VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO ROZPOZNÁNÍ DOPRAVNÍCH ZNAČEK PRO WINDOWS PHONE 7

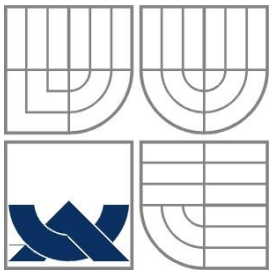
BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE MAREK DVOŘÁK

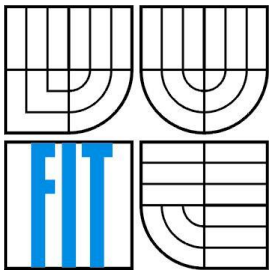
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO ROZPOZNÁNÍ DOPRAVNÍCH ZNAČEK PRO WINDOWS PHONE 7

WINDOWS PHONE 7 APPLICATION FOR TRAFFIC SIGN RECOGNITION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — MAREK DVOŘÁK

AUTHOR

VEDOUCÍ PRÁCE — Ing. BORIS PROCHÁZKA

SUPERVISOR

BRNO 2013

Abstrakt

Cílem této práce je vytvoření aplikace detekující a rozpoznávající dopravní značky pro mobilní telefony s operačním systémem Windows Phone 7. Teoretická část se zabývá vlastnostmi a historií systému Windows Phone 7, popisuje dopravní značení užívané v České republice a zkoumá metody detekce objektů v obraze. Praktická část práce obsahuje popis konkrétního řešení rozpoznávání značek ve fotografiích pořízených vestavěným fotoaparátem mobilního telefonu. Poslední část obsahuje testování funkčnosti aplikace v reálných podmínkách a zhodnocení výsledků těchto testů.

Abstract

Purpose of this thesis is to create an application that detects and recognizes traffic signs for mobile phones running Windows Phone 7 operating system. Theoretical part contains information about history and characteristics of Windows Phone 7 system, traffic signs in Czech Republic and various methods of image processing. Practical part of this thesis describes particular implementation of recognizing of traffic signs in photographs taken with mobile phone built-in camera. In last part this app is tested in real-life conditions and results of these tests are evaluated.

Klíčová slova

Windows Phone 7, mobilní telefon, dopravní značky, fotografie, detekce, segmentace, rozpoznávání, klasifikace

Keywords

Windows Phone 7, mobile phone, traffic signs, photography, detection, segmentation, recognition, classification

Citace

Dvořák Marek: Aplikace pro rozpoznání dopravních značek pro Windows Phone 7, bakalářská práce, Brno, FIT VUT v Brně, 2013

Aplikace pro rozpoznání dopravních značek pro Windows Phone 7

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.

……

Marek Dvořák

5. 5. 2013

Poděkování

Děkuji Ing. Borisi Procházkovi za rady a připomínky k této práci.

Obsah

Úvod 3

1 Windows Phone 7 5

1.1 Historie 5

1.2 Uživatelské prostředí 5

1.3 Doprovodné služby 6

1.4 Konkurenční operační systémy 7

1.4.1 iOS 7

1.4.2 Android 7

1.4.3 Symbian OS 7

1.5 Vývoj aplikací pro WP7 7

1.5.1 Hardwarové požadavky 7

1.5.2 Vývojové prostředí 8

1.5.3 Windows Phone Marketplace 10

2 Dopravní značení 12

2.1 Dělení dopravního značení 12

2.2 Rozměry a barevné řešení svislých dopravních značek 13

3 Segmentace obrazu 15

3.1 Metody založené na barevné informaci 15

3.1.1 Barevný model RGB 15

3.1.2 Barevný model HSV 16

3.1.3 Barevný model CieCam 17

3.1.4 Segmentace pomocí binární mapy 17

3.1.5 Segmentace pomocí histogramu 18

3.2 Metody založené na detekci hran 18

3.2.1 Cannyho hranový detektor 20

3.2.2 Detekce geometrických útvarů pomocí Houghovy transformace 21

3.2.3 Algoritmus RANSAC 22

4 Klasifikace dopravních značek 23
4.1 Porovnání se vzorem 23
4.2 AdaBoost 23
4.3 RBF neuronové sítě 25
4.4 FOSTS 26
5 Návrh aplikace 28
5.1 Výběr metod 28
5.2 Grafické rozhraní 30
5.3 Reprezentace datových struktur 31
6 Implementace 33
6.1 Pořízení a zpracování fotografie 33
6.2 Výběr kandidátních objektů 34
6.2.1 Detekce barevných objektů 34
6.2.2 Získání objektů z obrazu 35
6.3 Klasifikace objektů 36
6.3.1 Úprava objektů před porovnáním 36
6.3.2 Porovnání se vzorem 37
7 Testování aplikace 40
7.1 Testování výkonu 40
7.2 Testování přesnosti klasifikace 40
7.2.1 Maximální vzdálenost 41
7.2.2 Úhel záběru 41
7.2.3 Světelné podmínky 42
7.2.4 Výsledky testování 43
8 Pokračování v práci a možná rozšíření 44
Závěr 45
Použité zdroje 46

Úvod

V současné informační době se lidé snaží co nejvíce si zjednodušit život pomocí různých elektronických pomocníků. S tímto faktem úzce souvisí honba za automatizací téměř veškerých lidských činností. Manuální výroba je na ústupu, na infolinkách s lidmi komunikuje automat a také v oblasti automobilového průmyslu se výrobci snaží zákazníkům přinést co nejvíce pohodlí za pomoci různých pomocných systémů. Vedle zpříjemnění života pomáhají tyto systémy zejména zvýšit bezpečnost účastníků silničního provozu. Mezi tyto pomocníky můžeme zařadit i v současné době se prosazující systémy detekce dopravních značek. Vývojem aplikace s podobnou funkcionalitou se zabývá i tato práce. Konkrétně se jedná o mobilní aplikaci, která rozpoznává a klasifikuje dopravní značky na území České republiky. Jako cílová platforma byl zvolen systém Windows Phone 7 (WP7), obecné principy a postupy popsané v této práci jsou však využitelné i při vývoji pro libovolnou jinou mobilní platformu.

V první kapitole práce se věnuji především mobilnímu operačnímu systému Windows Phone 7, pro který je aplikace určena. Kapitola popisuje stručnou historii systému, jeho základní charakteristiky a také se věnuje nejvýznamnějším konkurentům. Dále obsahuje informace o charakteristikách vývoje pro tuto platformu. Čtenář je seznámen s vývojovým prostředím a podmínkami schvalování a zveřejňování aplikací.

Dopravními značkami, tedy předmětem detekce, se zabývá druhá kapitola. Kapitola dělí značky do kategorií dle použití a významu. Zabývá se i rozměry a barevnými schématy jednotlivých značek a uvádí také některá specifika pro jejich umístění.

Kapitola 3 seznamuje čtenáře s problematikou segmentace obrazu a její základní terminologií. Blíže popisuje několik metod, které je možné pro segmentaci využít. První variantou jsou metody z kategorie detekce hran, umožňující detekování zejména geometrických útvarů. Alternativou jsou metody založené na barevné informaci, které mohou využít některý z popsaných barevných modelů.

Čtvrtá kapitola obsahuje popis metod klasifikace dopravních značek. Zahrnuto je několik využívaných metod od prostého porovnání se vzorem, až po úzce specializované způsoby klasifikace, jako je metoda FOSTS.

V další kapitole se již soustředím na vlastní aplikaci. Provedu čtenáře výběrem vhodných metod a postupem návrhu grafického uživatelského rozhraní. Popsány jsou také některé speciální možnosti tvorby rozhraní v systému WP7. Na závěr kapitoly nastíním základní objektovou strukturu aplikace.

Kapitola 6 je jádrem celé práce, věnuje se vlastní implementaci rozpoznání dopravní značky ve fotografii. Popisuje celý postup od získání fotografie, přes její segmentaci, až k finální klasifikaci.

Předposlední kapitola předkládá čtenáři výsledky testování aplikace na reálných datech. Testovací sada zahrnuje testování úspěšnosti klasifikace a zjištění maximálních vzdáleností a úhlů, při kterých je ještě dosahováno kvalitních výsledků. Jako cílová hranice byla zvolena 80% úspěšnost rozpoznání značek. V poslední části navrhuji několik možných rozšíření a úprav aplikace.

Pro seznámení s platformou WP7 jsem využil především oficiálních zdrojů společnosti Microsoft [1, 2] a také knih věnujících se této problematice [3, 4]. Proniknout do oblasti zpracování obrazu mi pomohly publikace [5, 6, 7, 8]. Informace o dopravních značkách jsem čerpal z legislativy ČR [9].

1 Windows Phone 7

1.1 Historie

Windows Phone 7 (WP7) je mobilní operační systém vyvíjený společností Microsoft Corporation. Jeho přímým předchůdcem je operační systém Windows Mobile, který na trhu figuruje již od roku 2000 a jehož největším konkurentem byl Symbian OS společnosti Nokia. V roce 2007 však vstoupil na trh mobilních telefonů se svým prvním přístrojem Apple a svým zcela novým pojetím ovládání způsobil zemětřesení na trhu. Na tuto událost nejrychleji a nejúspěšněji zareagoval Google, který se svým systémem Android uvedeným v roce 2008 momentálně drží pozici nejoblíbenějšího mobilního OS na trhu.

V roce 2004 začal Microsoft pracovat na projektu Photon, který měl být nástupcem zastarávajícího Windows Mobile. Jeho vývoj byl ale velmi pomalý, proto byl celý projekt nakonec zrušen. Roku 2008 začaly práce na zcela novém mobilním operačním systému Windows Phone 7 Series. Slavnostní uvedení proběhlo v únoru 2010 v Barceloně již pod jménem Windows Phone 7 [10]. Tento systém můžeme prozatím nalézt hlavně v zařízeních HTC a Samsung, v únoru 2011 ale výkonný ředitel společnosti Microsoft Steve Ballmer oznámil spolupráci s finským výrobcem Nokia a WP7 se stal operačním systémem většiny smartphonů této společnosti [11]. Vlajkovou lodí přístrojů s operačním systémem Windows Phone je v současné době model Nokia Lumia 920.

V květnu roku 2011 byla oznámena a 27. září 2011 byla publikována ke stažení [12] první aktualizace systému označená jako Windows Phone 7.5 *„Mango“*. V říjnu 2012 Microsoft uvedl zcela novou generaci systému nazvanou Windows Phone 8. Tato je ovšem uvolněna pouze pro nejnovější zařízení. Některé nové funkce jsou implementovány v systému Windows Phone 7.8, který je poslední softwarovou aktualizací pro starší přístroje.

1.2 Uživatelské prostředí

Windows Phone 7 mimo jiné představil i nový styl uživatelského rozhraní nazvaný Metro (viz obrázek 1 [13]). Typograficky založené prostředí je velmi minimalistické a maximální důraz je kladen na co možná nejjednodušší a nejrychlejší obsluhu. Úvodní obrazovka systému je složena z tzv. „dlaždic“. Ty nefungují pouze jako odkazy ke spouštění aplikací, mohou také zobrazovat nejrůznější informace. Uživatel tak bez nutnosti spouštět vlastní aplikaci vidí např. počet nepřečtených zpráv či zmeškaných hovorů, aktuální teplotu ve zvolené lokaci či živou animaci složenou z fotografií svých kontaktů. Dlaždice si může na ploše seřadit podle svého přání a může si také vybrat z několika barevných schémat. Microsoft doporučuje především varianty s černým pozadím z důvodu úspory energie u OLED displejů, jimiž jsou WP7 telefony vybaveny. K dispozici jsou ovšem i světlé.

Podobné uživatelské rozhraní můžeme nalézt také u multimediálního přehrávače Microsoft Zune a na Start obrazovce systému Windows 8.

Unikátní vlastností systému je také spojení lokálního a vzdáleného obsahu pomocí přímé integrace se službami jako například Facebook nebo Twitter do tzv. „center“ (v originále Hubs). Díky tomu u jednotlivých kontaktů naleznete mimo osobní údaje, telefonní čísla či adresy také aktuální příspěvky a obrázky z uvedených sítí. Podobný systém funguje například při psaní zpráv, kdy v jedné konverzaci naleznete textové zprávy i zprávy

Obrázek 1: Domovská obrazovka WP7 a People Hub

z Facebooku, nebo při prohlížení obrázků a hudebního obsahu, kdy můžete přímo v telefonu nakupovat hudbu z Marketplace.

1.3 Doprovodné služby

Dnešní chytré telefony by se již neobešly bez dalších služeb, mezi které patří také obchod s aplikacemi pro danou platformu. Pro Windows Phone 7 je touto službou Windows Phone Marketplace. Marketplace byl spuštěn současně s uvedením WP7 na trh a v době představení aktualizace „Mango“ byla spuštěna také webová verze. V dnešní době již obsahuje přes 150000 aplikací [14]. Pro nákup hudby je využíván Zune Marketplace, ten ovšem prozatím není v České republice dostupný.

1.4 Konkurenční operační systémy

1.4.1 iOS

Mobilní operační systém pro zařízení společnosti Apple je průkopníkem na poli dotykového ovládání mobilních zařízení. Představil zcela nový způsob ovládání mobilních telefonů a odstartoval tak velkou vlnu zájmu o dotykové telefony, především díky jednoduchosti a efektivnosti tohoto přístupu. Byl uveden jako iPhone OS s nástupem iPhonu na trh v roce 2007. Aktuálně existuje ve verzi 6 a z iPhone se rozšířil také do dalších přístrojů se značkou Apple, a to iPod touch, iPad a Apple TV. Instalace na zařízení třetích stran je zakázána, proto i přes svou popularitu nedosahuje většinových podílů na trhu. Majitelé iOS přístrojů mohou využít nabídky Apple App Store a také stahovat multimediální obsah z iTunes Store, který v říjnu 2011 dorazil také do České republiky. Aplikace pro iOS jsou psány především v Objective-C, vývojové prostředí je z důvodu uzavřené politiky Apple dostupné pouze pro Mac OS. Samotný vývoj je tak podmíněn vlastnictvím počítače Mac.

1.4.2 Android

Největším hráčem na trhu mobilních operačních systémů je Android společnosti Google. Tento open-source operační systém založený na jádře Linux v současné době obsazuje více než polovinu trhu s mobilními zařízeními. Google tento systém nabízí jak ve vlastních telefonech, tak pro nasazení do přístrojů jiných výrobců. Také uživatelé Androidu mohou využívat online obchod s aplikacemi nazvaný Google Play. Ten v nabízeném množství již překonal App Store společnosti Apple, když v lednu 2013 dosáhl mety 800000 aplikací, zatímco Apple nabízí 775000 [15]. V porovnání s obchody ostatních společností také nabízí Android znatelně více aplikací zdarma. Jádro systému je napsané v jazyce C, aplikace se vyvíjí v jazyce Java. Aktuální verze systému s pořadovým číslem 4.2 se jmenuje Jelly Bean.

1.4.3 Symbian OS

Historie operačního systému Symbian sahá až do roku 2001, kdy byla na trh uvedena Nokia 9210. Od té doby se tento systém objevoval převážně v telefonech finského výrobce. Nokia ovšem oznámila přechod na Windows Phone 7 a dá se tak očekávat postupný ústup Symbianu do ústraní. Vývojovým prostředím pro Symbian je Qt a jazykem je C++, i když je možné vyvíjet také například v Javě nebo Pythonu.

1.5 Vývoj aplikací pro WP7

1.5.1 Hardwarové požadavky

Microsoft usnadnil vývojářům práci stanovením minimálních požadavků na hardware telefonů, které musí splňovat každý telefon využívající platformy WP7 [3]:

- Displej ve formátu WVGA (rozlišení 800x480 pixelů)
- Kapacitní dotykový displej s podporou detekce 4 různých doteků
- hardwarová tlačítka Start, Zpět a Hledat
- Fotoaparát s bleskem
- A-GPS[1] senzor, akcelerometr, kompas, světelné čidlo a detekci přiblížení objektů (například tváře při telefonování)
- Podpora datových přenosů
- minimálně 256 MB RAM a 8 GB paměti
- DirectX 9 hardwarová akcelerace

Jasné požadavky na HW umožňují vývojářům ušetřit si starosti s návrhy rozhraní pro různá rozlišení nebo ovládací prvky, což je nepopiratelnou výhodou oproti programování například pro systém Android, který může běžet na množství nesourodých zařízení. Uživatel díky tomu získá jistotu, že jakákoliv schválená aplikace poběží v pořádku a plynule.

Na plnění výkonových požadavků klade Microsoft velký důraz. Po nepříjemných uživatelských zkušenostech se systémem Windows Mobile, který byl podle spousty uživatelů nepřehledný a pomalý, je rychlost jedním z předních aspektů WP7 a cílem Microsoftu je zachovat tento trend i v externích aplikacích.

1.5.2 Vývojové prostředí

Aplikační platforma Windows Phone nabízí vývojářům dva různé frameworky, ve kterých je možné vyvíjet aplikace. Pro standardní aplikace je k dispozici Silverlight framework. Samotné zdrojové kódy jsou rozděleny do dvou částí. Pro definici layoutu prvků aplikace je využit značkovací jazyk označovaný jako XAML (eXtensible Application Markup Language). Vlastní programová část je pak psána v jazyce C# nebo Visual Basic .NET. Druhou variantou je XNA framework určený pro tvorbu náročných 2D a 3D her. Od Windows Phone 7.5 navíc můžeme při tvorbě jediné aplikace oba způsoby zkombinovat.

Kompletní vývojové prostředí je možné zdarma stáhnout v několika jazykových mutacích (čeština podporována není) z webových stránek Microsoftu. Podporované systémy jsou Windows Vista, 7 a 8 ve všech verzích kromě netbookové edice Starter. Emulátor navíc vyžaduje pro svůj běh DirectX 10. Balíček obsahuje následující komponenty [3]:

[1]Assisted GPS – využívá kromě GPS satelitů také informace z mobilní sítě.

- Microsoft Visual Studio 2012 Express for Windows Phone
- Windows Phone Emulator
- XNA Game Studio
- Expression Blend for Windows Phone
- příklady a dokumentace

Microsoft Visual Studio 2012 Express for Windows Phone

Tento program je rozšířením standardního Microsoft Visual Studia 2012. Pokud už je Visual Studio na počítači nainstalováno, dojde pouze k instalaci rozšíření pro Windows Phone.

Oproti klasickému rozhraní Visual Studia nabízí navíc některé prvky, které usnadňují vývoj mobilní aplikace. Design View je panel, který zobrazuje aktuální živý náhled uživatelského rozhraní aplikace, funguje také jako WYSIWYG[2] editor. Toolbox obsahuje seznam ovládacích prvků Windows Phone, které mohou být přetaženy do panelu Design View a umožňuje tak bleskovou tvorbu uživatelského rozhraní. Přímo pro tvorbu GUI je k dispozici Expression Blend 4, do kterého je také možné importovat soubory .PSD[3] a .AI[4].

Windows Phone Emulator

WPE je desktopová aplikace simulující Windows Phone zařízení a eliminující tak potřebu fyzicky jej vlastnit. Aplikace je navržena tak, aby simulovala i omezený výkon skutečných přístrojů, který je oproti desktopovým systémům pochopitelně nižší. V emulátoru nelze využívat fotoaparát přístroje, kompas a gyroskop. Naopak obsažen je virtuální akcelerometr a služby GPS včetně možnosti pořizovat a ukládat snímky z obrazovky.

Windows Phone Performance Analysis

Performance Analysis je nástroj, umožňující, jak ostatně napovídá název, měření a analýzu výkonu a požadavků aplikace pro její ladění. Při každém spuštění generuje soubor, obsahující tabulky a grafy znázorňující výkon a požadavky aplikace při daném spuštění. Množina údajů zobrazovaných v grafu obsahuje:

---

[2]What You See Is What You Get – vývojář přímo vytváří přesnou konečnou podobu GUI.

[3]Photoshop Document – nativní formát bitmapového editoru Adobe Photoshop.

[4]Adobe Illustrator Artwork - nativní formát vektorového editoru Adobe Illustrator.

- počet snímků/s
- využití CPU v %
- využití paměti v MB
- využití Garbage collectoru
- nahrávání obrázků
- starty animací

Detailnější statistiky je možné zobrazit při výběru časového úseku běhu aplikace. Vývojář je také automaticky upozorněn při překročení limitů využití CPU, paměti a podobně.

1.5.3 Windows Phone Marketplace

Aktuálně vládne trend umisťování aplikací pro mobilní telefony do internetových obchodů spravovaných výrobcem operačního systému. Nejinak je tomu v případě Microsoftu a Windows Phone. Toto schéma je výhodné pro všechny strany. Uživatel nemusí nikde složitě hledat, protože všechny dostupné aplikace nalezne na jednom místě, nejčastěji jde o webovou stránku. K obsahu může také přistupovat přes specializovanou aplikaci, a to jak z počítače, se kterým je telefon synchronizován, tak také přímo z mobilního zařízení. Rovněž má jistotu, že aplikace byla prověřena a neposkytne například jeho citlivá data vývojáři. Vývojář zase nemusí lákat zákazníky na svou osobní stránku, stačí jen svou aplikaci poslat na schválení. Odpadá tak i starost se servery, kde bude aplikace umístěna a podobně. Vlastník obchodu, zde tedy Microsoft, pak za svou službu dostává zaplaceno prostřednictvím podílu z výše tržeb za aplikace. Nás ovšem zajímá především role vývojáře.

Podmínkou pro zveřejnění vytvořené aplikace v Marketplace je registrace do služby App Hub. Tato registrace je placená, roční poplatek činí 99 amerických dolarů. Po zaplacení může vývojář zveřejnit libovolné množství placených aplikací a až 100 neplacených (případné další jsou zpoplatněny částkou 19,99 dolaru za aplikaci). Novým vývojářům z řad studentů otvírá Microsoft cestu prostřednictvím nulového registračního poplatku. Publikování aplikací je podmíněno pouze vlastnictvím platného DreamSpark účtu[5].

Samotný proces zveřejňování a ověřování je jednoduchý, stačí pouze odeslat vytvořený XAP soubor a vyplnit metadata (název, popis, kategorie, ikona reprezentující aplikaci v Marketplace apod.). Pokud aplikace vyhovuje všem požadavkům, je zveřejněna,

[5] Program Microsoftu poskytující studentům zdarma software pro návrh a vývoj.

v opačném případě vývojář dostane zprávu obsahující podrobné informace vysvětlující, proč nebylo možné aplikaci schválit.

2 Dopravní značení

2.1 Dělení dopravního značení

Účelem dopravního značení je regulovat a řídit spolu s dalšími prostředky silniční provoz na pozemních komunikacích. Značení můžeme rozdělit do následujících kategorií [9]:

- Dopravní značky
    - svislé
    - vodorovné
- Světelné, doprovodné akustické signály a výstražná světla
- Dopravní zařízení
- Zařízení pro provozní informace
- Speciální označení vozidel

V této práci se budeme zabývat pouze detekcí svislých dopravních značek. Jedná se o nejběžnější formu řízení silničního provozu, kdy grafická podoba značek je navíc napříč Evropou značně unifikovaná. Aplikace tedy bude využitelná i mimo území České republiky.

Podle zákona můžeme dělit svislé dopravní značky dle několika kritérií. Podle stálosti se může jednat o značky stálé, které jsou umístěny na konstrukcích pevně zabudovaných do terénu, značky proměnné, které mohou měnit svou podobu podle aktuální dopravní situace, nejčastěji zobrazované na speciálních panelech, a značky přenosné, umístěné na přemístitelných červenobíle pruhovaných stojanech [9].

Za normálních podmínek jsou značky umístěny při pravém okraji vozovky nebo nad ní. Pokud je nutné zdůraznit význam nebo zlepšit viditelnost, je možné značky umístit i k levému okraji vozovky, případně znásobit jejich množství. Přenosné dopravní značky pak mohou být umístěny i na vozovce, pokud to situace vyžaduje.

Podle významu pak můžeme značky dělit na [9]:

- Výstražné značky
- Značky upravující přednost
- Zákazové značky
- Příkazové značky

- Informativní značky
- Provozní značky
- Směrové značky
- jiné
- dodatkové tabulky

V této práci se budeme zabývat pouze značkami upravujícími přednost, výstražnými a zákazovými. Tyto značky mají největší dopad na bezpečnost silničního provozu a jejich přehlédnutí mívá často závažné důsledky, ať už v podobě majetkových škod, či újem na zdraví účastníků nehod.

2.2 Rozměry a barevné řešení svislých dopravních značek

Jednotlivé skupiny dopravních značek mají sjednocené barevné a rozměrové řešení. Pro účastníky silničního provozu je tedy snadné a intuitivní určit význam značky v co nejmenším čase, což je samozřejmě velmi důležité pro bezpečnost všech účastníků.

Výstražné dopravní značky upozorňující na různé objekty či nebezpečí mají až na výjimky (značky A31a-A32b označující železniční přejezd) tvar trojúhelníku položeného na hranu s bílým podkladem a červeným okrajem. Piktogramy umístěné v bílém poli jsou černé. Délka vodorovné hrany a výška značky činí 900 mm.

Obrázek 2: A1a - Zatáčka vpravo

Zákazové značky, jak napovídá název, ukládají omezení či zákazy. Jsou kulatého tvaru o průměru 700 mm, s bílým podkladem a červeným okrajem. Piktogramy jsou opět černé.

Obrázek 3: B1 - Zákaz vjezdu všech vozidel (v obou směrech)

Příkazové značky jsou také kulatého tvaru o průměru 700 mm, podklad je modrý a piktogramy jsou bílé.

Obrázek 4: C1 - Kruhový objezd

Informativní značky mají nejčastěji obdélníkový tvar, barevná řešení se různí. Generalizovanou podobu nemají ani značky upravující přednost. Veškeré čtvercové značky mají zadaný rozměr 500 mm, osmiúhelník používaný pro značku „P 6 – Stůj, dej přednost v jízdě“ má rozměr 700 mm. Rozměry ostatních tvarů odpovídají údajům uvedeným výše u ostatních typů značek.

Zákon také povoluje umístění dopravní značky na retroreflexní žlutozelený fluorescenční podklad pro zlepšení viditelnosti. Proměnné dopravní značky se mohou od stálých lišit v barevném provedení, kdy podklad je tmavý a piktogramy světlé. Červená barva ovšem musí být vždy zachována, stejně tak podoba příkazových značek.

3 Segmentace obrazu

K tomu, abychom vůbec mohli rozpoznat, o jakou značku se jedná, musíme ji nejprve v obraze lokalizovat. Pro člověka jde o samozřejmou a přirozenou akci, pro přístroj je naopak rozdělení na objekty jeden z nejnáročnějších úkolů, jaký mu v oblasti zpracování obrazu můžeme zadat. Na správné segmentaci závisí výsledek celé operace, a proto je důležité této části věnovat velkou pozornost.

Vlastní segmentace obrazu je proces, který rozdělí oblast R do n podoblastí R_1, R_2, …, R_n tak, že [16]:

- $\bigcup_{i=1}^{n} R_i = R$
- R_i je spojitá oblast pro všechna i
- $R_i \cap R_j = \emptyset$ pro každé i a j, $i \neq j$
- Všechny body podoblasti splňují jistou vlastnost Q
- Body dvou sousedních podoblastí se ve vlastnosti Q liší

Segmentační algoritmy se obecně zakládají buď na vzájemné odlišnosti, nebo naopak podobnosti jednotlivých bodů obrazu. Detekce dopravních značek má oproti detekci jiných objektům jisté výhody. Především jde o zákonem ustanovenou podobu, která zahrnuje tvar a barevné provedení. Obě tyto informace můžeme v detektorech úspěšně využít. Požadovaný tvar značky můžeme vyhledávat v hranové reprezentaci obrazu nebo můžeme v obraze vybrat oblasti splňující barevné předpoklady značky.

3.1 Metody založené na barevné informaci

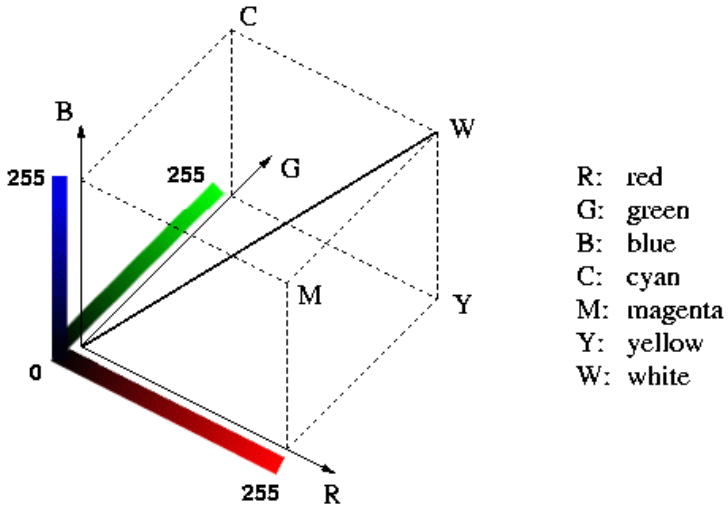
Pro popis barevné informace v obraze lze využít několika různých modelů. V této kapitole se zaměříme především na jejich výhody a nevýhody z pohledu detekce objektů. Informace v této kapitole byly čerpány zejména z prací [17] a [18].

3.1.1 Barevný model RGB

RGB je základním barevným modelem využívaným především v zobrazovacích zařízeních. Jedná se o aditivní model, kde je výsledná barva smíchána z červeného, zeleného a modrého světla (Red, Green, Blue – odtud název modelu). Hodnoty mohutnosti jednotlivých barevných složek můžeme vyjádřit buď procentuálně, nebo pomocí daného rozsahu hodnot (nejčastěji se používá 8-bitová barevná hloubka, tedy rozsah hodnot 0-255). Model můžeme zobrazit jako krychli umístěnou do počátku souřadného systému, kde

jednotlivé osy reprezentují barevné složky a barva je vyjádřena pomocí souřadnic v této krychli.

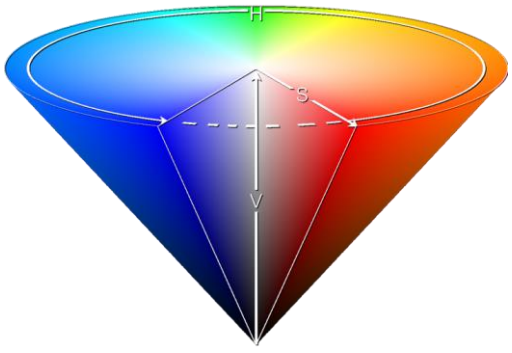
Výhodou tohoto modelu je jeho jednoduchost a skutečnost, že není třeba žádného převodu, práce s ním je tedy velmi rychlá. Nevýhodou z pohledu detekce oblastí je silná závislost na světelných podmínkách. Barevná reprezentace objektů se může na snímcích pořízených v různých podmínkách velmi lišit. Je to způsobeno především tím, že model přímo neukládá informaci o barevném odstínu a jeho jasu.



Obrázek 5: Barevný model RGB

3.1.2 Barevný model HSV

Model HSV (Hue, Saturation, Value) je velmi blízký lidskému vnímání barevné informace. Barvu popisuje taktéž pomocí tří složek. Hue vyjadřuje barevný tón, Saturation sytost barvu a Value její jas. Model můžeme znázornit pomocí kuželu. Díky oddělení informace o jasu je méně náchylný na změny světelných podmínek. Nevýhodou je nutnost převodu z původního RGB, což je poměrně časově náročná operace.



Obrázek 6: Barevný model HSV

Převod provedeme následovně [19]:

$$H = \begin{cases} \left(0 + \frac{G-B}{MAX-MIN}\right) \times 60 \; if \; R = MAX \\ \left(2 + \frac{B-R}{MAX-MIN}\right) \times 60 \; if \; G = MAX \\ \left(4 + \frac{R-G}{MAX-MIN}\right) \times 60 \; if \; B = MAX \end{cases} \quad (3.1)$$

$$S = \frac{MAX-MIN}{MAX} \quad (3.2)$$

$$V = MAX \quad (3.3)$$

3.1.3 Barevný model CieCam

Tento model byl vytvořen s cílem podat věrohodné informace o barvě za různých světelných podmínek. Výstupem je velké množství informací o barvě, model určuje světlost, odstín, sytost, barevnost, nasycení a jas barvy. Zároveň vyžaduje mnoho vstupních parametrů, přičemž jedním z nich je relativní tristimulus bílé barvy. Tento je definován pro různé světelné podmínky a umožňuje tak například různá nastavení aplikace podle počasí. Komplexní možnost nastavení a velké množství vstupních informací ale přináší vysokou výpočetní náročnost, a to především kvůli častému použití goniometrických funkcí a mocnin.

3.1.4 Segmentace pomocí binární mapy

Pro výběr oblastí s pravděpodobným výskytem dopravní značky se nejčastěji vytváří binární obraz pro příslušnou barvu. Toho dosáhneme nejsnadněji pomocí metody prahování. Základem této techniky je dělení obrazu podle hodnot intenzity jednotlivých bodů. Z histogramu obrazu získáme množinu 1 až n prahů T_1, T_2, …, T_n, podle kterých pak rozdělíme obraz f(x,y) do oblastí [20]:

$$g_i(x,y) = \begin{cases} 1 \; pokud \; T_{i-1} < f(x,y) < T_i \\ 0 \; jinak \end{cases} \quad (3.4)$$

Výsledkem je pak obraz, který obsahuje pouze dva typy bodů: body objektu (hodnota 1) a body pozadí (hodnota 0). Tento obraz pak vytváříme pro každou dvojici prahů. Jinou možností je přiřazení konkrétní hodnoty každé z oblastí v jednom výstupním obrazu.

V nejjednodušším případě hovoříme o tzv. globálním prahování, a to tehdy, pokud jsou prahy T_1, T_2, …, T_n aplikované na celý obraz. Pokud se hodnoty prahů v různých místech obrazu liší, jedná se o variabilní prahování.

Variabilní prahování přináší lepší výsledky u takových obrazů, kde je scéna například nerovnoměrně osvětlena a část objektu leží ve stínu. Zatímco globální prahování by nejspíše obraz rozdělilo na oblasti světla a stínu, variabilní prahování při vhodném algoritmu generování prahu přináší daleko přesnější výsledek. Variabilní prahování můžeme dále

rozdělit na lokální a adaptivní. U lokálního prahování závisí hodnoty prahů na vlastnostech (například průměrné intenzitě šedi) okolí daného bodu, při adaptivním prahování dělíme obraz na několik stejných částí a pro každou tuto část volíme hodnoty prahů. Tento postup je méně náročný než lokální prahování, mohou se ovšem vyskytnout problémy a nepřesnosti na hranicích oblastí.

Dalším pojmem spojeným s prahováním je tzv. poloprahování, jehož princip je prakticky shodný s obyčejným prahováním. Jediným rozdílným prvkem je absence přiřazování určité hodnoty bodům, které mají větší intenzitu než práh. Výsledkem je tedy obraz, který potlačuje body s nižší intenzitou. Části s vyšší intenzitou zůstávají zachovány.

Podle zvoleného barevného modelu pak pracujeme s různými prahy. Pro model RGB to budou hodnoty jednotlivých barevných složek, pro částečnou eliminaci vlivu světelných podmínek lze případně využít poměry mezi složkami. U modelu HSV se zaměříme na výseč požadovaného barevného odstínu doplněnou požadavkem na sytost a jas.

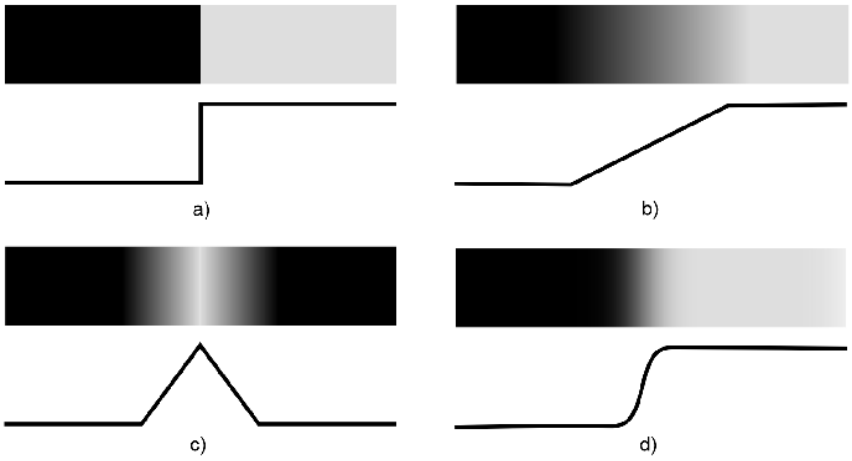
3.1.5 Segmentace pomocí histogramu

Kandidáty na dopravní značky je možné vyhledávat pomocí histogramu vertikální a horizontální projekce vyhledávané barvy. Barvy obrazu nejprve redukujeme například pomocí modelu HSV do několika základních barev. Poté vytvoříme histogram pro úrovně dané barvy v každém sloupci (vertikální projekce) nebo řádku (horizontální) obrazu. Z histogramu pak vybereme největší spojité oblasti s vysokou úrovní. Tvar těchto oblastí pak můžeme dále porovnávat a zjistit tak i tvar značky [18].

3.2 Metody založené na detekci hran

Rodina metod detekce hran využívá detekování oblastí s prudkou změnou intenzity bodů, tzv. hran. Ideální hranou je funkce typu step, případně line, kdy se intenzita obrazu mění nejvíce. Ve skutečnosti však na takové hrany narazíme jen zřídka, častěji dochází k postupnému přechodu intenzity, což reprezentují hrany typu ramp a roof (viz obr. 7 [16]).

K určení pozice hrany můžeme využít první či druhou derivaci. Zatímco u první derivace nám pozici hrany určuje vrchol funkce, u druhé derivace určíme polohu pomocí tzv. bodu „zero crossing“, což je průsečík mezi osou x a vrcholy druhé derivace (viz obr. 8 [16]).



Obrázek 7: Typy hran - a) step, b) ramp, c) roof, d) reálná hrana

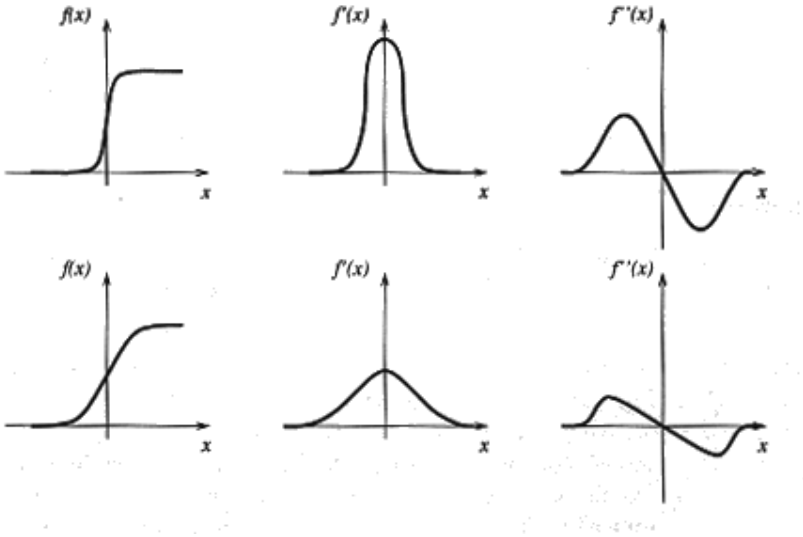
Při využití první derivace její hodnotu vypočítáme jako rozdíl okolních bodů, výsledný gradient pak pomocí následujícího vzorce [21]:

$$G(i,j) = \sqrt{\frac{\delta g^2}{\delta x} + \frac{\delta g^2}{\delta y}}, \tag{3.5}$$

kde parciální derivace vypočítáme jako

$$\frac{\partial g}{\partial x} = g(x+1) - g(x-1), \frac{\partial g}{\partial y} = g(y+1) - g(y-1) \tag{3.6}$$

Výsledný gradient pak porovnáváme s prahem. Na základě porovnání poté určíme, zda se opravdu jedná o hranu.



Obrázek 8: Průběh první a druhé derivace funkce

V praxi se k detekci používají hranové detektory využívající konvoluční filtrování obrazu. V konvolučních maskách je spojen výpočet gradientu pro detekci hrany i počáteční předrozmazání, které je u detekce hran nezbytné kvůli velké citlivosti těchto metod na šum

v obraze. Běžně se můžeme setkat s detektory využívajícími následující konvoluční masky [21]:

Prewittův operátor: $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$

Sobelův operátor: $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$

Detekce druhou derivací využívá nalezení průchodu funkce nulou, což je snazší než nalézat extrém funkce. Navíc odpadá fáze prahování tohoto extrému.

Jednou z možností je výpočet výsledku pomocí dvojího derivování, častější variantou je ovšem opět použití hranového detektoru, nejčastěji se využívají Laplaceovy operátory [21]:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

Laplaceovy operátory patří do kategorie označované jako Laplacian of a Gaussian (LoG). Tyto operátory jsou invariantní vůči rotaci, což odpovídá charakteristikám lidského vidění a také umožňuje shodné reakce na změnu intenzity nezávisle na směru. Další výhodou těchto operátorů je možnost volit velikost konvolučního jádra. S rostoucí velikostí jádra totiž klesá jejich citlivost na šum, ale na druhou stranu rychle roste časová náročnost výpočtu, takže je důležité zvolit velikost odpovídající našim požadavkům na detekci.

3.2.1 Cannyho hranový detektor

Mezi nejpoužívanější hranové detektory patří detektor navržený Johnem Cannym. Autor určil podmínky, které by měl dobrý detektor hran splňovat:

1. Detekce by měla být maximálně kvalitní. Množství nenalezených hran, případně falešně pozitivních nálezů by mělo být co nejnižší.
2. Detekce by měla být maximálně přesná. Nalezené hrana by měla ležet co možná nejblíže středu reálné hrany.
3. Každá hrana by měla být detekována pouze jednou.

Cannyho detektor se využívá především při detekci hran v šedotónových obrazech. Vyplývá to z omezení detektoru, který je schopen zpracovávat pouze obraz s jedním kanálem. Algoritmus můžeme popsat čtyřmi kroky:

1. Redukce šumu pomocí gaussovského filtru.
2. Nalezení velikosti a směru gradientu.
3. Zúžení hran pomocí non-maxima suppression. Ve směru gradientu jsou potlačeny hodnoty, které nejsou lokálním maximem.

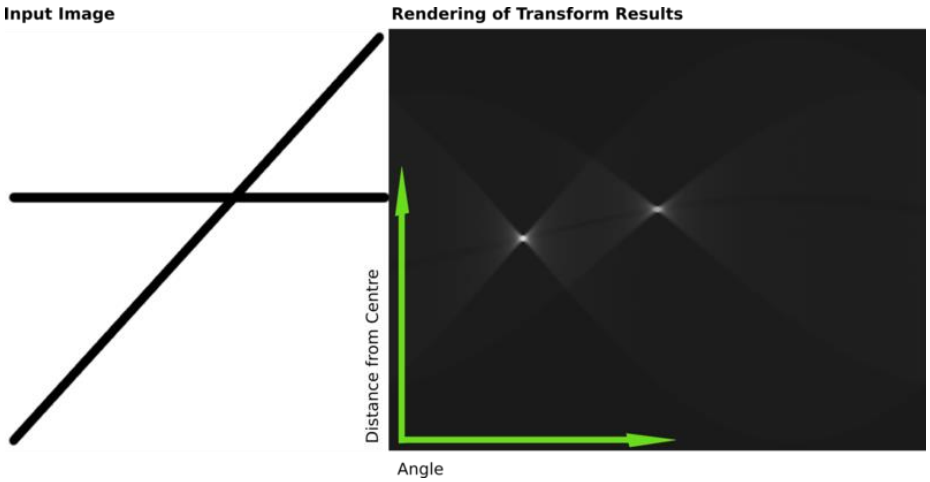
4. Prahování s hysterezí. Jedná se o prahování pomocí dvou prahů, které zvýhodňuje body ležící na hraně. Existují celkem dva prahy. Pokud je gradient v bodě vyšší než větší z prahů, je bod označen za součást hrany. Pokud je nižší než větší práh, ale zároveň vyšší než menší práh, rozhoduje se podle okolních bodů. Je-li alespoň jeden vedlejší bod součástí hrany, je označen i současný.

3.2.2 Detekce geometrických útvarů pomocí Houghovy transformace

Houghova transformace vychází z možnosti popsat geometrický útvar pomocí rovnice. Pro vyhledávání přímky tedy zjistíme z její rovnice (3.7), že potřebujeme nalézt parametry φ a r. Úhel φ je úhel, který svírá normála přímky procházející počátkem souřadného systému s osou x a parametr r udává vzdálenost přímky od počátku.

$$x\,cos\varphi + y\,sin\varphi = r \qquad (3.7)$$

Princip Houghovy transformace je prostý [22]. Každým bodem z prohledávané množiny (pro nás jde o množinu bodů hran) vedeme všechny možné přímky a jejich parametry zaznamenáváme do prostoru souřadnic φ a r. Pro každý bod nám tak v Houghově prostoru vznikne sinusoida. Pokud více bodů leží na jedné přímce, jejich sinusoidy se protnou v bodě označujícím parametry této přímky. Pro velké množství bodů tedy určíme parametry přímky jako maximální hodnotu v příslušném Houghově prostoru



Obrázek 9: Houghova transformace - vstupní obraz a výsledek

Pro vyhledávání kružnice postupujeme obdobně, hledáme ovšem tři parametry a, b, a c vycházející z rovnice kružnice (3.8), příslušný Houghův prostor tedy bude mít tři dimenze.

$$(x-a)^2 + (y-b)^2 = c^2 \qquad (3.8)$$

3.2.3 Algoritmus RANSAC

Algoritmus pracuje na principu ověřování hypotézy, že dva body leží na hledané přímce. Musíme určit tři parametry, které ovlivňují kvalitu výsledku. Je to počet náhodných pokusů vybrání bodů, maximální vzdálenost bodu od přímky, kdy ještě uznáváme, že bod podporuje hypotézu a minimální množství bodů, které musí hypotézu podpořit, abychom ji považovali za platnou.

Algoritmus náhodně vybere dva body a určí z nich rovnici přímky, na které tyto body leží. Pro všechny ostatní body určí jejich vzdálenost od této přímky, a pokud je menší, než zadaná hranice, označí bod jako podporující hypotézu. Pokud je hypotéza potvrzena dostatečným množstvím bodů, je označena jako správná a všechny body, které ji podpořily, jsou označeny jako body přímky [23].

Pro detekci kružnic algoritmus vybírá tři body a obdobně testuje hypotézu založenou na vzdálenosti bodů od kružnice určené těmito body.

4 Klasifikace dopravních značek

V této kapitole se seznámíme s několika základními metodami, které jsou využívány při rozpoznávání dopravních značek.

4.1 Porovnání se vzorem

Porovnání se vzorem nebo také *template matching* je nejjednodušší metodou na pochopení i implementaci. Postup je založen na konvoluci vstupních dat s množinou vzorů. Pracujeme se dvěma obrazovými komponenty. Prvním z nich je vzorový obraz, druhým pak zdrojový obraz, ve kterém budeme hledat výskyt vzorového obrazu. Postupně posouváme vzor po zdrojovém obrazu a zaznamenáváme hodnotu odchylky mezi oběma obrazy. Body s nejnižší hodnotou nám pak určují místa výskytu vzorového objektu.

Následující rovnice popisuje postup výpočtu odchylky vzorového obrazu T o rozměrech m,n od vstupního obrazu S v poloze x,y.

$$D(x,y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |S(x+i, y+j) - T(i,j)| \qquad (4.1)$$

Pro potřebu klasifikace dopravních značek je se předem nalezená kandidátní oblast upraví na velikost vzoru. Poté provádíme konvoluci s vzory odpovídajícími databázi značek a vybereme výsledek s nejmenší odchylkou. Náročnost metody je závislá na počtu vzorů, je tedy vhodné neporovnávat s kompletní databází, ale kandidátní objekt nejdříve zařadit do užší kategorie například podle tvaru nebo barvy značky. Jako nevýhodu můžeme také označit závislost výsledků porovnávání na natočení objektu na fotografii. Pro kvalitní výsledky je tedy nutné vstupní obraz transformovat do správné polohy tak, aby co nejvíce odpovídal vzorům. V porovnání metod provedeném autory článku [24] prostý *template matching* dosáhl úspěšnosti 63-86 %.

4.2 AdaBoost

Název algoritmu vznikl zkrácením slovního spojení „adaptive boosting“. Principem je vytvoření silného klasifikátoru pomocí kombinování množiny slabých. Pokud budeme uvažovat problém třídění do dvou tříd (např. je značka, není značka), pak můžeme algoritmus popsat rovnicí

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \qquad (4.2)$$

kde $h_t(x)$ je slabý klasifikátor, jehož výstupem je buď hodnota 1, nebo -1, a α_t je jeho váha.

Využít je možné jakékoliv klasifikátory, jedinou podmínkou je, aby nepracoval náhodně (s chybovostí $e_m = 1/2$). Takový klasifikátor, jak si ukážeme později, by dostal

přiřazenu nulovou váhu. V případě, že je chybovost $e_m > 1/2$, získá klasifikátor negativní váhu, řídíme se tedy opakem jeho rozhodnutí. Výsledek klasifikace, tedy zda se jedná či nejedná o náš požadovaný objekt, určuje znaménko výsledku rovnice 4.2.

$$H(x) = sign(f(x)) \quad (4.3)$$

Učení algoritmu, neboli tvorba požadovaného silného klasifikátoru, probíhá iterativně v M krocích [25]. Máme danou sadu trénovacích dat x_i s popisem y_i (uvažujeme problém dvou tříd – hodnoty +1, -1). Každému objektu přiřadíme váhu $w_i^{(1)} = 1$. V každé iteraci vypočítáváme součet vah objektů W a součet vah špatně klasifikovaných objektů W_e.

Pro $m = 1$ do M:

1. Ze sady dostupných klasifikátorů vybereme klasifikátor k_m tak, abychom minimalizovali hodnotu

$$W_e = \sum_{y_i \neq k_m(x_i)} w_i^{(m)} \quad (4.4)$$

2. Nastavíme váhu klasifikátoru na

$$\alpha_m = \frac{1}{2} \ln\left(\frac{1-e_m}{e_m}\right), \quad (4.5)$$

kde $e_m = W_e/W$

3. Upravíme váhy datových objektů pro další iteraci. Pokud je klasifikace $k_m(x_i)$ nesprávná, pak

$$w_i^{(m+1)} = w_i^{(m)} e^{\alpha_m} = w_i^{(m)} \sqrt{\frac{1-e_m}{e_m}} \quad (4.6)$$

v opačném případě

$$w_i^{(m+1)} = w_i^{(m)} e^{-\alpha_m} = w_i^{(m)} \sqrt{\frac{e_m}{1-e_m}} \quad (4.7)$$

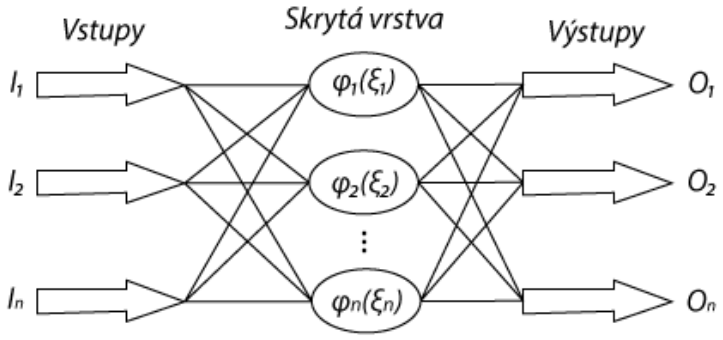
Výhodou algoritmu je jeho vysoká úspěšnost a odolnost proti přetrénování. Nevýhodou je náchylnost na šum v obraze a potřeba velkého množství trénovacích dat. Úspěšně byl využit při rozpoznávání lidských obličejů [26] a existují i modifikace pro dopravní značky. V práci [27] byl algoritmus natrénován na souboru 7000 pozitivních a 12000 negativních příkladů a byla dosažena úspěšnost 98%.

4.3 RBF neuronové sítě

Neuronová síť je systém prvků (neuronů), které jsou uspořádány tak, aby byl systém schopen zpracovat vstupní informaci požadovaným způsobem. Principem zpracování je získání výstupů pomocí aplikace transformační funkce na množinu vstupů. Správné reakce na vstupy se ale síť musí nejprve naučit. Do každého neuronu mohou vstupovat signály z libovolného množství dalších neuronů, přičemž tyto signály mohou mít nastaveny váhy. Pomocí těchto vah síť přizpůsobuje své reakce při učení. Při učení rozlišujeme dva přístupy:

- **S učitelem** – při tomto přístupu porovnává síť své výstupy s výstupy učitele. Poté upraví váhy jednotlivých signálů tak, aby co nejvíce snížila rozdíl mezi svým a požadovaným výstupem
- **Bez učitele** – síť nemá prostředky k určení správnosti výsledku. Vstupní data pak roztřiďuje podle určitých vlastností.

Při rozpoznávání dopravních značek se často využívají sítě typu RBF (Radial Basis Function). Tyto sítě se skládají ze tří vrstev – vstupní, skryté a výstupní. Vstupní a skrytá vrstva sítě jsou úplně propojeny, skrytá a výstupní vrstva nejčastěji také. Topologie sítě je zobrazena na obrázku.



Obrázek 10: Topologie RBF sítě

Základním prvkem RBF sítí je RBF neuron. Každý neuron obsahuje n vstupů ($x_1,\ldots,x_n$) a každý vstup x_i má přiřazenou váhu c_{ik}. Výstupní funkce neuronu je [28]:

$$y = \varphi(\xi), \tag{4.8}$$

kde ξ je potenciál neuronu

$$\xi = \frac{\|\vec{x}-\vec{c}\|}{b} \tag{4.9}$$

c a b označují střed, respektive šířku neuronu. Společně pak určují tzv. sféru vlivu neuronu. Pokud vstupní objekt leží v této sféře vlivu, považuje se za rozpoznaný neuronem, kterému sféra vlivu náleží. Aktivační funkcí neuronu φ je většinou Gaussova funkce

$$\varphi(z) = e^{-\xi^2} \tag{4.10}$$

Při učení RBF sítě se určuje několik parametrů:

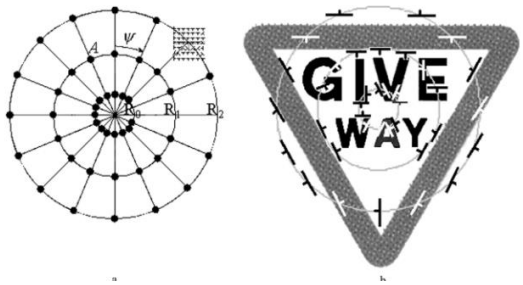
- Počet neuronů ve skryté vrstvě sítě
- Rozmístění neuronů v prostoru
- Určení šířky funkcí příslušnosti
- Výpočet vah výstupů neuronů do výstupní vrstvy

Úkolem prvních tří bodů je optimální pokrytí prostoru. Algoritmus použitý pro rozmístění neuronů závisí na rozložení vstupních dat. Používá se rovnoměrné rozdělení (vhodné pro data s rovnoměrným rozložením), náhodné rozdělení, nebo nejčastěji rozmístění pomocí algoritmu *k-means clustering*. Šířku (nebo také sféru vlivu) neuronu určujeme tak, abychom dosáhli co největšího pokrytí při současné minimalizaci překryvů jednotlivých sfér vlivu. Při špatném nastavení dochází ke zvýšení chybovosti sítě.

RBF neuronové sítě do svého srovnání zařadili autoři článku [24], kde uvádějí úspěšnost při klasifikaci přibližně 95 %. Záleží ovšem také na volbě vstupů. Autor práce [22] využil na vstupu přímo hodnoty pixelů objektu a úspěšnost u reálných snímků rapidně klesla oproti klasifikaci trénovacích vzorů.

4.4 FOSTS

Systém FOSTS (Foveal system for traffic signs) [29] se snaží obraz analyzovat podobným způsobem jako lidské oko. Zaměřuje se na důležité prvky v obraze a imituje také změnu vnímání ostrosti lidského oka. Vzory značek v databázi modelu popisují grafickou podobu značky pomocí množiny orientovaných segmentů, které odpovídají natočení hran ve vzorovém obrazu. Celkem je takto uloženo 49 hodnot v tzv. fixačních bodech. Tyto body leží na průsečících tří soustředných kružnic a šestnácti paprsků, úhel mezi jednotlivými paprsky je tedy 22,5 °, poloměry kružnic jsou 3, 9 a 15 (viz Obrázek 11 [29]).



Obrázek 11: a) FOSTS senzor s fixačními body, b) Detekované hrany ve fixačních bodech

Pro maximální úspěšnost rozpoznání je velmi důležité umístit senzor přesně do středu značky. Tento střed je určen jako geometrický střed barevného ohraničení značky, zároveň je odstraněno pozadí a je možno kandidátní objekt normalizovat. Ve vstupním obrazu je pak určena orientace hran α a jejich hustota ρ ve fixačních bodech senzoru A_i [17]:

$$\rho(A_i) = max\ \rho_\varphi(A_i) \quad (4.11)$$

$$\alpha(A_i) = \varphi\ když\ \rho_\varphi(A_i) = \rho(A_i) \quad (4.12)$$

$$\rho_\varphi(A_i) = \rho_\varphi(x_i, y_i) = \frac{1}{S(x_i,y_i)} \sum_{m,n} Sg_\varphi(Or(m + x_i, n + y_i)) \quad (4.13)$$

$$Sg_p(x) = \begin{cases} 1\ když\ x = p \\ 0\ jinak \end{cases} \quad (4.14)$$

kde $Or(x,y)$ je orientace detekované hrany v okolí se souřadnicemi (x,y), $S(x_i, y_i)$ je plocha oblasti zájmu a odpovídá 49 fixačním bodům.

Ze zjištěných hodnost je vytvořen vektor, který je porovnán pomocí rovnice 4.15 [17] s databází vektorů odpovídajících různým dopravním značkám. Z výsledků pak vybereme vektor s největší mírou podobnosti.

$$K^b = \sum_{i=0}^{48} \left[Sg\left({Or_i}^b - {Or_i}^{rw} \right) \cdot \left(1 - \left| {\rho_i}^b - {\rho_i}^{rw} \right| \right) \right] \quad (4.15)$$

$$Sg(x) = \begin{cases} 1\ když\ x = 0 \\ 0\ jinak \end{cases} \quad (4.16)$$

Or je orientace dominantní hrany, ρ její hustota, index b odpovídá vzoru značky, rw pak kandidátní oblasti.

Autoři práce [29] uvádějí úspěšnost algoritmu 95 %, úspěšně rozpoznali 93 z 98 dopravních značek. Výsledky za různých povětrnostních podmínek (uvažováno slunečno a zataženo) byly velmi podobné. Další výhodou metody je její rychlost.

5 Návrh aplikace

Po ujasnění si požadavků na aplikaci pokračuje proces vývoje etapou návrhu. V této fázi se snažíme najít nejlepší způsob, jakým dosáhnout požadované funkčnosti s ohledem na ostatní požadavky jako je výpočetní náročnost či vzhled výsledného programu. Ideálně vytvoříme více návrhů, ze kterých na základě důležitosti požadavků vybereme ten nejvíce vyhovující. U naší aplikace je nejdůležitějším faktorem ovlivňujícím úspěšnost a efektivnost detekce metoda, kterou budeme značky v obraze vyhledávat a posléze rozpoznávat.

5.1 Výběr metod

Projekt zaměřený na detekci a klasifikaci dopravního značení není v rámci závěrečných prací na VUT v Brně ničím novým. Tímto tématem už se zabývalo několik studentů přede mnou, ať už v rámci samostatné práce [22, 30, 31, 18, 32], či jako součást většího projektu, jakým může být například SMART CAR [17]. Nabízí se tedy možnost využít poznatky autorů těchto prací.

Nároky na aplikace, případně jejich rozsah, se pro jednotlivé práce liší. Některé projekty zahrnují pouze detekci a nezabývají se přesnou klasifikací, jiné naopak zvládají i klasifikaci v reálném čase. Společným prvkem je kromě tématu pouze skutečnost, že se jedná o aplikace určené pro desktopové systémy. Autoři tak mohli využít různých knihoven pro zpracování obrazu. Ve většině zmíněných projektů je využito knihovny OpenCV, kterou je možno využít jak pro akademické, tak pro komerční účely. Autoři tak měli k dispozici velké množství optimalizovaných funkcí počítačového vidění, jako jsou například Cannyho hranový detektor, Houghova transformace, nalezení kontur, funkce pro práci s histogramem a podobně.

V našem případě se ovšem jedná o mobilní aplikaci. Jednak tedy musíme počítat s omezenou výpočetní silou zařízení, jednak se musíme obejít bez pomoci specializovaných knihoven. Řešení se tedy nabízí dvě. Buď funkce, které budeme potřebovat, naprogramujeme sami, nebo zvolíme jiný postup. Vlastní implementace náročných funkcí je jistě možná, ale rozhodně nedosáhneme takové výkonnosti a optimalizace jako profesionální týmy vývojářů. O tomto faktu se můžeme přesvědčit například v práci [33], kde se autor pokusil o reimplementaci Houghovy transformace pro detekci čar. Funkční algoritmus sice implementoval, ve výsledku ale byl jeho program 4,2krát pomalejší a více než 2krát náročnější na paměť, než srovnatelný program využívající volně dostupné knihovny.

Pro segmentaci obrazu a nalezení kandidátních objektů je použita metoda binární mapy pro červenou barvu v HSV barevném prostoru. HSV model byl vybrán jako vhodný kompromis mezi kvalitou reprezentace barevné informace a náročností převodu z formátu

RGB, který je využíván v záznamových zařízeních (tedy i fotoaparátech mobilních telefonů). Více informací o vlastnostech jednotlivých modelů, které vedly k tomuto rozhodnutí je uvedeno v kapitole 3.1.

Jakou konkrétní značku objekt reprezentuje, je pak určeno pomocí prostého porovnání se vzorem, které bylo také využito například v pracích [31, 18, 32, 17]. Pro natrénování algoritmu AdaBoost bychom potřebovali velké množství trénovacích údajů, proto byla tato možnost zavržena. Neuronové sítě nebyly zvoleny z důvodu využití barevných hodnot jednotlivých bodů. Slabé výkony neuronových sítí v reálném provozu při použití jsou popsány v práci [22].

Porovnání se vzorem však bohužel není invariantní vůči rotaci. Pro otočení objektu do polohy vhodné pro porovnání bychom nejprve museli určit úhel, o který je natočen. U trojúhelníkových značek se nabízí nalezení hran značky, ze kterých by šlo tento úhel snadno vyčíst. Jak ale bylo uvedeno výše, vlastní implementace by byla značně neefektivní a zvýšila by náročnost a tím i délku zpracování. U mobilních zařízení navíc při náročných výpočtech rapidně roste spotřeba energie a klesá pohotovostní doba. Právě výdrž na baterii je přitom největší slabinou současných chytrých telefonů. V praxi pak k natočení dopravního značení dochází jen velmi zřídka, u kruhových dopravních značek jej pak nemůžeme detekovat vůbec. Proto je tento fakt ve výsledné aplikaci zanedbán.

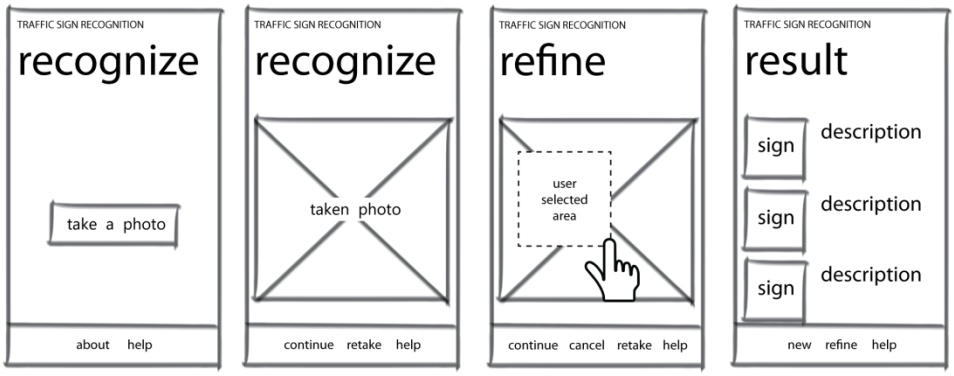
5.2 Grafické rozhraní

Grafické rozhraní aplikace bylo navrženo s ohledem na maximální sjednocení vzhledu s nativním vzhledem systému WP7. Díky tomu bude uživatel intuitivně očekávat umístění ovládacích prvků i jejich vzhled, což usnadní práci s aplikací a zároveň také zvýší rychlost a jednoduchost její obsluhy. Naše aplikace také nevyžaduje velké množství ovládacích prvků.

Na úvodní obrazovce nalezneme tlačítka pro pořízení či výběr fotografie. Ve spodní části obrazovky se pak nachází typický prvek uživatelského rozhraní Windows Phone 7, tzv. Application Bar. Sem soustřeďujeme veškeré prvky vyžadované pro ovládání aplikace. Na úvodní obrazovce se jedná o tlačítka zobrazující nápovědu a informace o aplikaci. Třetím tlačítkem, které se aktivuje po získání zdrojového obrázku, se spustí samotný proces rozpoznávání. Tlačítko pro znovupořízení fotografie bylo vypuštěno, protože tato možnost je uživateli nabízena přímo při procesu pořizování fotografie, kdy je nutno fotografii potvrdit. Posledním ovládacím prvkem je tlačítko zamítnutí fotografie, které uživatele vrátí na první obrazovku.

Prostředí systému Windows Phone 7 nabízí více možností jak pracovat s několikastránkovými aplikacemi. Vedle klasické možnosti přechodu z jedné stránky na druhou existují také dvě unikátní možnosti Pivot a Panorama. V režimu Pivot se pod titulkem aplikace zobrazují vedle sebe všechny podtitulky stránek a klepnutím na jejich název je uživatel přenesen na další stránku aplikace. To vše ve zcela plynulé horizontální animaci. Druhou možností je Panorama, kdy je obsah stránky rozprostřen nejen vertikálně, ale i horizontálně. Rozdíl oproti režimu Pivot je v tom, že Panorama je jediná stránka s možností horizontálního posuvu, zatímco Pivot nabízí seznam stránek, mezi kterými může uživatel cyklicky přecházet. Uživatel mezi stránkami přechází intuitivně pomocí tažení prstu horizontálně po obrazovce, přičemž při přechodu z poslední stránky je navrácen na stranu první.

V naší aplikaci si ovšem vystačíme s běžným režimem jednoduchých stránek, jelikož nezobrazujeme velké množství informací. Využít bychom jej mohli například při zobrazení katalogu dopravních značek, který bychom mohli rozdělit do kategorií a každé přiřadit jednu stránku typu Pivot. Tuto možnost ovšem v naší aplikaci implementovat nebudeme.



Obrázek 12: Prvotní návrh uživatelského rozhraní

Při prvotním návrhu bylo počítáno i s možnými uživatelskými zásahy do procesu detekce. V případě, že aplikace v obraze žádnou značku nedetekuje, by byl uživatel vyzván k manuálnímu upřesnění oblasti, kde by se značka měla nacházet. S postupujícím vývojem ale bylo rozhraní lehce pozměněno. Třetí obrazovka byla zcela vypuštěna z důvodu zbytečnosti vzhledem k použité metodě detekce. Jakékoliv uživatelské upřesnění totiž nemá

Obrázek 13: Výsledná podoba uživatelského rozhraní

na náš způsob vyhledání značek nejmenší vliv, pouze by zmenšilo oblast pro vyhledávání, ale výsledky z této oblasti by byly totožné.

Po úspěšné detekci je pak uživateli prezentován seznam výsledků. Každý z těchto výsledků se pak skládá z výseku fotografie a jejího odpovídajícího popisu.

5.3 Reprezentace datových struktur

V naší aplikaci budou použity pro reprezentaci některých objektů speciální třídy, konkrétně se jedná o třídy *TrafficSignObject*, *TrafficSignTemplate* a *TrafficSignResult*.

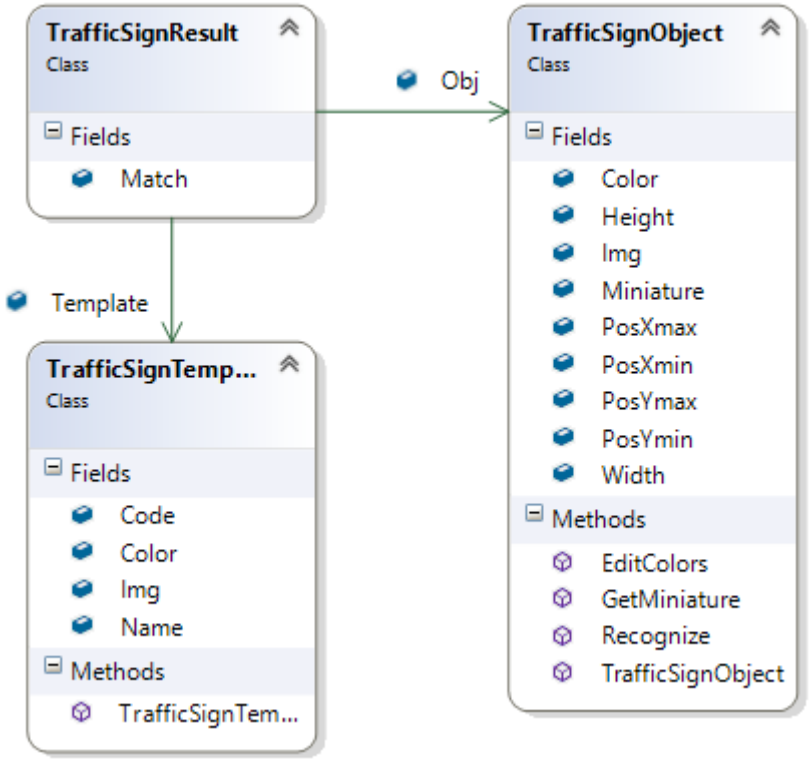
TrafficSignObject reprezentuje detekovaný objekt pomocí údajů o jeho souřadnicích a rozměrech, dále obsahuje výsek fotografie obsahující daný objekt a zpracovanou miniaturu určenou pro porovnání se vzorem. Třída také obsahuje metody *GetMiniature*, pomocí které je

výsek fotografie zmenšen na požadované rozměry, a *EditColors*, která upravuje hodnoty jednotlivých pixelů miniatury pro porovnávání se vzorem. Tyto metody budou později důkladněji popsány v kapitole věnované implementaci (kap. 6).

TrafficSignTemplate je vzorem, se kterým detekované objekty porovnáváme. Obsahuje údaje o dané dopravní značce spolu s její grafickou podobou pro porovnávání.

TrafficSignResult pak tvoří vlastní výsledek klasifikace a obsahuje odpovídající pár *TrafficSignObject* a *TrafficSignTemplate* a hodnotu výsledku porovnávání.

Poslední třídou je třída *Constants*, která obsahuje konstanty využívané při běhu programu.



Obrázek 14: Speciální třídy reprezentující objekty

6 Implementace

Následující praktická část této práce se zabývá převedením navržených myšlenek do praxe. Zveřejněním hotového produktu na Marketplace také můžeme zjistit, zda je o takový typ aplikace zájem. Případná analýza trhu pomocí monitoringu zájmu o jednoduché aplikace tak může poskytnout cenné údaje o segmentaci pro případné podobné složitější a dokonalejší komerční produkty.

6.1 Pořízení a zpracování fotografie

Aplikace umožňuje dva způsoby získání zdrojových obrazových dat. Prvním a nejspíše také nejvíce využitelným způsobem je přímé pořízení fotografie z mobilního zařízení, druhým způsobem je pak výběr již dříve vyfotografované scény z galerie mobilního telefonu.

Ať už se jedná o pořízení či výběr fotografie, systém telefonu implementuje obě tyto akce přímo ve svém API a velmi tak usnadňuje jejich použití. Pro využití fotoaparátu přístroje je definována akce *CameraCaptureTask*, která přepne telefon do režimu pořízení fotografie a po úspěšném vyfotografování obraz uloží do objektu *WriteableBitmap*, se kterým můžeme dále pracovat. Pro výběr fotografie z galerie obrázků slouží analogická akce *PhotoChooserTask*. Uživatel si na domovské stránce aplikace pouze vybere, kterou metodu chce využít.

Využití akce *CameraCaptureTask* s sebou ovšem přináší skrytý problém, který vyplynul až z testování aplikace na reálném zařízení. Fotoaparát telefonu využívá informací z akcelerometru a automaticky tak otáčí fotografii podle polohy telefonu při focení. Při využití *CameraCaptureTask* na tuto informaci ovšem není brán zřetel a fotografie je vždy uložena „na šířku“, tak jak by byla vyfocena „přirozeně“ s prstem na boční spoušti, kterou WP7 telefony mají. Předem vyfocené fotografie, které jsou do aplikace načteny z galerie obrázků, tímto neduhem netrpí a jsou vždy natočeny správně. Řešením je získání dat o správném natočení z EXIF informací uložených u každé fotografie. Na základě této informace pak fotografii vhodně otočíme. K vyřešení tohoto problému jsem využil již existující knihovny *ExifLib*, kterou přímo pro tento účel ještě dále upravil Tim Heuer, jehož řešení jsem v aplikaci využil [34].

Objekt *WriteableBitmap*, který je ve Windows Phone 7 využíván pro reprezentaci obrazových dat, pro jejich popis využívá vlastnosti *Pixels*, což je jednorozměrné pole integerů. Hodnota každého pixelu je tak uložena na 32 bitech, a to ve formátu ARGB[6]. To

[6] A – alpha (průhlednost), R – red, G – green, B – blue

znamená, že pro každou barevnou složku obrazu jsou vyhrazeny dva byty, ke kterým ovšem nelze přistupovat přímo. Intenzity jednotlivých barevných složek pro konkrétní pixel tak musíme z jeho číselné hodnoty extrahovat následujícím způsobem:

```
byte A = (byte)((pixel & 0xFF000000) >> 24);
byte R = (byte)((pixel & 0x00FF0000) >> 16);
byte G = (byte)((pixel & 0x0000FF00) >> 8);
byte B = (byte)(pixel & 0x000000FF);
```

Jednorozměrné pole *Pixels* také neumožňuje přistupovat k bodům obrazu pomocí souřadnicového systému. Pokud tedy takový přístup budeme potřebovat, je nutné získat tyto souřadnice z hodnoty indexu pole *i*. Nejjednodušším způsobem je využití další z hodnot objektu, a to *PixelWidth*, kdy:

```
int x = i % Image.PixelWidth;
int y = i / Image.PixelWidth;
```

6.2 Výběr kandidátních objektů

Jak již bylo uvedeno v popisu návrhu aplikace, pro detekování dopravních značek budeme využívat právě barevných hodnot jednotlivých segmentů obrazu. První akcí bude tedy detekce barevných oblastí ve fotografii. Detekovat budeme zákazové a výstražné (tedy červené) značky. Pro uložení dat použijeme opět objekt *WriteableBitmap*.

6.2.1 Detekce barevných objektů

Při detekci bodů, které by mohly být součástí objektů dopravních značek, procházíme obrazem pixel po pixelu a pomocí postupu uvedeného v kapitole 6.1 z něj extrahujeme informace o barevných složkách. Vzhledem k použití barevného modelu HSV je tyto hodnoty převést podle vzorců (3.1, 3.2, 3.3). Abychom ušetřili čas a eliminovali zbytečné výpočty, při nízkých hodnotách jasu a sytosti nepokračujeme ve výpočtu odstínu, což je nejnáročnější část. Abychom bod obrazu považovali za červený, tedy za součást potenciální značky, musí splňovat následující kritéria (konkrétní hodnoty byly zvoleny na základě prvotního testování aplikace):

Hue (0-360)	**Saturation** (0-255)	**Value** (0-255)
0-29, 324-360	>100	>25

Tabulka 1: Hodnoty pro segmentaci červených objektů

Na základě těchto hodnot pak určujeme, zda se jedná o bod některého z požadovaných objektů. Hodnoty musí splňovat minimální hodnotu intenzity požadované barvy a zároveň poměrové kritérium charakteristické pro objekt dané barvy. Výsledný typ bodu pak ukládáme do objektu *RedBinaryImage*.

Obrázek 15: Výsledek detekce červené barvy v obraze

6.2.2 Získání objektů z obrazu

Po průchodu původním obrazem a detekcí jednotlivých barevných bodů postoupíme do fáze vlastního výběru vhodných kandidátních objektů.

Postupně procházíme po sloupcích zleva doprava a shora dolů polem *BinaryImage* a jakmile narazíme na jakýkoliv barevný bod detekovaný v předchozím postupu, je spuštěn algoritmus, pomocí kterého získáme maximální a minimální souřadnice objektu. Tento algoritmus funguje na stejném principu jako rekurzivní algoritmus semínkového vyplňování. Při splnění podmínky (zde stejná hodnota jako semínkový bod) se bod sám stává dalším semínkem a jeho hodnota je vynulována, aby nebyl stejný objekt detekován vícekrát. Souřadnice bodu jsou pak porovnávány s dosavadními extrémy:

```
Flood (int color, int x, int y) {
    int i = y * BinaryImage.PixelWidth + x;
    if (BinaryImage.Pixels[i] == color) {
        BinaryImage.Pixels[i] = 0;
        if (x > xmax) xmax = x;
        if (y < ymin) ymin = y;
        if (y > ymax) ymax = y;
        Flood(color, x, y - 1);
        Flood(color, x + 1, y);
        Flood(color, x, y + 1);
    }
}
```

Pro detekovaný objekt jsme se rozhodli použít požadavek na minimální šířku a výšku 100 pixelů, což zaručuje dostatečné rozlišení pro porovnání drobných vzorů, jako jsou piktogramy na značkách. Navíc tím eliminujeme množství menších barevných objektů, které v obraze můžeme detekovat. Vzhledem k minimálnímu rozlišení fotoaparátů přístrojů se systémem WP7, které činí v současné době 5 Mpx, jde o rozměr více než dostačující. Model, na kterém byla aplikace testována, měl dokonce fotoaparát s rozlišením 8 Mpx. S rostoucím rozlišením fotoaparátu tak získáváme také možnost fotografovat značky z větší vzdálenosti.

Nakonec však maximální rozlišení fotoaparátu nepoužíváme, což bude vysvětleno v kapitole věnované testování (kap. 7). Druhou podmínkou, kterou musí detekovaný objekt splňovat, je maximální poměr stran 2:1. Tento poměr je dán tvarem dopravních značek s přihlédnutím k možné perspektivní deformaci. Předem tak vyřadíme množství dalších objektů. Pokud náš objekt obě tyto podmínky splňuje, je přidán do seznamu kandidátů.

6.3 Klasifikace objektů

Závěrečnou fází celého postupu je úprava barev a porovnání detekovaného objektu s uloženou databází dopravních značek pro získání informace, o kterou se jedná, případně jeho zahození při nedostatečné shodě.

6.3.1 Úprava objektů před porovnáním

V našem případě se jedná o jednoduché porovnání se vzorem, porovnáváme tedy objekty pixel po pixelu s předlohou. Nejprve je nezbytné zdrojová data upravit tak, aby toto porovnání bylo možné.

Prvním krokem je úprava velikosti objektu na rozměr 100 x 100 pixelů. V oblasti počítačové grafiky takovou operaci nazýváme změnou měřítka (scale) a nové souřadnice bodů získáváme následujícím způsobem [35]:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (6.1)$$

kde S_x a S_y jsou poměry jednotlivých rozměrů nového a původního objektu. V našem případě tedy:

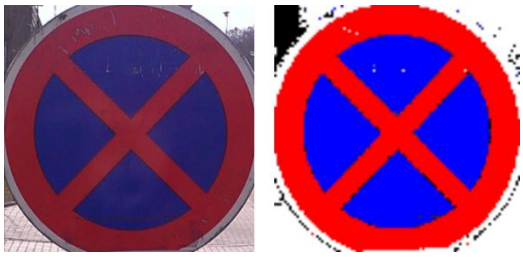
$$S_x = \frac{100}{Object.width} \quad (6.2)$$

$$S_y = \frac{100}{Object.height} \quad (6.3)$$

Následujícím krokem je úprava jednotlivých pixelů, abychom mohli porovnat barevné hodnoty na přesnou shodu. Jedná se o podobný proces jako ten, který využíváme v prvotní fázi zpracování fotografie. V případě, že bod splňuje předem dané vlastnosti, je jeho barva změněna. Jediným rozdílem je rozšíření množiny barev, které zpracováváme. Takto tedy v objektu zvýrazňujeme všechny černé, bílé, červené, modré, zelené a žluté body. Body, které neodpovídají žádné z uvedených kombinací vstupních hodnot, zůstávají nedotčeny a v důsledku na ně nebude při porovnání se vzorem brán zřetel. Prahové hodnoty pro barvy byly zvoleny na základě výsledků prvotního testování. Hodnoty pro odstín (Hue) jsou zvoleny tak, aby pokryly širší spektrum barev. Důvodem je požadavek na co možná největší odolnost detekce vůči různému osvětlení a vybledlým barvám značek.

Výsledná barva (RGB 0-255)	**Požadované hodnoty**		
	Hue (0-360)	**Saturation** (0-255)	**Value** (0-255)
Bílá (255, 255, 255)	0-360	< 100	> 120
Černá (0, 0, 0)	0-360	< 100	< 120
Červená (255, 0, 0)	0-29, 324-360	> 100	> 60
Žlutá (255, 255, 0)	40-130	> 100	> 60
Zelená (0, 255, 0)	140-190	> 100	> 60
Modrá (0, 255, 0)	190-260	> 100	> 60

Tabulka 2: Barevné hodnoty pro přebarvování objektů



Obrázek 16: Vyříznutý a přebarvený objekt

6.3.2 Porovnání se vzorem

Nyní, když máme vybrané a do vhodné podoby upravené kandidátní objekty, můžeme přistoupit k vlastnímu porovnávání s uloženými vzory.

Pro naši aplikaci bylo do databáze uloženo 77 výstražných a zákazových dopravních značek používaných na území České republiky doplněných o vybrané značky upravující přednost v jízdě. Vzory vychází ze značek uvedených v databázi na internetu [36]. Jejich grafická podoba byla upravena, aby umožňovala testování přesné shody, a uložena v šestibarevném PNG obraze o rozměrech 100 x 100 pixelů. Formát PNG byl zvolen díky tomu, že je bezztrátový. Máme tedy jistotu, že jednotlivé pixely budou mít opravdu takovou hodnotu, jakou jsme jim při vytváření vzoru přidělili. Asi nejrozšířenější grafický formát JPEG jsme z tohoto důvodu vyloučili, při kompresi se totiž mohou hodnoty jednotlivých bodů změnit a znemožnit porovnání na přesnou shodu. U formátu PNG navíc můžeme snadno určit barevnou hloubku, a tím i velikost souboru. Další alternativou je grafický

formát GIF, který ovšem systém Windows Phone nedokáže nativně dekódovat a nahrávání vzorů do aplikace by se převodem značně prodloužilo.

Protože detekovaný obraz značky obsahuje také pozadí, nemůžeme porovnávat všechny pixely obrazu. Porovnání tedy probíhá ve dvou částech. Vzhledem k tomu, že vzor značek je uložen na bílém podkladu, postupujeme po řádcích zleva do poloviny obrazu, přičemž vlastní porovnávání se zahájí, jakmile algoritmus narazí na první nebílý bod vzoru na konkrétním řádku, tedy okraj značky. Po porovnání levé poloviny obrazu se postupuje analogicky zprava doleva na pravé polovině. Tento postup popisuje také následující pseudokód:

```
// left half
for (rows in image) {
    compare = false;
    for (columns from 0 to 49) {
        if (compare == false) {
            if (templatepixel != WHITE)) compare = true;
            else continue;
        }
        cnt++;
        if (objectpixel == templatepixel) cntOk++;
    }
}

// right half
for (rows in image) {
    compare = false;
    for (columns from 99 to 50) {
        if (compare == false) {
            if (templatepixel != WHITE)) compare = true;
            else continue;
        }
        cnt++;
        if (objectpixel == templatepixel) cntOk++;
    }
}
```

Po porovnání vypočítáme podobnost objektu s daným vzorem jako podíl shodných a všech porovnaných pixelů

$$MATCH = \frac{cntOk}{cnt} \cdot 100$$

Takto postupujeme pro všechny vzorové obrazy. V případě, že je vypočítaná hodnota *MATCH* vyšší než dosavadní nejlepší shoda, upravíme hodnoty v objektu *TrafficSignResult* (viz kapitola 5.3) a postoupíme na porovnání s dalším vzorem.

Abychom poskytli uživateli pouze kvalitní výsledky, do konečného seznamu zahrneme pouze takové objekty, u nichž hodnota *MATCH* přesáhla předem danou hraniční hodnotu *X*. Na základě testování byla jako základní hranice *X* zvolena hodnota 60 %, která je využita pro většinu značek. Pro některé případy však byly zvoleny lehce odlišné hodnoty.

Speciální hodnoty mají značky „B 2 – Zákaz vjezdu všech vozidel“ a „P 6 – Stůj, dej přednost v jízdě“, pro ty platí práh 75 % a pro značku „B 1 – Zákaz vjezdu všech vozidel (v obou směrech)“ je podmínkou shoda 90 % pixelů.

Výsledky rozpoznávání jsou pak uživateli prezentovány na poslední obrazovce v podobě seznamu výřezů se značkou, které jsou doprovozeny označením a názvem. Byl zvažován i nápad doplnit výsledek slovním popisem značky, který můžeme nalézt ve sbírce zákonů, nakonec od něj ale bylo upuštěno. Popis by znepřehlednil výsledný seznam a zahltil uživatele zbytečnými informacemi. Tato možnost by mohla být doplněna v dalších rozšířeních aplikace (viz kap. 8). Jediným ovládacím prvkem na této stránce je tlačítko návratu na domovskou stránku, kde je uživateli umožněno spustit novou detekci.

7 Testování aplikace

7.1 Testování výkonu

Jak bylo zmíněno v kapitole 1.5, Microsoft poskytuje vývojářům možnost důkladně testovat výkonovou stránku vytvořené aplikace pomocí Windows Phone Performance Analysis. Systém WP7 ale při připojení k počítači se synchronizačním softwarem Zune zablokuje přístup ke knihovnám, a to jak k hudbě, tak především k obrázkům. Běh programu Zune je přitom vyžadován pro umožnění nahrání aplikace i její debugging. Protože je blokováno i použití fotoaparátu, zdálo se, že aplikace nebudeme moci otestovat. Existuje však utilita WPConnect, která je skrytá v balíku Windows Phone SDK a umožňuje použití Media API i při připojení zařízení k počítači a debugovat tak i aplikace pracující s médii.

Testy byly prováděny na fotografiích o rozměru 1600 x 1200 pixelů, tedy při rozlišení dvou megapixelů. Aplikace zvládne pracovat s fotografiemi libovolných rozměrů, s rostoucími rozměry ovšem roste také časová náročnost zpracování. Zpracování fotografií použitých při testování trvá asi sekundu a půl.

7.2 Testování přesnosti klasifikace

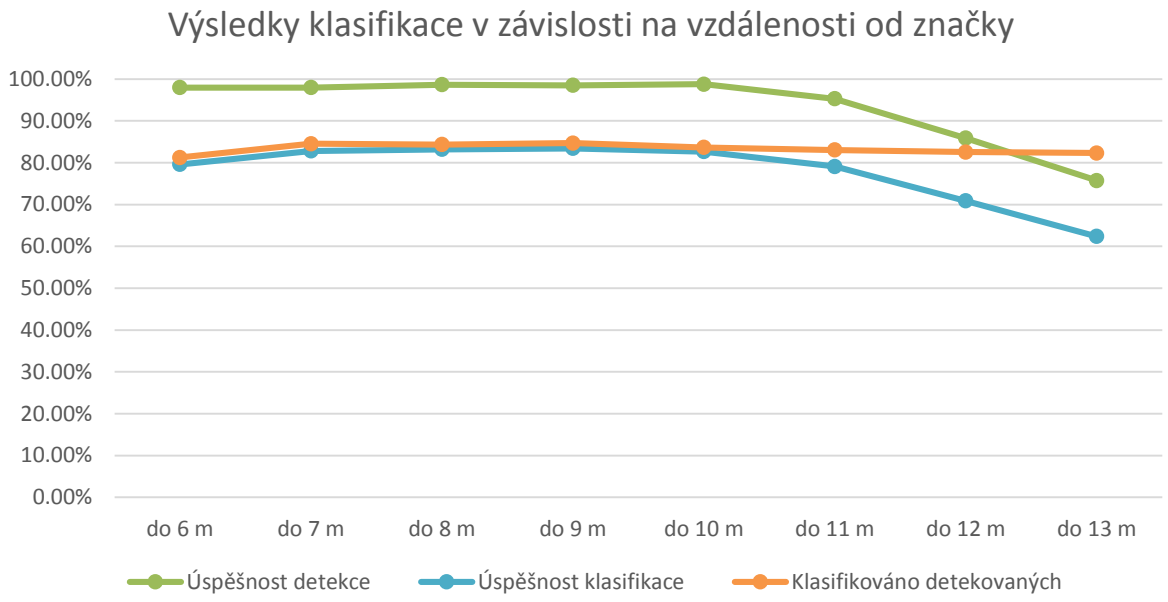
Vedle ověření funkčnosti všech možných průběhů aplikace je nejdůležitějším úkolem testovací fáze zjistit, zda aplikace opravdu dělá to, co od ní vývojářský tým a především uživatel očekává. V našem případě je otázkou, zda opravdu dokáže správně detekovat a rozpoznat dopravní značky. Jedinou možností, jak zjistit odpověď, je vyzkoušet aplikaci v terénu. V našem případě testování proběhlo na sadě fotografií pořízených především v centru Brna. Pochopitelným důsledkem testů v reálném prostředí je nemožnost vyzkoušet rozpoznání opravdu všech značek, protože není snadné všechny nalézt. Zároveň však důsledně otestujeme alespoň ty značky, které se v provozu nejčastěji vyskytují.

Testovány byly dva parametry, maximální vzdálenost a maximální odchylka od svislé osy, kdy je aplikace schopna ještě značku úspěšně detekovat. V každém bloku bylo testování podrobeno 50 různých dopravních značek. Celkem testovací sada obsahuje 502 fotografií, které jsou umístěny na datovém nosiči přiloženém k této práci.

Omezení vyplývající z implementace jsou minimální rozměry a poměr stran detekovaného objektu. Pro naši aplikaci musí objekt reprezentující značku mít na fotografii alespoň 100 pixelů na výšku a 100 pixelů na šířku, poměr stran přitom nesmí přesáhnout 2:1. Při současných možnostech digitálních fotoaparátů je taková velikost snadno dosažitelná. Přestože pro testování bylo zvoleno nižší rozlišení, testovací přístroj (HTC 7 Mozart) disponoval fotoaparátem o rozlišení až 8 MPx.

7.2.1 Maximální vzdálenost

Pro testování maximální možné vzdálenosti, byly značky fotografovány ve vzdálenostech 6 až 13 metrů. Splnění podmínek uvedených výše se liší pro jednotlivé typy značek. Trojúhelníkové výstražné značky mají větší rozměry, než značky kruhové, ve výsledku je tedy teoreticky možné je detekovat z větší vzdálenosti. Tento předpoklad se potvrdil, aplikace je detekovala až do vzdálenosti asi 11 m, zákazové značky kruhového tvaru pak do 10 m. V některých případech byly tyto limitní vzdálenosti i překonány, to ovšem můžeme přičíst drobným odchylkám ve velikostech konkrétních značek.



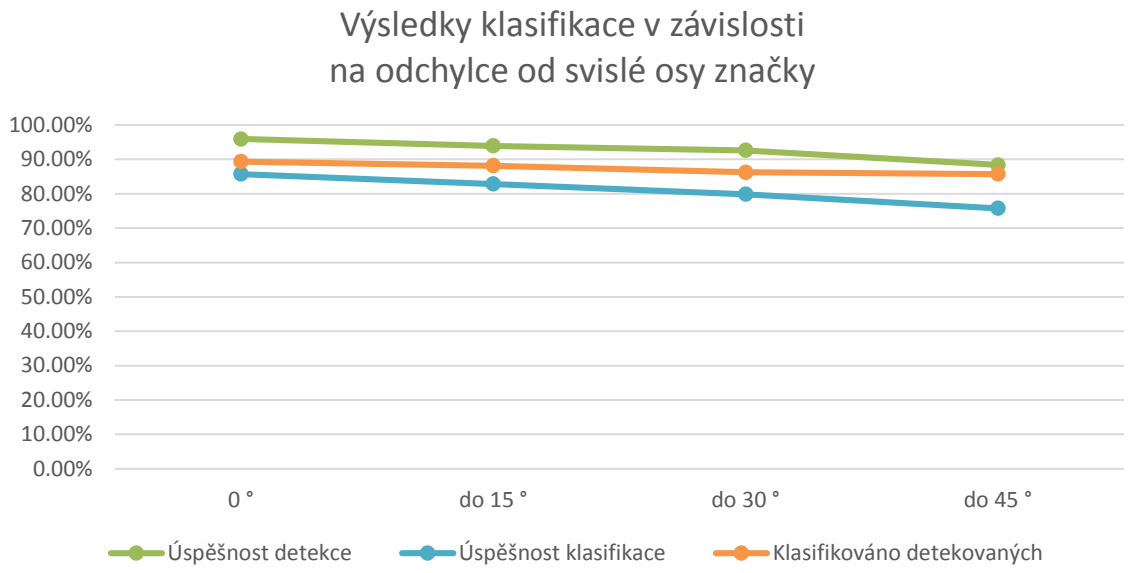
Obrázek 17: Graf výsledků klasifikace v závislosti na vzdálenosti

Můžeme tak prohlásit, že aplikace spolehlivě detekuje dopravní značky do vzdálenosti 10 m, dosažená úspěšnost kvalifikace činí 82,6 %. V případě většího odstupu může být značka přiblížena, ale vzhledem k absenci optického zoomu u mobilních fotoaparátů musíme využít zoomu digitálního, který značně znehodnocuje obraz a dochází tak k destrukci jemných detailů v piktogramech značky. Pro jednodušší značky je jeho využití možné, pro všeobecné použití je ale nevhodný.

7.2.2 Úhel záběru

Značky nemusí být vždy spolehlivě natočeny kolmo ke směru jízdy, proto je zapotřebí zjistit, z jakého úhlu jsme ještě schopni značku správně rozpoznat. Uvažovat musíme především natočení podle svislé osy, podle vodorovné osy značky natočeny nebývají. Jediným problémem je deformace obrazu vlivem perspektivy při pořizování z podhledu, s jejich pozicí vzhledem k řidičům či jiným účastníkům silničního provozu je při umisťování počítáno a při pořízení fotografie z „normální“ pozice by neměl nastat žádný problém.

Při fotografování z úhlu musíme stále dbát na dodržení minimální šířky značky. Díky perspektivě se totiž její šířka na fotografii rychle zmenšuje. Při vyhodnocení testů musíme brát v potaz tvar značky. Kruhové se vlivem perspektivy deformují méně než čtvercové a trojúhelníkové. Druhým faktorem je pak komplexnost značky, jednoduché značky snesou mnohem větší natočení než značky se složitými piktogramy. Ve výsledku ovšem testy dopadly velmi dobře, přibližně 80% značek bylo správně klasifikováno i z úhlu 30 °. Pro praktické použití je ovšem doporučeno nepřesáhnout odchylku 15 °, a to z důvodu zužování objektu na fotografii a tím pádem klesající maximální možné vzdálenosti pro úspěšnou detekci.



Obrázek 18: Graf výsledků klasifikace v závislosti na odchylce

7.2.3 Světelné podmínky

Pro úspěšné detekování a rozpoznání dopravní značky požaduje naše aplikace maximální možné zachování barevné informace. V případě příliš tmavého obrazu se tato informace vytrácí a objekt pak není detekován. Toto může být u dopravních značek problém. Fotoaparáty mobilních telefonů většinou disponují pouze automatickým měřením expozice, nejčastěji pracují s průměrem jasu v celém obraze. Vzhledem k tomu, že značky nejčastěji zabíráme z podhledu, velkou část fotografie tvoří obloha. V případě jasného počasí tak bývá fotografie neúnosně tmavá. Telefon, na kterém byla aplikace testována, umožňoval nastavení středového, případně i bodového měření expozice, čímž se barevné podání fotografií výrazně zlepšilo. Díky využití barevného modelu HSV si aplikace poradí i s barevně utlumenými fotografiemi pořízenými v protisvětle. Větším problémem než nalezení barev je pak ale nalezení vhodného prahu mezi bílou a černou barvou.

Druhým extrémem jsou pak noční podmínky. Fotoaparáty mobilních telefonů neobsahují objektivy s velkou světelností. Tento jev při nedostatku osvětlení scény

kompenzují především zvýšením citlivosti. Spolu se zvýšením citlivosti ale výrazně roste zašumění obrazu. Pro vyloženě noční snímky jsou telefony stále častěji vybavovány i bleskem, a to nejčastěji ve formě jedné nebo několika LED. Jak ale vyplynulo z testování, při fotografování dopravních značek příliš nápomocný není. Povrch značek je pokryt reflexní vrstvou, která zvyšuje jejich viditelnost a čitelnost při osvětlení reflektory projíždějících automobilů. LED blesk telefonu je ovšem natolik výkonný, že v kombinaci s automaticky zvýšenou citlivostí při nočních záběrech ze značky na fotografii zobrazí pouze zářivý bílý objekt. Jelikož pracujeme s barvami a bez nich nedokážeme značku ani detekovat, natož rozpoznat, výsledný efekt je naprosto nulový. Velmi záleží na pozici pozorovatele vůči značce. Zatímco při postavení přímo proti značce dojde k výše popsanému jevu, při lehké odchylce je již barevné podání v pořádku. V případě umístění telefonu do automobilu by

Obrázek 19: Fotografie s použitím blesku

tedy teoreticky aplikace mohla fungovat dobře.

Nejlepší podmínky pro použití aplikace se v podstatě shodují s ideálními fotografickými, tedy oblačný den, případně focení se sluncem v zádech, kdy je kontrast a barevné podání značek nejlepší.

7.2.4 Výsledky testování

Z testování vyplynulo, že naše aplikace spolehlivě pracuje do vzdálenosti deseti metrů od značky a poradí si i s fotografiemi pořízenými z úhlu. Celková úspěšnost algoritmu činí asi 82,5 %. Segmentace červené barvy je téměř stoprocentní, větším problémem je nalezení vhodného prahu mezi černou a bílou barvou v piktogramech značky. Vhodným řešením problému by bylo využití adaptivního prahování jasové složky místo stálého prahu, kterým je v současné verzi hodnota 120 (viz kapitola 6.3.1). K falešné detekci došlou pouze ve dvou případech, vždy šlo o červený objekt (automobil a vrata), který byl zaměněn za značku „B 2 – Zákaz vjezdu všech vozidel“.

8 Pokračování v práci a možná rozšíření

Práci by bylo v budoucnosti možné rozšířit či upravit hned několika způsoby.

Mohli bychom přidat podporu ostatních dopravních značek, tedy příkazových a informativních, spolu s katalogem značek, který by posloužil především pro studijní účely vzhledem k tomu, že velké množství značek člověk poprvé a naposledy uvidí v autoškole.

Dalším, a asi nejvíce využitelným rozšířením, by byla implementace úspornějšího algoritmu, který by umožnil detekování nikoliv ve fotografiích, ale přímo v náhledu fotoaparátu v reálném čase. Tato tzv. „live detekce" by našla využití například v propojení s elektronikou automobilu nebo GPS senzorem telefonu. Při detekování značky „Nejvyšší povolená rychlost" by například mohla aplikace řidiče upozornit na její překročení, na přehlédnutí zákazu vjezdu či jiné události nebo nebezpečí. V současnosti podobné systémy nabízí především vozy prémiových značek jako Mercedes, Audi či BMW, pomalu se ale začínají objevovat i v levnějších automobilech. Detekci a upozorňování pomocí mobilního telefonu by však mohl využít každý vlastník WP7 telefonu nezávisle na řízeném autě.

Detekované značky by také bylo možné ukládat do paměti telefonu spolu s GPS údaji o místě pořízení a vytvářet tak offline databázi, kterou by bylo možné využívat při jízdě v automobilu.

V aplikaci také můžeme rozšířit databázi vzorů i o dopravní značky jiných zemí než aktuálně podporované České republiky a umožnit tak její využití v zahraničí. Stát by mohl uživatel určit manuálně nebo opět vycházet z GPS souřadnic zaznamenaných telefonem. Balíčky vzorů by mohly být uvolňovány jako rozšíření k aplikaci, podobně jako mapy u navigačních aplikací.

Samozřejmě bychom také mohli upravit a vyladit vyhledávací algoritmus pro dosažení přesnějších a rychlejších výsledků, detekovat a zpracovávat i menší objekty, což by zvětšilo vzdálenost, ze které je možné značky detekovat, a podobně.

Závěr

Cílem teoretické části práce bylo seznámit se s vývojem aplikací pro Windows Phone 7, s metodami segmentace obrazu a klasifikace a také s dopravními značkami na území České republiky. Nejprve jsem se pokusil najít případné aplikace, které by se zabývaly podobnou problematikou, žádné jsem ale v obchodech nenalezl. Při vlastním vývoji aplikace jsem nastudoval různé metody zpracování obrazu a klasifikace objektů. Počáteční segmentace obrazu využívá vlastností barevného modelu HSV. Výběr kandidátních objektů byl proveden modifikovaným algoritmem semínkového vyplňování a závěrečná klasifikace využívá porovnání se vzorem. Pro implementaci bylo nutné osvojit si základy práce s obrazovými daty v prostředí Windows Phone a převedení výše uvedených algoritmů do odpovídající podoby. Grafické rozhraní jsem navrhl s důrazem na co nejpřirozenější začlenění do prostředí systému, abych uživatelům usnadnil ovládání aplikace.

Hlavní výhodou a přínosem aplikace je především její osamocená pozice na trhu, neboť nikdo z konkurentů podobnou aplikaci ve svých obchodech nenabízí. Aplikace je plně automatická a nevyžaduje od uživatele žádné zásahy při procesu detekce, stačí pouze pořídit zdrojovou fotografii. Výsledek pak závisí pouze na kvalitě této fotografie. Kromě otáčení obrazu podle natočení přístroje při fotografování jsou veškeré využité funkce vlastní, ke komunikaci s fotoaparátem pak slouží API systému WP7. Aplikace není nijak omezena počtem značek v obraze.

Velký důraz jsem kladl na důkladné otestování schopností aplikace z hlediska vstupů a odpovídajících výstupů, stejně jako na prověření všech možných stavů, do kterých by se aplikace mohla dostat. Při testování v terénu byla určena maximální doporučená vzdálenost 10 metrů a maximální odchylka od svislé osy značky 15 °. Při zohlednění těchto limitů dosahovala aplikace 95% úspěšnosti detekce a úspěšnost klasifikace přesáhla 80 %. Po otestování jsem aplikaci zveřejnil na Windows Phone Marketplace, kde je nyní aplikace zdarma dostupná ke stažení pod názvem „*TrafficSignRecognizer*“. Schválení ze strany Microsoftu proběhlo bez problému.

Na závěr jsem uvedl několik možných rozšíření aplikace, které nebyly požadovány v rámci zadání, a které by bylo možno implementovat v případné další práci na tomto projektu. Mimo jiné jsem navrhl například možnost implementace „live detekce“, spolupráce s GPS či doplnění rozšiřujících balíčků pro jiné státy.

Použité zdroje

1. Microsoft Corporation. Windows Phone Development. [Online] 15. 12 2011. http://msdn.microsoft.com/en-us/library/ff402535(v=vs.92).aspx.

2. —. Developing a Windows Phone Application from Start to Finish. [Online] 21. 10 2011. http://msdn.microsoft.com/en-us/library/gg680270(v=pandp.11).aspx.

3. Lee, Henry a Chuvyrov, Eugene. Beginning Windows Phone 7 Development. Second Edition. New York : Apress, 2011. ISBN 978-1-4302-3596-5.

4. Petzold, Charles. Programming Windows Phone 7. One Microsoft Way, Redmond, Washington 98052-6399 : Microsoft Press, 2010. ISBN 978-0-7356-4335-2.

5. Russ, John C. The Image Processing Handbook. 4th ed. Boca Raton : CRC Press, 2002. ISBN 0-8493-1142-X.

6. Šonka, M., Hlaváč, V. a Boyle, R. Image Processing, Analysis and Machine Vision. 3. vydání. Toronto : Thomson, 2008. ISBN 978-049-5082-521.

7. Gonzalez, Rafael C. a Woods, Richard E. Digital Image Processing. 3rd edition. Upper Sadle River, New Jersey 07458 : Pearson Prentice Hall, 2008. ISBN 0-13-168728-X.

8. Haralick, Rober M. and Shapiro, Linda G. Computer and Robot Vision. New York : Addison-Wesley Publishing Company, 1993. Vol. II. ISBN 0-201-56943-4.

9. Vyhláška Ministerstva dopravy a spojů ze dne 10. ledna 2001, kterou se provádějí pravidla provozu na pozemních komunikacích a úprava a řízení provozu na pozemních komunikacích. Sbírka zákonů, Česká republika. 2001. 11. http://www.mdcr.cz/NR/rdonlyres/DFE87D07-9E39-467D-95F9-35D3AB525369/0/MicrosoftWord30.pdf.

10. Chen, Brian X. Microsoft Blends Zune Media, Xbox Live Into New Phone OS. Wired.com. [Online] 15. 2 2010. http://www.wired.com/gadgetlab/2010/02/microsoft-phone/.

11. Microsoft Corporation. Nokia and Microsoft Announce Plans for a Broad Strategic Partnership to Build a New Global Mobile Ecosystem. Microsoft News Center. [Online] 10. 2 2011. http://www.microsoft.com/en-us/news/press/2011/feb11/02-11partnership.aspx.

12. Savov, Vlad. Microsoft announces Windows Phone 'Mango' update, coming in autumn (updated). Engadget.com. [Online] 24. 5 2011. http://www.engadget.com/2011/05/24/microsoft-announces-windows-phone-mango-update-early-and-in/.

13. Carmody, Tim. Windows Phone 7 Is the Real Facebook Phone. [Online] 14. 10. 2010. http://www.wired.com/gadgetlab/2010/10/windows-phone-7-is-the-real-facebook-phone/.

14. Svetlik, Joe. Windows Phone Store hits 150,000 apps, doubled in last year. CNET UK. [Online] 31. 12. 2012. [Citace: 18. 3. 2013.] http://crave.cnet.co.uk/software/windows-phone-store-hits-150000-apps-doubled-in-last-year-50010072/.

15. Google Play Store: 800,000 apps and overtake Apple AppStore! RRSPhone.com. [Online] 16. 1. 2013. [Citace: 18. 3. 2013.] http://www.rssphone.com/google-play-store-800000-apps-and-overtake-apple-appstore/.

16. Castleman, Kenneth R. Digital Image Processing. Upper Sadle River, New Jersey 07458 : Pearson Prentice Hall, 1996. ISBN 0-13-211467-4.

17. Brzoza, Martin. SMART CAR: Detekce dopravních značek. Brno : FIT VUT v Brně, 2009.

18. Vránský, Radovan. Detekce a rozpoznávaní dopravních značek v obraze, bakalářská práce. Brno : FIT VUT v Brně, 2010.

19. Ishizuka, Y. a Hirai, Y. Segmentation of Road Sign Symbols using Opponent-Color Filters. Nagoya : IEEE Conference on Intelligent Transportation Systems, 2004.

20. Španěl, Michal a Beran, Vítězslav. Obrazové segmentační techniky: Přehled existujících metod. [Online] 19. 1 2006. http://www.fit.vutbr.cz/~spanel/segmentace/.

21. Straka, Stanislav. Segmentace obrazu, diplomová práce. Brno : Fakulta informatiky Masarykovy univerzity, 2009.

22. Číp, P. Detekce a rozpoznávání dopravních značek. Brno : FEKT VUT v Brně, 2009.

23. Derpanis, G. Konstantinos. Overview of the RANSAC Algorithm. Toronto : York University, 13. 5. 2010.

24. Chourasia, K. and Chourasia, J. N. A Review and Comparative Analysis of Recent Advancements in Traffic Sign Detection and Recognition Techniques. Uttar-Pradesh : S-JPSET, 2011. Vol. 2. ISSN: 2229-7111.

25. Rojas, Raúl. AdaBoost and the Super Bowl of Classifiers, A Tutorial Introduction to Adaptive Boosting. Computer Science Department, Freie Universität Berlin. [Online] 2009. http://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf.

26. Viola, Paul and Jones, Michael J. Robust Real-Time Face Detection. ICCV. 2001, Vol. 2, p. 747.

27. Baró, Xavier a Vitrià, Jordi. Fast Traffic Sign Detection on greyscale images. [Online] [Citace: 4. 3. 2013.] http://bcnpcl-repo.cvc.uab.cat/21/1/DetSenyals_CCIA04.pdf.

28. Bishop, C. M. Neural Networks for Pattern Recognition. New York : Oxford University Press, 1995.

29. Gao, X. W., Podladchikova, L. a Shaposhnikov, D. Application of Vision Models to Traffic Sign Recognition. Berlin Heidelberg : Springer-Verlag, 2003. stránky 1100-1105.

30. Sochor, Jakub. Rychlá detekce dopravních značek v obraze, bakalářská práce. Brno : FIT VUT v Brně, 2012.

31. Svoboda, Tomáš. Detekce, lokalizace a rozpoznání dopravních značek, diplomová práce. Brno : FIT VUT v Brně, 2011.

32. Ťapuška, Tomáš. Detekce dopravních značek, bakalářská práce. Brno : FIT VUT v Brně, 2008.

33. Leikep, Bořek. Houghova transformace pro detekci čar, bakalářská práce. Brno : FIT VUT v Brně, 2009.

34. Heuer, Tim. Handling picture orientation in CameraCaptureTask in Windows Phone 7. [Online] 23. 7 2010. http://timheuer.com/blog/archive/2010/09/23/working-with-pictures-in-camera-tasks-in-windows-phone-7-orientation-rotation.aspx.

35. Kršek, Přemysl. Základy počítačové grafiky, studijní opora. Brno : Fakulta informačních technologií Vysokého učení technického v Brně, 2006.

36. Dopravní značky. Zákruta.cz. [Online] http://www.zakruta.cz/dopravni-znaceni/. ISSN 1802-8608.

37. Šochman, Jan a Matas, Jiří. AdaBoost. České vysoké učení technické v Praze. [Online] http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf.

Seznam obrázků

Obrázek 1: Domovská obrazovka WP7 a People Hub 6
Obrázek 2: A1a - Zatáčka vpravo 13
Obrázek 3: B1 - Zákaz vjezdu všech vozidel (v obou směrech) 13
Obrázek 4: C1 - Kruhový objezd 14
Obrázek 5: Barevný model RGB 16
Obrázek 6: Barevný model HSV 16
Obrázek 7: Typy hran - a) step, b) ramp, c) roof, d) reálná hrana 19
Obrázek 8: Průběh první a druhé derivace funkce 19
Obrázek 9: Houghova transformace - vstupní obraz a výsledek 21
Obrázek 10: Topologie RBF sítě 25
Obrázek 11: a) FOSTS senzor s fixačními body, b) Detekované hrany ve fixačních bodech 26
Obrázek 12: Prvotní návrh uživatelského rozhraní 31
Obrázek 13: Výsledná podoba uživatelského rozhraní 31
Obrázek 14: Speciální třídy reprezentující objekty 32
Obrázek 15: Výsledek detekce červené barvy v obraze 35
Obrázek 16: Vyříznutý a přebarvený objekt 37
Obrázek 17: Graf výsledků klasifikace v závislosti na vzdálenosti 41
Obrázek 18: Graf výsledků klasifikace v závislosti na odchylce 42
Obrázek 19: Fotografie s použitím blesku 43

Seznam tabulek

Tabulka 1: Hodnoty pro segmentaci červených objektů 34
Tabulka 2: Barevné hodnoty pro přebarvování objektů 37