaná do zariadenia priamo, odpadá teda nutnosť byť neustále online pripojený. V tejto práci sa budem zaoberať iba týmto spôsobom jej tvorby. Natívna aplikácia pre iPhone môže byť vyvíjaná iba v prostredí operačného systému Mac OS, ktorý spoločnosť Apple nasadzuje do všetkých svojich počítačov a MacBookov[6]. Súčasťou tohoto systému je možnosť využitia iPhone SDK spolu s integrovaným vývojovým prostredím (IDE) Xcode, ktorý v sebe obsahuje modifikovanú verziu GCC prekladača s podporou jazykov C, C++, Java, Python ale najmä Objective-C. Ten je totiž základným pilierom

[5] Smartphone je slangové označenie telefónov so zabudovaným vyšším operačným systémom.

[6] Macbook je notebook, ktorý produkuje Apple. Jeho pomenovanie vychádza z toho, že je poháňaný operačným systémom Mac OS

tvorby iPhone aplikácií. Pri vývoji sa využíva Cocoa framework, doplnený o špecifické vlastnosti pre mobilnú platformu. Vznikol tak Cocoa Touch a spolu s ním sa vývojárom otvorila brána do sveta k tvorbe iOS aplikácií.

2.3 iPhone SDK

2.3.1 Xcode

Vývojové prostredie Xcode je možné v najnovšej verzii[7] zakúpiť v online obchode s aplikáciami[8], prípadne staršiu verziu zdarma stiahnuť z webových stránok spoločnosti Apple. Pri spustení je možné zvoliť jeden z prednastavených projektov, teda či budeme vytvárať klasickú aplikáciu pre Mac OS alebo iOS aplikáciu. V rámci druhej možnosti je na výber šablóna pre iPhone i iPad. Xcode následne sám vytvorí základnú **adresárovú štruktúru**, programátor už iba upravuje a pridáva potrebné súbory (triedy, objekty, rozhrania, GUI) vo vstavanom editore. Xcode podporuje vytváranie **snapshotov**, chronologických verzií vytvoreného zdrojového kódu s možnosťou kedykoľvek vrátiť sa k ľubovoľnej z nich. Disponuje vstavanou dokumentáciou a automatickým doplňovaním zdrojového kódu (*code completion*).

Grafické užívateľské rozhranie (GUI) je možné programovať takisto iba pomocou kódu [5]. Druhou alternatívou je intuitívnejšie, o pár funkcií obmedzenenejšie prostredie nástroja Interface Builder. Xcode disponuje integrovaným **debuggerom**, pomocou ktorého je možné aplikácie ladiť ako v simulátore, tak aj priamo v reálnom zariadení. Pre debuggovanie a analýzu programov sa používa nástroj *Instruments*. Ten umožňuje programátorovi sledovať výkonnostné charakteristiky aplikácie v reálnom čase počas behu a zachytiť a odhaliť tak miesta, kde sa hromadí zbytočná neuvoľnená pamäť, prípadne sledovať a krokovať jednotlivé stavy viacvláknových aplikácií.

2.3.2 Interface Builder

Tento nástroj slúži k tvorbe grafického užívateľského rozhrania aplikácie jednoduchým *drag & drop* štýlom a následným napojením jednotlivých komponentov na ich funkčnú časť. Nevýhodou tvorby rozhrania v tomto nástroji je, že samotná aplikácia negeneruje zdrojový kod, ale prvky zapúzdruje do oddelených XIB súborov, ktoré musí programátor ručne do projektu začleňovať. Je v nich obsiahnuté všetko, od popisov objektov rozhraní až po súradnicové pozície rozložení jednotlivých komponent. Interface Builder (IB) vytvára **okná** (*windows*), **pohľady** (*views*) a ďalšie ovládacie komponenty (*tlačítka, taby, navigačné lišty*). Neoceniteľným pomocníkom je IB hlavne v momente, keď potrebujeme v rámci MVC architektúry spojiť pohľad (*View*) s riadením (*Controller*). Je to možné dosiahnuť jednoduchým skĺbením klávesových skratiek s gestami myši. Tento nástroj výrazne pomáha pri lokalizovaní aplikácie. Pre každú ďalšiu jazykovú mutáciu programátor vytvára samostatný XIB súbor, ktorý sa načíta automaticky pri zmene jazyka.

[7] Xcode 4.0 ku dňu 18.5.2011.

[8] Apple sa už nejaký čas snaží presadiť jednotný obchod s aplikáciami pre Mac OS s názvom Mac App Store, podobne ako je to momentálne s App Store - obchodom s mobilnými aplikáciami.

2.3.3 iPhone Simulator

Tento veľmi užitočný nástroj dovoľuje spúšťať a testovať vytvorené aplikácie v simulovanom prostredí počítača, bez nutnosti vlastniť vývojársky certifikát a nahrávať aplikáciu do reálneho prístroja. Je ale na mieste pripomenúť, že simulátor nezodpovedá realite 1:1, keďže počítač, na ktorom simulátor spúšťame, v sebe zahŕňa úplne inú, rádovo oveľa vyššiu konfiguráciu ako skutočný iPhone. Bolo to jednoznačne cítiť pri samotnom spracovaní obrazu v mojej aplikácii, kde simulátor zvládal náročnejšie procesy niekoľkonásobne rýchlejšie. Napriek tomu som aplikáciu v simulátore testoval často, keďže sa jedná o rýchlejší spôsob ako prostredníctvom pripojeného telefónu. Musel som sa však zaobísť bez funkcie vstavaného fotoaparátu, tá v simulátore totiž podporovaná nie je. Namiesto nej som využíval fotogalériu a vopred nahraté fotografie štátnych poznávacích značiek.

2.4 Cocoa Touch Framework

Jedná sa o API[9] pre programovanie iOS aplikácií. Vychádza z Cocoa API, je ale optimalizovaný pre dotykové, mobilné platformy. Všetky aplikácie majú pôvod vo frameworku UIKit, odkiaľ dedia základné objekty používané na zobrazovanie obsahu na displej, čítanie vstupu od užívateľa a správu udalostí. Od operačného systému aplikácia zachytáva správy od objektu UIApplication.

Základom celého frameworku a vytváraní aplikácií v ňom je návrhový vzor MVC (*Model- View-Controller*).

2.4.1 Model-View-Controller

Jedná sa návrhový vzor, softwarovú architektúru, kde je kód aplikácie striktne rozdelený do 3 vrstiev a modifikácia niektorej z nich má minimálny vplyv na ostatné.

1. **aplikačná a bussiness logika (*model*)**
   Je doménovo špecifická reprezentácia informácií, s ktorými aplikácia pracuje.

2. **prezentačná vrstva, pohľad (*view*)**
   Prevádza dáta reprezentované modelom do podoby vhodnej k interaktívnej prezentácii užívateľa. Vytvára užívateľské rozhranie.

3. **riadiaca časť (*controller*), ktorá spája aplikačnú a prezentačnú vrstvu**
   Reaguje na udalosti, typicky od užívateľa, a zaisťuje zmeny v modele alebo pohľade. Predstaviteľom je `UIViewController`.

[9] API - Application Programming Interface = rozhranie pre programovaciu aplikáciu.

2.5 Objective-C

Objective-C je jediný programovací jazyk, v ktorom je možné tvoriť natívne aplikácie pre iPhone [12]. Týmto jazykom je naprogramovaný aj samotný framework Cocoa Touch. Ide o objektovo orientovaný dynamický jazyk s pôvodom v jazykoch ANSI C a Smalltalk. Druhý menovaný je špecifický hlavne svojou syntaxou pri písaní objektového kódu, kde sa namiesto volaní metód posielajú tzv. správy inštanciám objektov. Aplikácie je možné písať aj v čistom C. Zdrojové implementačné súbory majú príponu *.m*, hlavičkové deklaračné *.h*. Objective-C prináša aj niekoľko nových dátových typov akými sú `SEL`, `id` a podobne. Pri vývoji aplikácií pre iPhone sa programátor nemôže spoliehať na vstavaný garbage collector[10], aj keď ho jazyk Objective-C v skutočnosti podporuje. Podpora však zostáva len pri vývoji Mac OS aplikácii a správa pamäti pri iOS aplikáciách je čisto v réžii vývojára.

2.5.1 Popis základných komponentov použitých v aplikácii

Pre zaujímavosť spomeniem niektoré odlišnosti Objective-C od ostatných objektovo orientovaných jazykov [4]:

Všetky metódy sú typu `public`. Neexistuje nič ako `private`, či `protected` metóda. Inštančné premenné sú pre zmenu všetky typu `protected`.

Rozlišujeme inštančné a triedne metódy. Deklarácie inštančných metód začínajú prefixom „`-`“ a sú volané odoslaním správy aktuálnej inštancii (objektu) triedy. Metódy triedy nepotrebujú pre svoje fungovanie vytvorenú inštanciu objektu. Deklarácia začína prefixom „`+`“.

Špecifický tvar odosielania správ objektom. Špecifický je v tom, že miesto guľatých zátvoriek a bodkovej notácie[11] sa používajú hranaté zátvorky. Na ľavej strane je objekt, ktorému správu odosielame, za ním nasleduje správa zložená z kľúčového slova a parametrov.

Príklad: `[Object setParameter1:X parameter2:Y]`.

Manažovaná pamäť. Objective-C implicitne automatickú správu pamäti obsahuje, iPhone vývojár je ale ochudobnený o garbage collector a musí sa spoľahnúť na tzv. manažovanú pamäť. Aplikácia sa sama musí postarať o to, aby bola všetka pamäť správne uvoľňovaná vzhľadom na veľmi obmedzenú pamäť akou iPhone disponuje.

NSAutoreleasePool. Je to trieda, ktorá riadi uvoľnovanie istých zdrojov v pamäti sama. Konkrétne tých, na ktoré odkazuje aspoň jedna referencia. Každé vlákno má svoj vlastný `AutoreleasePool`.

Kategórie. Jedná sa o možnosť rozšírenia schopností triedy o nové metódy. Je možné rozširovať dokonca triedy, ku ktorým nemáme priamy prístup do zdrojového kódu. Takto rozšírené triedy je možné používať iba v rámci nášho programu.

Protokoly. Jazyk Objective-C nedisponuje viacnásobnou dedičnosťou tried. Namiesto toho sa používajú protokoly - zoznamy metód, ktoré sú buď vyžadované alebo voliteľné pri dedení triedou.

[10] Garbage collector - automatická správa pamäti, známa napr. z jazyka Java.
[11] Bodková notácia je používaná napr. v jazyku Java.

Hlavné použité komponenty:

- `UIView` - základná, abstraktná trieda pre zobrazenie obsahu na obrazovke.
- `UIViewController` - riadiaca časť inštancií objektov zdedených od triedy `UIView`.
- `UIImageView` - podtrieda triedy `UIView`, má podporu pre zobrazenia obrázkov (`UIImage`).
- `UIScrollView` - zobrazuje obsah, ktorý je väčší ako okno obrazovky.
- `UIActivityIndicatorView` - indikátor aktivity aplikácie
- `UINavigationController` - vytvára hierarchickú štruktúru jednotlivých pohľadov (obrazoviek) aplikácie. Objekt na vrchole zásobníka je aktuálne zobrazený na displeji.
- `UIImage` - zobrazenie dát obrázka
- `CGImage` - bitmapová reprezentácia obrázka
- `UIButton`, `UILabel` - tlačítko, štítok
- `NSString`, `NSMutableString` - reťazec, reťazec s možnosťou úpravy

Štandardné metódy a správy:

- `- (void)viewDidLoad` - metóda sa zavolá po načítaní pohľadu
- `- (void)viewDidUnload` - metóda sa zavolá po zatvorení pohľadu
- `- (void)didReceiveMemoryWarning` - metóda sa zavolá pri nedostatku pamäti
- `- (void)dealloc` - metóda, ktorá odošle správu objektu po uvoľnení z pamäte
- `- (BOOL)application:(UIApplication *) didFinishLaunchingWithOptions: (NSDictionary *)` - metóda, ktorá sa zavolá po spustení aplikácie
- `- (void)applicationWillTerminate:(UIApplication *)` - metóda, ktorá sa zavolá pred vypnutím aplikácie. Slúži obvykle pre uloženie aktuálneho stavu aplikácie pre potreby ďalšieho spustenia.

2.5.2 Práca s fotoaparátom a galériou obrázkov

Pri práci so vstavaným fotoaparátom (príp. videokamerou) a galériou fotografií sa používa trieda `UIImagePickerController`. V nej nastavíme zdroj buď na fotogalériu v pamäti

```
source. = UIImagePickerControllerSourceTypePhotoLibrary
```

alebo na fotoaparát

```
source. = UIImagePickerControllerSourceTypeCamera.
```

2.6 Schvaľovanie a distribúcia aplikácií

Aby sme mohli nami vytvorenú aplikáciu otestovať na reálnom prístroji, je potrebné v prvom rade vlastniť certifikát vývojára od spoločnosti Apple a aplikáciu ním elektronicky podpísať. Certifikát je možné získať za minimálny poplatok 99 dolárov a po dobu jedného roka od jeho vystavenia má vývojár možnosť svoju aplikáciu nasadiť do svojho telefónu, ktorý ma zaregistrovaný k svojmu účtu (maximálne 100 zariadení). Okrem toho má nárok svoju aplikáciu odoslať do *AppStore* - obchodu s aplikáciami a legálne ju predávať ostatným užívateľom. To je však podmienené akýmsi schvaľovacím procesom samotného Apple, ktorý je v niektorých prípadoch dlhotrvajúci a riadi sa vnútornými podmienkami a politikou firmy. Licenciu ako takú je možné získať aj v rámci univerzity (iOS University Developer Account), ktorá dovoľuje študentom, ktorí sú členmi tohoto programu, zdarma vyvíjať a distribuovať aplikáciu pre svoje vlastné použitie. Naša fakulta v čase písania tejto práce univerzitným účtom nedisponovala, aplikácia tak bola vyvíjaná prostredníctvom osobného účtu autora práce.

Kapitola 3

Štátne poznávacie značky

V tejto práci sa zameriam z väčšej časti na štátne poznávacie značky (ŠPZ) Slovenskej republiky, pomimo spomeniem aj špecifiká českých značiek všeobecne. Voľba práve na slovenské značky padla hlavne preto, že počas písania tejto práce a vývoja aplikácie som sa pohyboval prevažne na území Slovenska. Ľahšie tak prebiehalo testovanie a experimentovanie. Druhým dôvodom bolo pre mňa ako slovenského rodáka, väčší prehľad práve u týchto tzv. „domácich“ značiek. Len pre úplnosť spomeniem, že od 1.1.2005 nahradil pojem „štátna poznávacia značka“ iný, novší - „evidenčné číslo vozidla“ (EČV). V tejto práci budem i naďalej používať univerzálnu všeobecnú skratku ŠPZ.

3.1 Integritné obmedzenia slovenských ŠPZ

Evidenčné číslo vozidla (EČV, ŠPZ) tvoria písmená a čislice. Prvá dvojica písmen evidenčného čísla označuje okres, kde je automobil evidovaný, za ňou nasleduje trojica číslic a dvojica písmen.

Štandardne má slovenská ŠPZ tvar (obr. 3.1): **LL-NNNLL** [17], kde **L** označuje písmeno (ang. *letter*) a **N** číslo (ang. *number*).

Obrázok 3.1: Tvar súčasnej slovenskej štátnej poznávacej značky automobilu

Poznávacia značka je na automobile pripevnená spredu aj zozadu a má jednu z troch predpísaných veľkostí obdĺžnikového tvaru. Klasická pre osobné a terénne automobily má rozmery 52 x 11 cm a menej tradičná, využívajúca sa hlavne na automobiloch dovezených zo zahraničia a poľnohospodárskych strojoch má veľkosť 34 x 20 cm. Najmenej často používanou a zároveň najmenšou (24 x 15 cm) je značka, ktorá si svoje uplatnenie našla na prípojných vozíkoch a motocykloch (obr. 3.2).

Obrázok 3.2: ŠPZ motocyklov a prípojných vozíkov

Značku, ako už bolo hore spomínané, tvoria typicky čierne písmená a čísla, reliéfovito vsadené na bielom podklade. Na ľavej strane značky je umiestnený znak Slovenskej republiky, od roku 2006 je to modré pole s 12 hviezdami usporiadných v kruhu - jeden zo symbolov Európskej únie, nad písmenami „SK“. Znak Slovenska nahradil znak pomlčky v pôvodnom formáte spred roka 2006 (obr. 3.3). S touto novelou zákona k nám prišla aj zmena v grafickom znázornení písmena O, ktoré sa pre lepšiu prehľadnosť odlišuje od číslice 0 (nula) prerušením čiary v pravom hornom rohu znaku. Súčasťou starších ŠPZ môže byť aj nálepka potvrdzujúca, že auto prešlo technickou alebo emisnou kontrolou.

Moja aplikácia rozpoznáva staré aj nové tvary slovenských ŠPZ s minimálnymi obmedzeniami, ktoré budú bližsie popísané v časti s implementáciou.

Obrázok 3.3: Tvar starej slovenskej štátnej poznávacej značky

Evidenčné čísla vozidiel cudzích zastupiteľských úradov na území Slovenskej republiky tvoria písmená EE a za nimi päť číslic. Vozidlá zastupiteľstiev zriadených diplomatickou misiou tvorí dvojica písmen ZZ a päť číslic. Ďalšie typy evidenčných čísiel, ktoré ale pre potreby tejto aplikácie nie je potrebné ďalej rozoberať, z toho hľadiska, že ich rozpoznávanie nie je podporované, sú:

- zvláštne evidenčné číslo u novovyrobeného neevidovaného vozidla;
- zvláštne evidenčné číslo u historického vozidla;
- zvláštne evidenčné číslo športového vozidla.

Extra skupinu tvoria tzv. špeciálne ŠPZ (obr. 3.4), ktoré na žiadosť držiteľa vozidla alebo vlastníka vozidla možno prideliť. V nich sa namiesto písmen a číslic za štátnym znakom Slovenskej republiky uvádzajú

- písmená na prvom až piatom mieste (napr. NZ-AAAAA);

- písmená na prvom až štvrtom mieste a číslica na piatom mieste (napr. NZ-AAAA1) alebo
- písmená na prvom až treťom mieste a kombinácia číslic na štvrtom a piatom mieste (napr. NZ-AAA11).

(a) Špeciálna ŠPZ č.1

(b) Špeciálna ŠPZ č.2

Obrázok 3.4: Špeciálne ŠPZ

Väčšinou sa potom jedná o slová naviazané na prvé dve písmená označujúce okresné mesto (obr. 3.4(a), obr. 3.4(b)), prípadne o niekoľko rovnakých číslic za sebou. Je zakázané použiť písmená X a O na prvom mieste za štátnym znakom a skladať výrazy a slová urážlivé, hanlivé alebo pohoršujúce spoločnosť. Moja aplikácia takéto značky nerozpozná celé, pretože dokáže spracovať len ŠPZ v pevne definovanom tvare, aký je popísaný vyššie v tejto práci.

3.2 Využitie rozpoznania ŠPZ

Rozpoznávanie poznávacej značky automobilu našlo uplatnenie v rôznych oblastiach, do mnohých sa postupne pomaly ale isto presadzuje. Databázou ŠPZ disponujú hlavne štátne zložky (polícia) a preto snímanie značiek našlo uplatnenie pri dokazovaní dopravných priestupkov (prejazd na červenú, zákaz zastavenia, ...), pri autentizácii v parkovacích domoch, pri overovaní totožnosti automobilu v logistických firmách (nakladanie a vykladanie tovaru), prípadne overenie totožnosti majiteľa vozidla pri vstupe do firiem a podnikov. V zahraničí už existujú mýtne systémy, ktoré na základe značky majiteľovi vyúčtujú poplatok za užívanie cestných komunikácií.

3.3 Značky v Českej republike

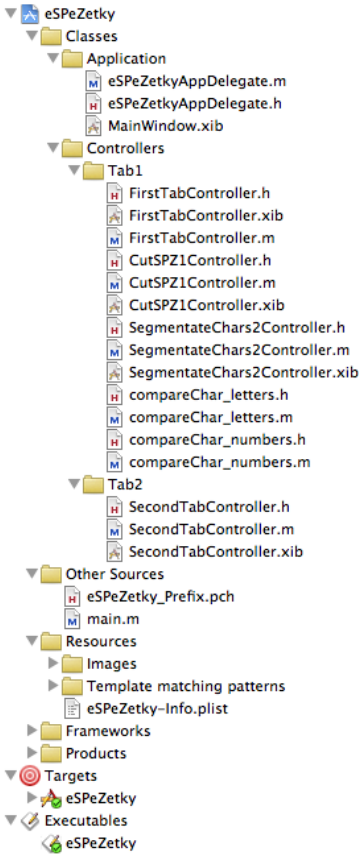
Registračná značka (RZ) je oficiálny názov pre štátnu poznávaciu značku v Českej republike. Tvarom sú podobné slovenským, majú však odlišné integritné obmedzenia. Môžu obsahovať 5 až 7 znakov, pričom aspoň jeden z nich musí byť písmeno a jeden čislica. Písmeno najviac vľavo je kód kraja. Od 1. 5. 2004 majú na ľavej strane rovnaký modrý EÚ pás s 12 hviezdami, podobne ako slovenské. V Českej republike sú osobné automobily od tých špeciálnych (diplomatických, vojenských, záchranných, ...) odlíšené rôznou kombináciou farby písma a pozadia. Moja aplikácia rozpoznávanie týchto značiek nepodporuje.

Kapitola 4

Návrh aplikácie

4.1 Objektový návrh

Aplikácia je postavená na čisto objektovom návrhu, ktorý je mierne špecifický, prispôsobený pre moju iPhone aplikáciu (obr. 4.1 a Príloha B). Pracujem s triedami a objektmi, z ktorých každý jeden predstavuje samostatný pohľad (*view*) - obrazovku a v rámci nich deklarujem identifikátory, ktoré k nim prislúchajú a implementujem metódy, ktoré sa vykonávajú. Najpodstatnejšie sú tri základné pohľady, riadené triedami `FirstTabController`, `CutSPZ1Controller` a `SegmentateChars2Controller` zdedené od triedy `UIViewController`. Tú istú triedu potom dedí aj trieda `SecondTabController`, o ktorej sa zmienim nižšie.



Obrázok 4.1: Adresárová štruktúra aplikácie

`FirstTabController` a `SecondTabController` predstavujú riadiace kontroléry pre dve triedy a zároveň objekty a pohľady, ktoré sa načítajú po stlačení jedného z dvoch tlačidiel v `UINavigationBar`e v spodnej časti aplikácie. Táto navigačná lišta je implementovaná iba v Interface Builderi a programovo sa k nej pristúpiť nedá (odntuje ju prvý kontrolér od triedy `UINavigationControllerDelegate`). `FirstTabController` predstavuje pohľad - obrazovku, ktorá sa zobrazí ako prvá po spustení aplikácie (po splash screene[1]) (obr. 4.4(a)). Implementuje základné prvky, akým je `imageView`, ktorý zobrazí načítaný obrázok z galérie alebo odofotografovanú snímku z fotoaparátu, tri tlačítka a obsahuje pomocný pohľad s identifikátorom aktivity, zobrazený pri spracovávaní tohoto pohľadu a prechodu na ďalší. Obsahuje základné metódy potrebné pre prvotné spracovanie

[1]Splash screen je obrazovka s úvodným logom aplikácie, ktorá sa zobrazí 2 sekundy od spustenia programu.

vstupného obrazu - prevod na šedotónový obraz, čiernobiely obraz prahovaním, vertikálnu a horizontálnu detekciu hrán a prácu s *pickerom*, teda vstavaným fotoaparátom (za pomoci triedy `UIImagePickerControllerDelegate`). Pre prácu s obrázkom a jednotlivými jeho pixelmi využívam mnou vytvorenú štruktúru `SPixels`, ktorá je deklarovaná nasledovne:

Pseudokód 4.1: Deklarácia štruktúry SPixels

```
typedef struct {
            int r; // red color channel
            int g; // green color channel
            int b; // blue color channel
            int a; // alpha channel
} SPixels;
```

Je tu obsiahnutý ešte jeden špeciálne upravený pohľad, ktorý sa zobrazuje pri snímaní fotoaparátom ako pomocný layout. Metódy s prefixom `imageProcess_SPZdetection_` implementujú automatickú lokalizáciu riadka a stĺpca, kde sa ŠPZ pravdepodobne nachádza a postupuje tento údaj ďalšiemu, v poradí druhému kontroléru.

`CutSPZ1Controller` je kontrolérom pre druhý pohľad (obr. 4.4(b)). Jeho dominantnými prvkami sú najmä ovládacie prvky (ich obsluhu zabezpečuje metóda `buttonArrow:sender:`) pre posun obrazu v tzv. *Scroll View* - pohľade, ktorého obsah je možné približovať, oddaľovať a posúvať vo všetkých smeroch multidotykovými gestami. Táto trieda svojimi metódami dokáže lokalizovanú ŠPZ vyrezať do pevne stanovených rozmerov a metódou `cutChar:image :col:` nasekať jednotlivé znaky.

`SegmentateChars2Controller` je tretí a zároveň posledný základný kontrolér pre zobrazovanie tretieho pohľadu (obr. 4.4(c), obr. 4.4(d)). Obsahuje sedem `UIImageView` prvkov pre separované znaky poznávacej značky, ktoré metódou `toBW:image:` prahuje dynamickým prahom a následne klasifikuje každý obraz jednému, najviac podobnému vzorovému znaku. Dáta týchto vzorov má aplikácia uložené v samostatných súboroch. Túto triedu rozširujú dve tzv. kategórie, ktoré slúžia prakticky len pre prehľadnosť zdrojového kódu a istú ucelenú časť triedy tak oddeľujú do samostatných súborov. Sú nimi `compareChar_letters` a `compareChar_numbers`. Táto trieda tiež implementuje metódy pre zápis výsledných rozpoznaných ASCII znakov do vopred pripravených labelov.

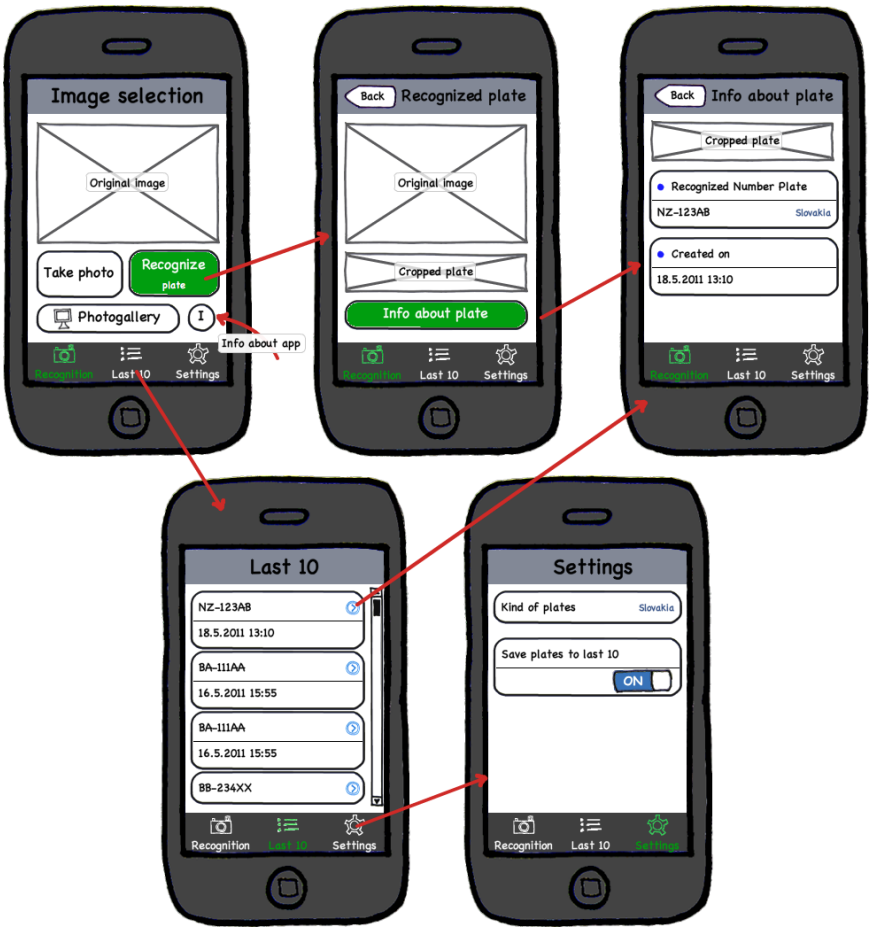
`SecondTabController` je kontrolér, ktorý zabezpečuje zobrazenie pohľadu s informáciami o aplikácii. Spolu s kontrolérom `CutSPZ1Controller` implementujú `UIScrollView` z triedy `UIScrollViewDelegate`. `eSPeZetkyAppDelegate` je špeciálna trieda, ktorá riadi beh celej aplikácie, konkrétne čo sa má stať po jej spustení a tesne pred ukončením. Dedí od hierarchicky najvyššie postavenej triedy `NSObject`.

4.2 Užívateľské rozhranie

Je potrebné si uvedomiť, že mobilné aplikácie majú špecifické podmienky pri ich vývoji. Prostredie mobilných aplikácií sa navrhuje s prihliadnutím k tomu, že ich typická doba používania sa odhaduje na maximálne niekoľko minút, prevažne niekde v teréne. Rozhranie preto musí byť čo najjednoduchšie, intuitívne a ideálne s natívnymi ovládacími prvkami korešpondujúcimi s danou platformou. Úspech aplikácie nespočíva v kvantite častokrát zbytočných funkcií ale v kvalite jednej či zopár funkcií, k čomu je primárne určená. Pri takomto koncepte sa výrazne zvyšuje bezpečnosť celej aplikácie.

4.2.1 Prvotný mockup

Kvôli lepšej orientácii v aplikácii, už na začiatku jej vývoja som si vyrobil v prostredí softwaru[2] pre tvorbu návrhu dizajnu prvotný mockup[3] programu (obr. 4.2) a načrtol tak možné usporiadanie ovládacích prvkov. Toto prostredie som otestoval na vzorke niekoľkých ľudí a na základe toho doplnil, prípadne zrušil zbytočné možnosti a prebytočné prvky.



Obrázok 4.2: Mockup aplikácie

[2]Balsamiq Mockups - program pre tvorbu návrhu dizajnu aplikácie.

[3]Termíny mockup a wireframe označujú návrhovú skicu výsledného programu, ktorá môže ale nemusí mať interaktívne prvky.

4.2.2 Finálny vzhľad

Po spustení aplikácie sa užívateľovi zobrazí úvodná stránka s jej logom a názvom. Užívateľské rozhranie je rozdelené do niekoľkých obrazoviek, tzv. pohľadov (views), medzi ktorými sa uživateľ sekvenčne pohybuje. Tento spôsob som zvolil hlavne preto, že jednotlivé kroky názorne demonštrujú priebeh celého rozpoznávania a celkovú funkcionalitu aplikácie. Zároveň je užívateľ oboznámený s každým úspechom či neúspechom v priebehu rozpoznávania a môže do neho dokonca zasahovať, vstupovať a rozhodovať o tom, či algoritmus správne detekoval to, čo mal.

Obrázok 4.3: Ukážka aplikácie - pohľad prvej obrazovky Snímanie ŠPZ

Prvá obrazovka, ktorá sa zobrazí ihneď po spustení aplikácie a úvodnom splash screene je zameraná na voľbu spôsobu výberu poznávacej značky. Užívateľ si zvolí buď ako zdroj fotogalériu, odkiaľ následne vyberie obrázok, na ktorom chce značku detekovať, alebo má možnosť priamo odfotiť novú ŠPZ. Pri fotení novej ŠPZ sa užívateľovi priamo v hľadáčiku fotoaparátu na obrazovke objaví pomocný šedobiely obdĺžnik, ktorý slúži ako pomocný navigátor, kde by sa mala ŠPZ správne nachádzať, aby bola aplikáciou rozpoznaná s väčšou pravdepodobnosťou (obr. 4.4(a)). V oboch režimoch, fotení aj výberu z fotogalérie, môže užívateľ v editačnom móde fotografiu priblížiť či posunúť. V podstate ak užívateľ dodrží veľkosť značky zodpovedajúcej pomocnému obdĺžniku, je v ďalších krokoch nutná iba minimálna, ak vôbec nejaká interakcia zo strany užívateľa.

Keď užívateľ fotografiu vyberie, zobrazí sa mu náhľad a aktivuje sa tlačítko pre ďalší proces rozpoznávania (obr. 4.3). Po stlačení sa fokus presunie na ďalšiu, druhú obrazovku programu.

Druhá obrazovka je primárne zameraná na to, aby užívateľa informovala o tom, či poznávacia značka bola vo fotografii detekovaná alebo nie. V prípade, že nie, zobrazí sa hláška o neúspechu a je možné detekciu zopakovať, prípadne zvoliť iný zdrojový obrázok. Ak však značka bola detekovaná, zobrazí sa jej náhľad v hornej časti obrazovky a je na vlastnom rozhodnutí užívateľa, či stlačí tlačítko pre pokračovanie alebo gestami pozíciu značky doladí (obr. 4.4(b)). Je možné využívať posúvanie obrazu ťahom jedným prstom a približovanie i odďalovanie značky tzv. *pinch-zoom*om použitím multigesta dvomi prstami naraz. Keď je potrebné značku doladiť naozaj minimálne, je výhodné využiť tlačítka ovládacích prvkov, ktoré poskytujú prakticky tú istú, pre človeka presnejšiu funkcionalitu ako gestá prstami. Ako pomôcka pre čo najpresnejšie polohovanie značky do vopred pripraveného výrezu slúži 7 vertikálnych čiar, ktoré symbolizujú časti značky, kde sa nachádza hranica medzi jednotlivými znakmi. Keď je značka na správnom mieste, užívateľ potvrdí voľbu stlačením zeleného tlačítka.

Ak tak spraví, je presunutý na tretiu, poslednú obrazovku zameranú na segementáciu - - oddelenie jednotlivých znakov v značke (obr. 4.4(c)). Opäť, ak je užívateľovi okom zrejmé, že segmentácia prebehla v poriadku a znaky - písmená a čísla sú dobre oddelené a čitateľné, môže pristúpiť k stlačeniu posledného tlačidla - rozpoznaniu znakov.

Rozpoznanie jednotlivých písmen a číslic sa už deje na rovnakej, tretej obrazovke (obr. 4.4(d)). Môže nastať situácia, že sa nepodarí rozpoznať úplne všetky znaky. Pod každý jeden rastrový, oprahovaný znak je zobrazený jeho rozpoznaný ASCII znak a percento pravdepodobnosti, ktorým bol identifikovaný. Znaky, ktoré sa podarilo identifikovať, ale percento pravdepodobnosti toho, že sa jedná o daný znak, je nižsie ako 70%, aplikácia tento znak vyhodnotí za neklasifikovaný. Toto číslo - hranica medzi rozpoznaným a nesprávne rozpoznaným alebo nerozpoznaným znakom, bolo určené na základe optického testovania. Znaky, ktorých percentuálne vyjadrenie zhody so vzorovým znakom bolo pod hranicou 70%, boli vo veľkej väčšine rozpoznané a klasifikované nesprávne.

(a) Režim fotoaparátu (b) Ladenie pozície ŠPZ (c) Zatiaľ nerozpoznaná ŠPZ (d) Úspešne rozpoznaná ŠPZ

Obrázok 4.4: Ukážka ďalších obrazoviek aplikácie

Kapitola 5

Metódy rozpoznávania značky

5.1 Existujúce riešenia

K dnešnému dňu som nenašiel na trhu mobilných aplikácií, dostupných pre širokú verejnosť, jediný software, ktorý by primárne slúžil rovnakému účelu ako môj program. Existuje ale mnoho riešení a hotových aplikácií pre iOS, ktoré umožňujú rozpoznanie textu (OCR), či lokalizáciu predmetov a ďalšiu manipuláciu s nimi. Ak ale berieme do úvahy software pre osobné počítače, tam je to o niečo pestrejšie. Za všetkých spomeniem firmu Eyedea Recognition s.r.o. z Prahy, ktorá okrem iného ponúka software k detekcii a rozpoznaniu ŠPZ pre statické aj mobilné kamery. Využíva moderné algoritmy počítačového videnia a samoučiace sa algoritmy, ktoré sa dokážu prispôsobiť konkrétnym podmienkam (kamere, výkonu CPU, svetlu, typu značky) [6].

5.2 Vlastné riešenie

Moja aplikácia nerozpoznáva značky plne automaticky. Znamená to, že ihneď po odfotení automobilu so značkou užívateľ nedostáva konečný výstup v podobe textovej reprezentácie znakov ŠPZ. Bola zvolená poloautomatická metóda rozpoznávania z dvoch dôvodov.

Prvým je prehladnosť a možnosť demonštrovať funkčnosť celej aplikácie krok po kroku.

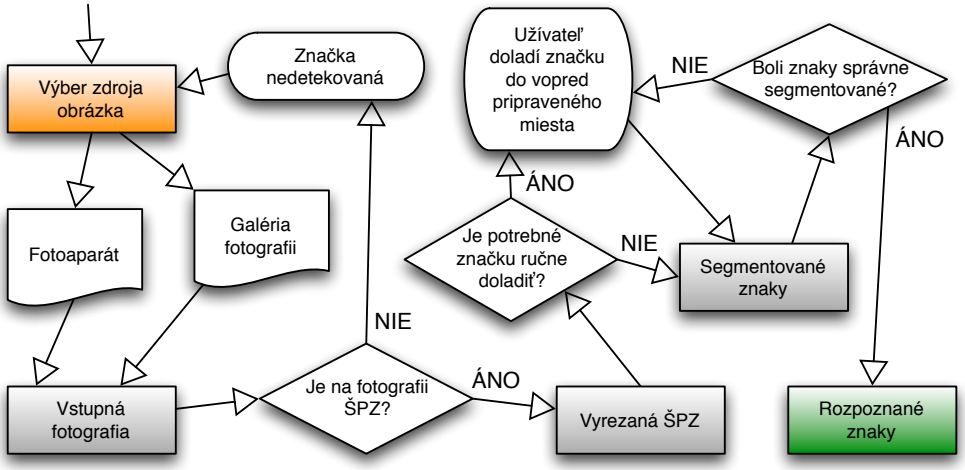
Druhým dôvodom je, že je požadovaná aspoň minimálna interakcia zo strany užívateľa a celý proces rozpoznávania ŠPZ tak pozostáva zo 4 krokov[1] (obr. 5.1):

1. Fáza lokalizovania ŠPZ v obraze
2. Fáza segmentácie znakov na značke
3. Fáza odstraňovania nečistôt a objektov mimo znakov
4. Fáza klasifikácia znakov v ŠPZ

Kedže mojou úlohou nebolo vytvoriť aplikáciu, ktorá plne automaticky rozpozná ŠPZ v obraze a mobilným telefónom je možné do veľkej miery prispôsobiť fotený objekt, vstupom

[1] V mojom prípade rozpoznávanie prebieha v 3 krokoch, bez fázy čistenia nečistôt mimo znakov.

pre vstavaný fotoapárát je predná alebo zadná strana automobilu s dobre viditeľnou značkou. Stupeň vychýlenia značky od jej horizontálnej polohy je stanovených na maximálne 2°, stupeň rotácie značky okolo osí X a Y na 5° (obr. 6.2.2).



Obrázok 5.1: Vývojový diagram aplikácie

5.3 Fáza lokalizovania ŠPZ v obraze

V tejto časti je hlavným cieľom správne identifikovať tú oblasť fotografie, kde sa s najväčšou (ideálne s určitou) pravdepodobnosťou ŠPZ nachádza. Pre nás ľudí je táto fáza jednou z najľahších, strojové rozpoznávanie je ale o dosť náročnejšie.

5.3.1 Prevod do tónu šedej a prahovanie celej fotografie

Vstupom pre túto fázu procesu rozpoznávania je šedotónová (*greyscale*) fotografia vytvorená z pôvodného plnefarebného obrazu (*originálu*) s rozmermi 640 x 640 pixelov. S hodnotami odtieňa šedej sa nám totiž bude pracovať oveľa ľahšie a presnejšie. Každý pixel obrázka triedy `CGImage` je reprezentovaný tromi farebnými kanálmi, zložkami - červenou, zelenou a modrou.

Keďže ľudské oko je na každú z týchto troch farebných zložiek citlivé inak[2], vzorec pre prevod farebného obrázka na *greyscale* má nasledovnú podobu [20]:

Pseudokód 5.1: Výpočet hodnoty pixela pri prevode na odtieň šedej

```
output_color_of_pixel = red_color*0.3 + green_color*0.59+ blue_color*0.11;
```

[2]Ľudské oko je najmenej citlivé na modrú farbu, najviac na zelenú.

Nasleduje prevod obrázka do čiernobielej varianty. To sa deje prostredníctvom prahovania (*thresholding*) vstupného obrazu. Vstupom je tentokrát šedotónový obraz a na každý jeho pixel aplikujeme predpis:

$$f(x) = \begin{cases} 0 & \text{if } x < threshold \\ 255 & \text{else} \end{cases}$$

kde

- **x** je vstupná hodnota ľubovoľnej z 3 farebných zložiek pixela;
- **f(x)** je funkcia pre výpočet novej hodnoty pixela;
- **threshold** je nastavený prah, v tomto prípade je to napevno hodnota 127[3];
- konštanty **0** (*čierna*) a **255** (*biela*) sú nové výsledné hodnoty pixela.

5.3.2 Detekcie hrán

Ďalší krokom je tzv. detekcia hrán (*edge detection*) - horizontálna, následne vertikálna. Vertikálna detekcia hrán spočíva v tom, že každé dva susediace pixely v stĺpcoch medzi sebou porovnáme a ak zistíme, že sú značne odlišné, nahradíme porovnávaný prvý pixel čiernym, druhý naopak bielym pixelom.

Obrázok 5.2: Aplikovanie detekcie hrán na obrázok automobilu

Spomínaná odlišnosť je daná nasledujúcim vzorcom:

$$f(x,y) = \begin{cases} 0 & ((\text{if } x < limit) \text{ AND } (\text{if } y > limit)) \\ & \text{OR } ((\text{if } x > limit) \text{ AND } (\text{if } y < limit)) \\ 255 & \text{else} \end{cases}$$

kde

- **x** je vstupná hodnota ľubovoľnej z 3 farebných zložiek prvého porovnávaného pixela;
- **y** je vstupná hodnota ľubovoľnej z 3 farebných zložiek druhého porovnávaného pixela;
- **f(x, y)** je funkcia pre výpočet novej hodnoty prvého pixela;
- **limit** je nastavená hranica, v tomto prípade je to napevno hodnota 110[4];
- konštanty **0** (*čierna*) a **255** (*biela*) sú nové výsledné hodnoty prvého porovnávaného pixela.

Takýmto spôsobom prechádzame obrazom najprv zľava doprava pri horizontálnej detekcii hrán, potom zhora dole pri vertikálnej. Výsledné dva obrazy spojíme do jedného (obr. 5.2). V takomto obraze hľadáme kandidátnu oblasť, kde je v horizontánom smere aspoň 400 pixelov a vo vertikálnom smere minimálne 100 pixelov neprerušovane za sebou. Tieto hodnoty boli testovaním namerané ako minimálna predispozícia pre lokalizovanie značky v obraze, kde má značka už relatívne malé rozmery.

[3]Bola zvolená stredná hodnota 127, pretože dynamicky počítaný prah v tomto prípade nevykazoval obstojné výsledky.

[4]Testovaním bola zvolená hodnota 110, pretože vykazovala najobjektívnejšie výsledky.

Pseudokód 5.2: Lokalizácia značky v obraze

```
converting input image to greyscale;
converting greyscale image to black and white;
vertical edge detection;
horizontal edge detection;
merging images with vertical and horicontal edge detection

searching row where number plate can be situated
black_pixels = 0;
for (all columns) {
    for (all rows - minimal height of number plate) {
        black_pixels = 0;
        if pixel is black one {
            for (tolerance rows near by row that is examining) {
                if (pixel is black one)
                    black_pixels++;
                if (black_pixels == minimal height of number plate)
                    return ROW;
            }
        }
    }
}

searching col where number plate can be situated (according to found row)
black_pixels = 0;
for (all columns in found row) {
    black_pixels = 0;
    if pixel is black one {
        for (tolerance rows near by row that is examining) {
            if (pixel is black one)
                black_pixels++;
            if (black_pixels == minimal width of number plate)
                return COL;
        }
    }
}

cut number plate (row, col, width, height);
```

5.4 Fáza segmentácie znakov v značke

V tomto momente máme z predchádzajúcej fázy výrez fotografie, na ktorom sa nachádza ŠPZ. Ideálne by ďalším krokom bolo vytvorenie stĺpcového profilu (histogramu) a tým zistenie hustoty čiernych bodov v jednotlivých stĺpcoch značky. V miestach, kde by bol tento profil najmenší, by sa nachádzala medzera medzi znakmi a práve tu by sme značku „nastrihali“ z celku na jednotlivé znaky.

Mnou zvolená metóda je o dosť jednoduchšia a dovolil som si ju zvoliť najmä preto, že výrez značky je v mojom prípade konštantných rozmerov a každý znak sa v rámci značky nachádza vždy na rovnakej pozícii. Všetky znaky máju teda presne určené začiatočné a koncové súradnice a rovnakú veľkosť 60 x 90 pixelov. Explicitne tak nedetekujem medzery medzi nimi. Výstupom je 7 samostatných menších obrázkov, pričom každý reprezentuje práve jeden zo 7 znakov.

Pseudokód 5.3: Segmentácia znakov v značke

```
cut character (7 times) :
for (rows where number plate is located) {
    for (columns where number plate and the character is located) {
        pixels working and copying
    }
}
```

5.5 Fáza odstraňovania nečistôt a objektov mimo znakov

Na úvod opäť pripomeniem, že túto fázu rozpoznávania nemám vôbec implementovanú kvôli tomu, že v časti so segmentáciou znakov je možné značku a jednotlivé znaky presne napozíciovať do pre ne vopred vyhradených priestorov. Táto fáza je pre nás praticky zbytočná a popíšem ju len v teoretickej rovine. Zvyčajne je táto fáza posledná pred samotným rozpoznávaním znakov. Hovorí sa jej tiež *preprocessing*. Prispieva k presnejšej klasifikácii znakov vďaka tomu, že z okolia písmen odstráni zbytočné oblasti čiernych pixelov, ktoré nie sú súčasťou znakov.

V prvom rade musíme všetky dielčie obrazy previesť prahovaním na čiernobiele. Odstránime riadky a stĺpce, ktoré obsahujú iba čierne pixely, nakoľko sa s určitosťou jedná o okraj samotnej značky. Ďalej odstránime úplne malé samostatné skupiny pixelov - jedná sa o skrutky, špinavé fľaky, atď. Tieto zmeny aplikujeme na pôvodné šedotónové výrezy znakov s tým, že biele miesta nahradíme farbou pozadia.

5.6 Fáza klasifikácie znakov v ŠPZ

Segmentované znaky podrobíme opäť thresholdingu - prevodu do čiernobielej verzie, tentokrát bez medzikroku v podobe prevodu na stupeň šedi. Nie je to nutné, pretože samotné znaky neobsahujú žiadne pixely, ktoré by nás zaujímali kvôli svojej farbe. Na druhej strane, pri thresholdingu tentokrát využijeme **dynamický prah**. Ten spočítame tak, že súčet hodnôt všetkých pixelov vydelíme počtom pixelov. Vysvetľuje to nasledovný pseudokód:

Pseudokód 5.4: Výpočet dynamického prahu

```
for each pixel {
  output_color_of_pixel = red_color*0.3 + green_color*0.59+ blue_color*0.11;
  sum = sum + output_color_of_pixel;
}
threshold = sum / count_of_pixels;
```

kde

- `output_color_of_pixel` je hodnota pixela spriemerovaná na základe jej 3 farebných zložiek;
- `sum` je súčet zatiaľ načítaných spriemerovaných hodnôt pixelov;
- `count_of_pixels` obsahuje počet všetkých pixelov obrazu;
- `threshold` je hodnota dynamicky vypočitaného prahu.

5.6.1 Template matching

V databáze aplikácie je uložených 34 vzorových znakov[5] v podobe dátových súborov, kde je daný znak reprezentovaný postupnosťou bitov. Spolu sa porovná 5400 pixelov vzorového obrazca s cieľovým bit po bite a vyhodnotí sa percentuálna úspešnosť zhody. Na konci porovnávania dostávame výsledok v podobe identifikácie znaku, kde výpočet dosiahol maximálne percento zhody. Ak je toto číslo väčšie ako 70%, vyhodnotí sa proces klasifikácie znaku za úspešný a na obrazovke sa zobrazí rozpoznaný znak v ASCII podobe.

Celkovú úspešnosť rozpoznania je možné sledovať pre každý znak zvlášť, nechýba ani celková komplexná úpešnosť rozpoznania značky.

Aby sa zvýšila presnosť a kvalita rozpoznávania, je nutné striktne dodržať formát klasického slovenského evidenčného čísla vozidla v tvare takom, že písmená sa nachádzajú na pozíciách 1, 2, 6 a 7, čísla na 3. až 5. mieste. Nebolo tak potrebné riešiť zhodné vlasnosti znakov 0 a O, 5 a S a 2 a Z. Najväčsie problémy však stále vykazuje zámena písmen D a O.

[5]Počet znakov, ktoré sa môžu vyskytovať na slovenských ŠPZ je 34, z toho 24 písmen (ABCDEFGHIJKLMNOPRSTUVZXY) a 10 číslic (0123456789).

Pseudokód 5.5: Rozpoznanie a klasifikácia jednotlivých znakov

```
classify and recognize character (7 times) :
    converting cut character to black and white;
    template matching :
        compate character with all patterns (A-Z or 0-9) {
            for (pixels in all rows) {
                for (pixels in all columns) {
                    matching percentage update
                }
            }
        }

    if (percentage >= 70%)
        return CHARACTER WAS SUCCESSFULY RECOGNIZED AND CLASSIFIED;
    else
        return CHARACTER WASN'T RECOGNIZED;
```

Kapitola 6

Testovanie aplikácie

Každú vyvíjanú aplikáciu je dobré nielen na konci, ale aj počas vývoja podrobiť určitým testovacím scenárom (Príloha A). Cieľom je nájdenie a odhalenie čo najväčšieho počtu chýb a slabých miest programu a následne ich opraviť. Testujeme dve veci - to, čo fungovať má a to, čo naopak nemá. Je dôležité, aby na obraze, kde sa žiadna ŠPZ nenachádza, nebola ani detekovaná. Testovanie spočíva v definovaní vstupu testovacieho scenára, správania programu a očakávaných výstupných hodnôt, ktoré sú následne porovnávané s vopred pripravenými predpokladmi. Každý test je zameraný na istú špecifickú udalosť, ktorá môže nastať pri práci s aplikáciou. Preto je dobré pokryť každú extrémnu, ale i všetky bežné situácie, ktoré môžu nastať. Tieto testy sa delia na dve väčšie skupiny, podľa toho akou technikou boli vytvorené. Oba druhy testov sa v praxi väčšinou kombinujú (tzv. *Greybox*), my si ale popíšeme každý samostatne [16].

6.1 Whitebox štrukturálne testovanie

Testujeme predovšetkým vnútornú logiku aplikácie. Cieľom je zaistenie, že všetky časti programu zdrojového kódu fungujú bezchybne čo sa týka riadiacej logiky. V praxi je úplný priechod všetkými cestami prakticky nedosiahnuteľný. Snahou je pokryť čo najväčšie množstvo hodnôt, ktoré môžu podmienky (`if`, `for`, `while`, ...) nadobúdať.

6.1.1 Asymptotická zložitosť algoritmov aplikácie

Najčastejšie uvádzaným hodnotiacim kritériom pre algoritmy je asymptotická časová zložitosť. Vyjadruje sa na základe porovnania algoritmu s nejakou funkciou pre N blížiace sa k nekonečnu. Toto porovnanie má niekoľko podôb. Najzákladnejšie a pre nás najpodstatnejšie sú O (Omikron) pre najhoršiu - hornú hranicu chovania a Θ (Theta) vyjadrujúca triedu chovania. Pre úplnosť spomenieme, že existuje aj zložitosť Ω (Omega) - najlepšia, dolná hranica chovania [9].

Lokalizácia značky v obraze (pseudokód 5.2) sa zaraďuje svojim algoritmom do triedy s kvadratickou časovou zložitosťou ($O(n^2)$) (rovnica 6.1). Obsahuje dvakrát za sebou dva vnorené `for` cykly.

$$2 \times O(n^2) = O(n^2) \tag{6.1}$$

Je na mieste poznamenať, že sa naozaj jedná o hornú hranicu chovania (najhorší prípad), ktorá nastane iba ak je značka umiestnená v pravom dolnom rohu alebo sa v obraze vôbec nenachádza. Pokiaľ je kdekoľvej inde, čas výpočtu sa skracuje, pretože algoritmus po detekcii príznaku výskytu značky ďalšiu nehľadá a úspešne skončí. Najlepší prípad a najkratší čas výpočtu lokalizácie značky nastáva ak sa nachádza vľavo hore.

Segmentácia znakov v značke (pseudokód 5.3) má zložitosť takisto kvadratickú. Môžeme ju vyjadriť nasledujúcim vzťahom (rovnica 6.2).

$$7 \times O(n^2) = O(n^2) \tag{6.2}$$

Záverečná fáza - rozpoznávanie a klasifikácia znakov (pseudokód 5.5) je časovo najpomalšia, asymptotickou zložitosťou sa nijak neodlišuje od predchádzajúcich dvoch častí aplikácie. Vystihuje ju vzťah (rovnica 6.3).

$$7 \times (O(n^2) + O(n^2)) = 7 \times O(n^2) = O(n^2) \tag{6.3}$$

Môžeme teda skonštatovať, že každá ucelená časť programu, ako aj celá aplikácia má **kvadratickú asymptotickú časovú zložitosť**.

6.2 Blackbox užívateľské testovanie

Tiež nazývané ako testovanie čiernej skrinky alebo funkčné testovanie. Pri tomto testovaní zisťujeme kedy sa program chová tak, ako je uvedené v špecifikácii a zadaní. Nezaujíma nás samotná implementácia programu.

6.2.1 Vzdialenosť

Vstupom pre aplikáciu je ideálne značka vyfotografovaná cca 30 - 40 cm od objektívu. V podstate je jedno z akej vzdialenosti je originálna fotografia nasnímaná, je potrebné ju ale následne pomocou jednoduchého editačného režimu, ktorý je súčasťou aplikácie a práce s fotoaparátom ako takej, dostatočne priblížiť, aby šírka značky takmer zodpovedala šírke snímanej oblasti, prípadne o niečo menšej. Algoritmus detekuje značku ak je minimálne 100 pixelov široká a 100 pixelov vysoká. Tento krok je dôležitý kvôli správnej detekcii značky.

Na druhej strane, pre správnu segmentáciu jednotlivých znakov, je potrebné značku presne napozíciovať dovnútra ohraničenej oblasti. Táto fáza je najcitlivejším miestom celého procesu rozpoznávania a aj preto je tu potrebných štatisticky najviac interaktívnych zásahov a korekcií zo strany užívateľa. V tomto prípade sa dostávame skoro stále k veľkosti značke, ktorá korešponduje so značkou zosnímanej zo vzdialenosti 30-40 cm.

6.2.2 Uhol

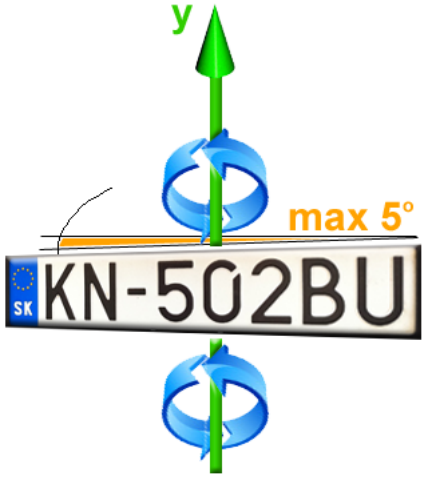
Podstatné v mojej aplikácii je, aby bola značka na fotografii vo vodorovnej polohe s maximálnym vychýlením 2° (obr. 6.1). V tomto prípade je možné prakticky každú značku úspešne lokalizovať a následne v nej klasifikovať znaky.

Obrázok 6.1: Maximálny uhol natočenia značky voči vodorovnej polohe

Podobne je to aj pri snímaní značky z iného uhla pootočenia značky okolo osi X (obr. 6.3) alebo osi Y (obr. 6.2). Tu je tolerancia o niečo miernejšia - 5° v oboch smeroch, aby bola značka úspešne lokalizovaná.

Keďže vo vyrezanej ŠPZ segmentované znaky prahujem pomocou dynamicky upravovaného prahu na základe priemeru hodnôt všetkých pixelov v značke, je možné úspešne rozpoznať aj pomerne špinavú značku a značku s tmavším podkladom ako je bežné.

Program bol otestovaný na sade 250 slovenských ŠPZ (200 s EÚ pruhom, 50 starých). Všetky ŠPZ boli nasnímané pomocou našej aplikácie. U 150 z nich bol dodržaný postup snímania pomocou pomocného *layout*u. Mali teda potrebné rozmery a prípustný uhol naklonenia. Sto ďalších bolo nasnímaných z rôznych uhlov a slúžili len ako testovacia vzorka pre stanovenie maximálnych hraníc uhlov natočenia a možnej rotácie.



Obrázok 6.2: Maximálny uhol natočenia značky okolo osi Y

Obrázok 6.3: Maximálny uhol natočenia značky okolo osi X

Po skončení testovania[1] som dospel k nasledovným záverom:

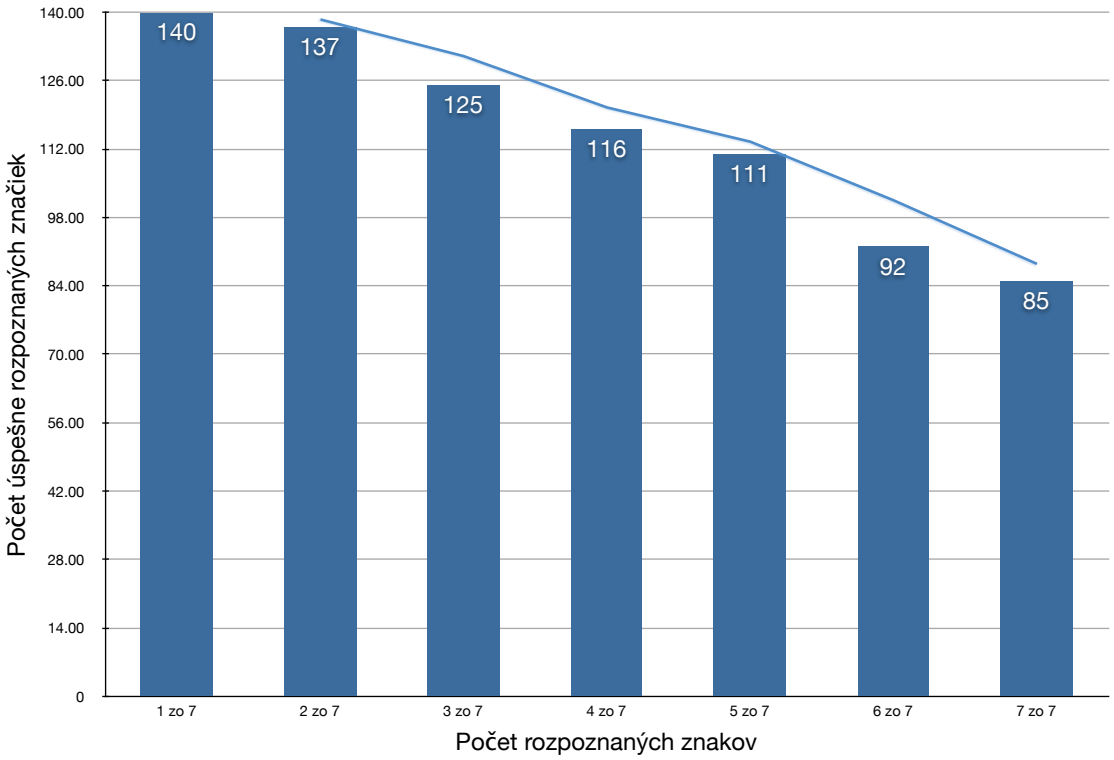
Celkový počet testovaných ŠPZ : **150**

[1] Zahrňuje iba majoritnú časť (150) ŠPZ určených na reálne testovanie

- 140 úspešne lokalizovaných značiek (značka na obraze bola v dovolenej horeuvedenej odchýlke, čo sa týka rotácie a natočenia okolo osí. Bola dostatočne veľkých rozmerov, minimálne 100 x 100 pixelov). Z nich po vyhodnotení testov vzišli nasledujúce výsledky:

  Ručné dolaďovanie značky povolené (obr. 6.4):

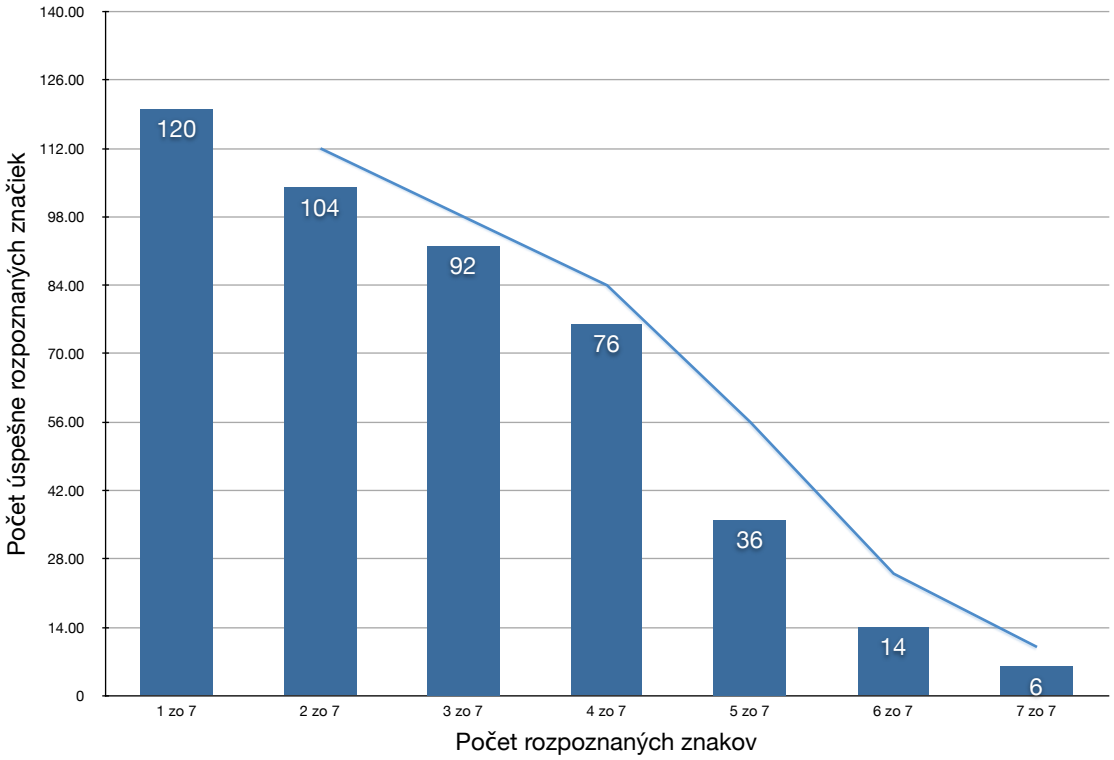
- 100% - úspešne segmentovaný a rozpoznaný aspoň 1 zo 7 znakov
- 98% - úspešne segmentované a rozpoznané aspoň 2 zo 7 znakov
- 89% - úspešne segmentované a rozpoznané aspoň 3 zo 7 znakov
- 83% - úspešne segmentované a rozpoznané aspoň 4 zo 7 znakov
- 79% - úspešne segmentovaných a rozpoznaných aspoň 5 zo 7 znakov
- 66% - úspešne segmentovaných a rozpoznaných aspoň 6 zo 7 znakov
- 61% - úspešne segmentované a rozpoznané všetky znaky v značke



Obrázok 6.4: Graf znázorňujúci počet úspešne rozpoznaných testovaných ŠPZ s povoleným ručným dolaďovaním

Značka bola rozpoznávaná bez akejkoľvek interakcie a manipulácie s pozíciou značky zo strany užívateľa (obr. 6.5):

- 86% - úspešne segmentovaný a rozpoznaný aspoň 1 zo 7 znakov
- 74% - úspešne segmentované a rozpoznané aspoň 2 zo 7 znakov
- 66% - úspešne segmentované a rozpoznané aspoň 3 zo 7 znakov
- 54% - úspešne segmentované a rozpoznané aspoň 4 zo 7 znakov
- 26% - úspešne segmentovaných a rozpoznaných aspoň 5 zo 7 znakov
- 10% - úspešne segmentovaných a rozpoznaných aspoň 6 zo 7 znakov
- 4% - úspešne segmentované a rozpoznané všetky znaky v značke



Obrázok 6.5: Graf znázorňujúci počet úspešne rozpoznaných testovaných ŠPZ bez dodatočného ručného dolaďovania

6.2.3 Nočné zábery ŠPZ

Blesk fotoaparátu veľmi výrazne uľahčuje rozpoznanie značky v úplnej tme a v svetelne nepriaznivých podmienkach. Keďže karoséria automobilu svetlo v tme neemituje, značky sú vybavené antireflexnou vrstvou (obr. 6.6) a zmesou polykarbonátu a polyesteru vyžarujúceho svetlo. Pri zapnutom osvetlení vozidla alebo nasmerovaní svetelného zdroja naň, prebieha lokalizácia presne a s takmer 100% spoľahlivosťou.

Obrázok 6.6: Príklad ŠPZ nasnímanej v tme za pomoci blesku

Kapitola 7

Pokračovanie v práci a možné rozšírenia

Ďalej je možné rozšíriť aplikáciu o rozpoznanie dvoch a viacerých ŠPZ na jednej snímke, čo má uplatnenie hlavne pri dvoj- a viacprúdových komunikáciách - diaľniciach. Implementovať by bolo možné aj rozšírené rozpoznávanie značiek z extrémnejších uhlov fotenia a stupňa natočenia fotografie.

Keďže existuje viac druhov a typov značiek, takisto by bolo možné telefónom ihneď s určitou pravdepodobnosťou identifikovať, či sa jedná o osobný automobil alebo motocykel.

Efektné by bolo takisto na základe rozpoznanej ŠPZ určiť štát a krajské mesto kde je vozidlo evidované, prípadne z akejsi internej databázy[1] priamo určiť typ aj identitu majiteľa vozidla. Jedným z nápadov, ako by program mohol ďalej naložiť s touto informáciou a pekne by tak simuloval prostredie, v akom pracuje napríklad polícia, by bola kontrola značky a jej porovnávanie s uloženým zoznamom "*hľadaných*" ŠPZ. Aplikácia by v prípade zhody oznámila užívateľovi výstražnú informáciu.

Nakoľko telefón disponuje pomerne citlivým GPS prijímačom, môže byť spolu s vyfotenou a rozpoznanou značkou zaznamenaný aj čas a aktuálna poloha zachyteného vozidla. Táto pozícia sa potom prehľadne zobrazí na mapovom podklade.

Neimplementovaný zostal aj môj nápad a zámer zaniesť do *layoutu* snímania fotoaparátom akcelerometrom podporovanú vodováhu, ktorá by užívateľovi priamo pri snímaní ukazovala polohu telefónu a dokázal by tak presnejšie posúdiť, či sníma značku v horizontálnej polohe bez rotácie.

Ďalším rozšírením by bola možnosť v aplikácii uchovávať a vracať sa k posledným rozpoznaným ŠPZ.

[1] Súčasťou každej iPhone aplikácie je zlinkovaná SQLite - odľahčená SQL knižnica pre uchovávanie užívateľských dát.

Záver

Cieľom tejto práce bolo zoznámenie sa s vývojom aplikácií pre mobilné zariadenie Apple iPhone. Preniknúť do tejto oblasti bolo miestami dosť náročné, nakoľko som bol, čo sa týka programovania pre operačný systém iOS v jazyku Objective-C, úplný začiatočník. Výhodou bolo, že moje skúsenosti z iných programovacích, či už objektovo orientovaných ale aj klasických funkcionálnych jazykov, boli na podstatne vyššej úrovni.

V druhom, najpodstatnejšom bode práce, som sa pustil do štúdia literatúry ohľadne spracovania obrazu a počítačového videnia. Musel som si osvojiť dátové štruktúry, ktoré sa pri práci s obrazom používajú a prispôsobiť rôzne metódy analýzy a vyhodnocovania obrazu k tomu, čo bolo pre moju aplikáciu najpotrebnejšie. Využil som najmä štandardné metódy, akými sú prahovanie obrazu statickým i dynamickým prahom, prevod farebného obrazu do obrazu so stupňami šedi, detekcia horizontálnych a vertikálnych hrán a konkatenácia[2] viacerých obrazov do jedného. Ďalšou fázou po lokalizácii štátnej poznávacej značky bola segmentácia znakov. Tam som využil fakt, že v rámci môjho zadania som sa mohol spoľahnúť na to, že sa bude jednať iba o slovenské značky a tie majú pevne stanovený počet aj pozície jednotlivých znakov. To mi výrazne uľahčilo implementáciu.

Na rad prišlo rozpoznávanie znakov a ich klasifikácia do triedy písmen alebo číslic, na základe najvyššieho vyhodnoteného *template matchingu*. Dôkladne som sa zameral najmä na testovanie aplikácie, a to z pohľadu programátorského - otestoval som rôzne chybové stavy a stavy, pri ktorých by aplikácia mohla *spadnúť* následkom zle uvoľňovanej pamäte, takisto z pohľadu užívateľského, kde sa jednalo o testovanie najbežnejších užívateľských vstupov a validity výstupov.

V závere som dal do povedomia niektoré z možných rozšírení, ktoré sa mi buď nepodarilo implementovať a boli nad rámec zadania alebo som nemal dostatočné prostriedky k ich realizácii.

[2] Konkatenácia je spájanie dvoch alebo viacerých objektov do jedného.

Literatúra

[1] Apple.com [online]. Apple Inc. [cit. 13.4.2011] Dostupné tiež z WWW: http://www.apple.com/iphone/specs.html.

[2] Ali, M.: *iPhone SDK programming : developing mobile applications for apple iPhone and iPod touch*. Chichester: Wiley, 2009, 382 s., ISBN 978-047-0742-822.

[3] iKnow Club ČVUT Praha: Seminář Programování pro iPhone [online]. Video dostupné tu: http://cvut.iknow.eu/iphone/.

[4] Dave, M.; LaMarche, J.: *Beginning iPhone development : exploring the iPhone SDK*. Berkely: Apress, 2009, 508 s., ISBN 978-143-0216-261.

[5] Dave, M.; LaMarche, J.: *iPhone SDK : průvodce vývojem aplikací pro iPhone a iPod touch*. Brno: Computer Press, 1. vydanie, 2010, 480 s., ISBN 978-802-5128-206.

[6] Eyedea Recognition s.r.o: Number Plate Reading Software for static cameras. Dostupné tiež z WWW: http://www.eyedea.cz/index.php?load=products/lpr.

[7] Goenka, K.: Image Processing iPhone App. University of Georgia: 16. 9. 2009, 16 s., CSCI 8810 Project Report.

[8] Gonzales, R. C.; Woods, R. E.: *Digital image processing*. Upper Saddle River: Prentice Hall, 2. vydanie, 2002, 793 s., ISBN 02-011-8075-8.

[9] Honzik M Jan, Prof. Ing., CSc.: *IAL - Algoritmy - Studijní opora*. Brno: FIT VUT Brno, 2009, 261 s.

[10] Ketelaars, N.: Final Project: Automated License Plate Recognition. AIMe Magazine. 2010, 1. číslo. Dostupné tiež z WWW: http://www.win.tue.nl/aime/Files/apr2002_license.pdf.

[11] Khalil, M. I.: Car Plate Recognition Using the Template Matching Method. International Journal of Computer Theory and Engineering. 5. 10. 2010, 2. ročník, 5. číslo, s. 683-687. Dostupné tiež z WWW: http://www.ijcte.org/papers/224-G290.pdf . ISSN 1793-8201.

[12] Kochan, S. G.: *Objective-C 2.0 : výukový kurz programování pro Mac OS X a iPhone*. Brno: Computer Press, 1. vydanie, 2010, 550 s., ISBN 978-802-5126-547.

[13] Parker, J. R.: *Algorithms for image processing and computer vision*. New York: Wiley Computer Publishing, 1997, 417 s., ISBN 04-711-4056-2.

[14] Petzold, C.: *Programming Windows Phone 7*. USA: Microsoft Press, 2010, 997 s., ISBN 978-0-7356-4335-2.

[15] Sadun, E.: *The iPhone developer's cookbook : building applications with the iPhone SDK*. Upper Saddle River: Addison-Wesley, 2009, 356 s., ISBN 978-032-1555-458.

[16] Sládeček Roman a kolektív: *Formální jazyky a překladače - Dokumentace k implementaci interpretu imperativního jazyka IFJ09*. Brno: FIT VUT Brno, 2009, 14 s.

[17] SR, N.: Zákon o cestnej premávke 8/2009 Z.z., Čl. 6: EVIDENCIA VOZIDIEL, EVIDOVANIE VOZIDIEL A EVIDENČNÉ ČÍSLA. Dostupné tiež z WWW: http://blog.sme.sk/blog/115/181318/2009-8-zakon-o-cestnej-premavke.html.

[18] Watt, A.; Policarpo, F.: *Computer image*. Massachusetts: Addison-Wesley, 1. vydanie, 1998, 749 s., ISBN 02-014-2298-0.

[19] Šonka, M.; Hlaváč, V.: *Počítačové vidění*. Praha: Grada, 1992, 272 s., ISBN 80-854-2467-3.

[20] Šonka, M.; Hlaváč, V.; Boyle, R.: *Image processing, analysis, and machine vision*. Toronto: Thomson, 3. vydanie, 2008, 829 s., ISBN 978-049-5082-521.

Zoznam obrázkov

Zoznam pseudokódov

Zoznam príloh

Príloha A: Testovanie aplikácie na konkrétnych značkách

- Správne odfotografovaná značka s dobrými svetelnými podmienkami
- Značka mierne natočená s horšími svetelnými podmienkami
- Tmavá a špinavá značka
- Nočná značka odfotografovaná pomocou blesku

Príloha B: Diagram tried aplikácie

Obsah CD

- Zdrojové kódy aplikácie: v adresári `/src/`
- Testovacie štátne poznávacie značky - 250ks: v adresári `/test_images/`
- Táto práca v LaTeXu: v adresári `/thesis/latex/`
- Táto práca vo formáte PDF: (cesta k súboru) `/thesis/pdf/bp.pdf`

Príloha A

Testovanie aplikácie na konkrétnych značkách

Správne odfotografovaná značka s dobrými svetelnými podmienkami

Bola odfotografovaná značka zo vzdialenosti 30 cm, bez rotácie okolo žiadnej z osí ani voči horizontálnej polohe.

V režime editácie a úprav fotografie sme nepreviedli žiadne zmeny a prijali fotografiu tak, ako bola nasnímaná.

Na (obr. A.1.1a) vidieť ako presne bola značka automaticky lokalizovaná. Ak sa pokúsime takto detekovanú značku rozpoznať, dostávame výsledok (obr. A.1.1b), ktorý pre nás nie je uspokojivý. Rozpoznané boli 4 znaky zo 7, z toho 1 zle. Ak však značku jemne doladíme pomocou gest alebo ovládacích prvkov (obr. A.1.1c) tak, aby každý znak bol presne umiestnený v pomocnom pruhu, dostávame sa pri klasifikácii znakov na 100% úspešnosť (obr. A.1.1d).

(a) Automatická lokalizácia

(b) Rozpoznanie znakov bez doladovania pozície značky

(c) Doladená lokalizácia značky užívateľom

(d) Rozpoznanie znakov s dodatočným doladením užívateľom

Obrázok A.1.1 - Test 1 - dobré svetelné podmienky

Značka mierne natočená s horšími svetelnými podmienkami

Značka bola nasnímaná zo vzdialenosti asi 1,5 m, z uhla pootočenia asi 6° okolo osi X aj Y.

V režime editácie a úprav fotografie sme ručne značku priblížili do ideálnej vzdialenosti (obr. A.2.1a) a vycentrovali na stred.

Ďalší obrázok (obr. A.2.1b) potvrdzuje, že značka nebola automaticky lokalizovaná. Museli sme ju ručne lokalizovať a jej pozíciu prispôsobiť pomocnému obdĺžniku (obr. A.2.1c).

Značku sme sa pokúsili rozpoznať a získali sme celkom slušný výsledok - 83% úspešnosť rozpoznaných znakov. Vidíme ale, že znak číslice 6 bol nesprávne klasifikovaný ako znak číslice 8, dokonca s percentuálnou úspešnosťou 80% (obr. A.2.1d). Je to dôsledok toho, že aplikácia obsahuje vzory znakov, ktoré sú vo východiskovej pozícii bez rotácie. Vplyvom toho sa niektoré pootočené znaky môžu javiť ako iné, nepootočené.

(a) Zvolená fotografia (b) Značka nebola automaticka rozpoznaná (c) Doladená lokalizácia značky užívateľom (d) Rozpoznanie znakov s dodatočným doladením užívateľom

Obrázok A.2.1 - Test 2 - horšie svetelné podmienky a pootočená značka

Tmavá a špinavá značka

Ďalším popisovaným scenárom je test rozpoznania značky poznačenej nečistotami a tmavšími miestami (obr. A.3.1a).

Značka nebola automaticky lokalizovaná (obr. A.3.1b) v tomto prípade preto, že bola príliš tmavá a detekcia hrán nedokázala detekovať jednoznačne hranice medzi značkou a okolím. Nastalo splynutie týchto vecí.

Ručne, pomocou ovládacích prvkov sme ju doladili (obr. A.3.1c). Nastala zaujímavá vec - dynamicky počítaný prah v rámci značky, pri segmentácii znakov, dokázal vynikajúco oddeliť znaky od pozadia aj pri takto tmavej a špinavej značke (obr. A.3.1d). Aj vďaka tomu je úspešnosť rozpoznania až 80% aj keď 2 zo 7 znakov bolo rozpoznaných zle.

(a) Zvolená fotografia (b) Značka nebola automaticka rozpoznaná (c) Doladená lokalizácia značky užívateľom (d) Rozpoznanie znakov s dodatočným doladením užívateľom

Obrázok A.3.1 - Test 3 - tmavá a zašpinená značka

Nočná značka odfotografovaná pomocou blesku

K poslednému testu bola vybraná značka vyfotená večer, za šera, v spolupráci s LED bleskom, ktorý má iPhone v sebe integrovaný. Značka nebola nijak extrémne naklonená voči osiam (obr. A.4.1a).

Lokalizovaná bola navlas presne (obr. A.4.1b), keďže detekcia hrán je v tomto prípade snáď najpresnejšia kvôli kontrastu pozadia fotografie a značky samotnej.

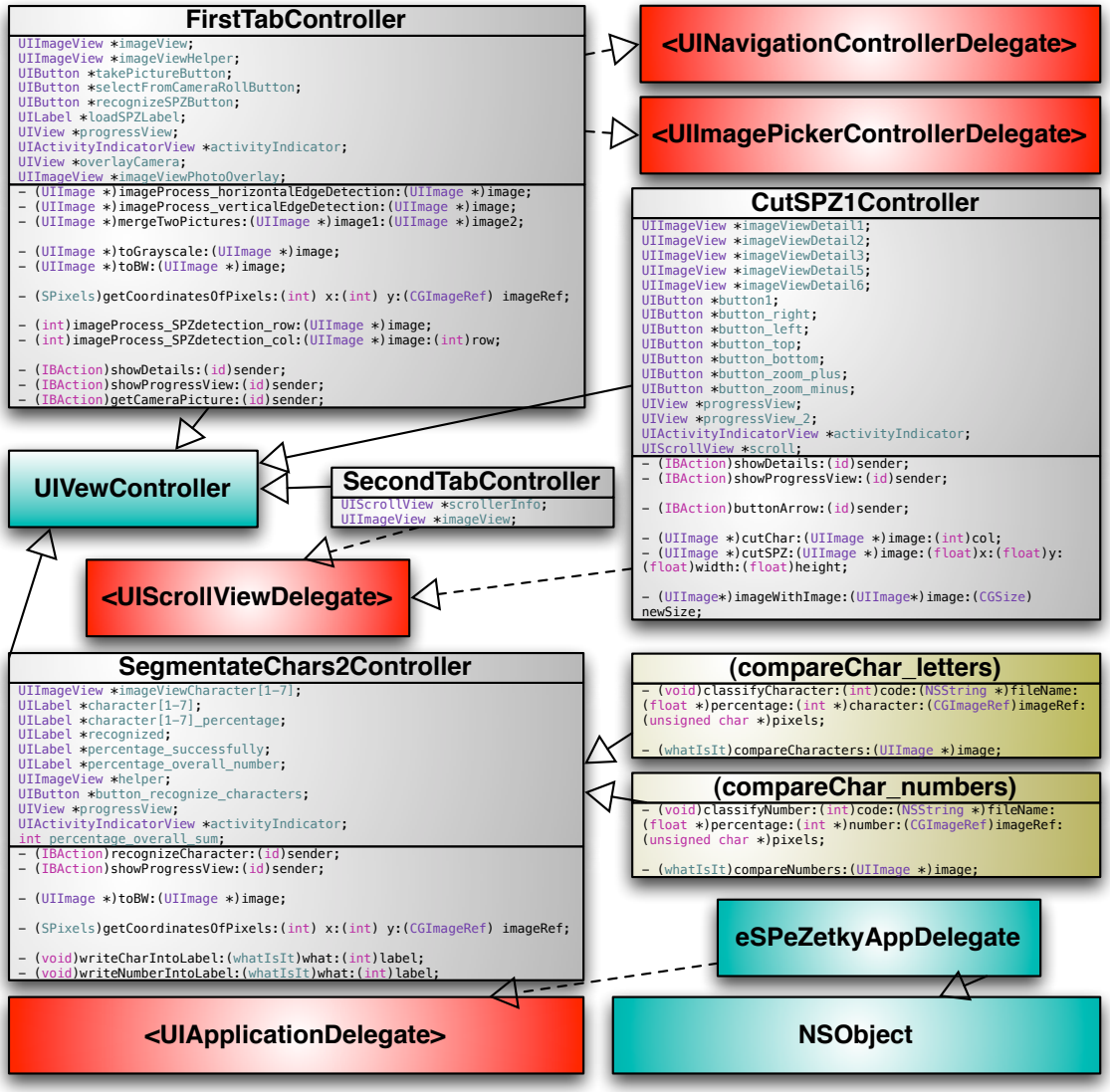
Klasifikácia prebehla bez problémov a bez chýb (obr. A.4.1c). Úspešnosť: 87%, všetkých 7 znakov klasifikovaných správne.

(a) Zvolená fotografia (b) Automaticky lokalizovaná značka (c) Rozpoznanie znakov

Obrázok A.4.1 - Test 4 - nočná značka odfotografovaná pomocou blesku

Príloha B

Diagram tried aplikácie



Diagriam tried - Class diagram