VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS. MECHATRONICS AND BIOMECHANICS

DETEKCE A ROZPOZNÁNÍ OMEZENÍ RYCHLOSTI Z DOPRAVNÍCH ZNAČEK

DETECTION AND RECOGNITION OF SPEED LIMIT ROAD SIGNS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. VOJTĚCH SOLNICKÝ
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. ROBERT GREPL, PH.D.
SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2014/15

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Vojtěch Solnický

který/která studuje v **magisterském studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Detekce a rozpoznání omezení rychlosti z dopravních značek

v anglickém jazyce:

Detection and recognition of speed limit road signs

Stručná charakteristika problematiky úkolu:

Tato práce se bude zabývat studiem současných metod a algoritmů pro rozpoznání dopravních značek v reálném provozu se zaměřením na značky upravující povolenou rychlost vozidla. Celkovým výstupem práce bude funkční prototyp asistenčního systému, který bude pracovat v reálném čase. Důležitou součástí práce přitom bude provedení experimentů za různých světelných a jiných podmínek a zhodnocení vlastností vyvinutého řešení.

Cíle diplomové práce:

1) Proveďte rešerši v oblasti detekce a rozpoznání dopravních značek se zaměřením na značky omezující rychlost.
2) Nalezněte, implementujte a otestujte několik metod, použijte přitom offline testovací obrazová data. Zhodnoťte metody z hlediska spolehlivosti detekce a náročnosti výpočtu. Uvažujte vliv kvality obrazu a světelných podmínek.
3) Na základě zvolených kritérií vyberte jednu z testovaných metod a tuto implementujte do prototypového hardwaru (kamera a počítač). Metodu odlaďte a otestujte v reálném provozu.

Seznam odborné literatury:

[1] Číp, P.: Detekce a rozpoznání dopravních značek, DP FEKT VUT v Brně, 2009
[2] Image processing toolbox for MATLAB - dokumentace
[3] Computer vision toolbox for MATLAB - dokumentace.

Vedoucí diplomové práce: doc. Ing. Robert Grepl, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2014/15.

V Brně, dne 19. 12. 2014

L.S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan

ABSTRAKT

Diplomová práce se zabývá návrhem a implementací systému pro detekci a rozpoznání omezení rychlosti z dopravních značek. Konkrétně jde o rozpoznání červených kruhových značek *Nejvyšší povolená rychlost* z obrazových dat za použití metod počítačového vidění.

V rámci práce je naprogramováno a otestováno několik metod. Ve finálním řešení je použita segmentace pomocí YCbCr barevného modelu. Detekce kruhové značky i závěrečná klasifikace je vyřešena porovnáním se vzorem. Pro rozpoznání v reálném čase je použit algoritmus sledování detekovaných značek mezi snímky videosekvence. Program je vytvořen v prostředí MATLAB a Simulink.

Výsledkem je prototyp jednoduchého asistenčního systému, který je možné implementovat na jakýkoli počítač s kamerou. Správná funkce algoritmu byla prokázána při testech v reálném provozu.

KLÍČOVÁ SLOVA

počítačové vidění, dopravní značky, omezení rychlosti, real-time, asistenční systém, MATLAB, Simulink, Computer Vision System Toolbox

ABSTRACT

This master‘s thesis describes the design and implementation of the system for detection and recognition of speed limit road signs. It focuses on the recognition of the red circular speed limit sign from the image data using the computer vision methods.

Several methods were programmed and tested as a part of this thesis. In the final solution, the segmentation based on YCbCr color model is used. Detection of the circular sign and final classification is performed by template matching method. Algorithm for the tracking of the detected signs between frames of the video is used for better performance in real-time recognition. Application is developed using MATLAB and Simulink.

The result is a simple driver assistance system prototype, which can be implemented in any computer with camera. The correct function of the algorithm was confirmed during a testing in a traffic.

KEYWORDS

computer vision, road signs, speed limit, real-time, driver assistance system, MATLAB, Simulink, Computer Vision System Toolbox

BIBLIOGRAFICKÁ CITACE

SOLNICKÝ, V. *Detekce a rozpoznání omezení rychlosti z dopravních značek.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 70 s. Vedoucí diplomové práce doc. Ing. Robert Grepl, Ph.D..

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem diplomovou práci na téma „Detekce a rozpoznání omezení rychlosti z dopravních značek“ vypracoval samostatně na základě rad a pokynů vedoucího práce, a že jsem veškeré literární zdroje uvedl v seznamu použité literatury.

V Brně dne 29. 5. 2015

…….....

Vojtěch Solnický

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat doc. Ing. Robertu Greplovi, Ph.D. za vedení mé práce. Dále bych rád poděkoval všem svým přátelům, kteří mne doprovázeli na mé cestě vysokoškolským studiem, bez nichž si tuto cestu nedokážu představit.

A v neposlední řadě patří velké díky celé mé rodině, ale především mým rodičům, za veškerou podporu, kterou mi kdy poskytli.

OBSAH

1. ÚVOD

V posledních letech se v nových automobilech stále častěji uplatňují tzv. pokročilé asistenční systémy (ADAS – Advanced Driver Assistance Systems), jejichž cílem je především zvýšení bezpečnosti na silnicích. Všechny tyto systémy fungují principiálně ve třech krocích. Prvním krokem je získávání informací o okolí a vnitřních stavech vozidla prostřednictvím senzorů. Zadruhé jde o kontinuální vyhodnocování situace na základě pokročilých algoritmů. A třetím krokem je výstup systému, což v závislosti na konkrétní funkci může být zásah do chování vozidla inteligentně řízenými aktuátory, nebo může jít o poskytnutí důležitých informací řidiči. Celkově takový systém představuje příklad synergického propojení několika inženýrských disciplín.

Jako jeden z klíčových senzorů, co se týče analyzování okolí, se v čím dál více případech objevuje videokamera. Člověk získává většinu informací o okolním prostředí prostřednictvím zraku, je tedy přirozené, že se snaží naučit „vidět“ i stroje. Obor počítačového vidění se vyvíjí velmi rychle a vzhledem k neustálému nárůstu výpočetního výkonu standardních procesorů je možné aplikovat stále složitější algoritmy na rozpoznání obrazu.

Díky tomuto bylo možné do automobilů implementovat systémy rozpoznání překážek, systémy udržování vozidla v jízdním pruhu nebo systém rozpoznání dopravních značek.

A právě návrhem systému detekce a rozpoznání omezení rychlosti z dopravních značek se zabývá tato práce. Nejprve je popsáno standardní dělení jednotlivých dílčích problémů rozpoznání značek a nejčastěji používané přístupy k těmto problémům. Následuje krátký přehled funkčních řešení, které svým zaměřením a rozsahem odpovídají této práci.

V části popisu řešení je na základě specifik práce popsáno několik metod, které byly implementovány pro každý krok algoritmu. U každé metody jsou popsány její výhody a nevýhody. Dále je popsáno obsáhlé testování jednotlivých metod, kdy byl kladen důraz na úspěšnost rozpoznání a výpočetní náročnost, ale v potaz byl brán i vliv světelných podmínek a kvalita (rozlišení) analyzovaného obrazu. Následuje popis využití vybraných metod při řešení rozpoznání ve videosekvenci, implementace do prototypového zařízení a zhodnocení výsledků práce.

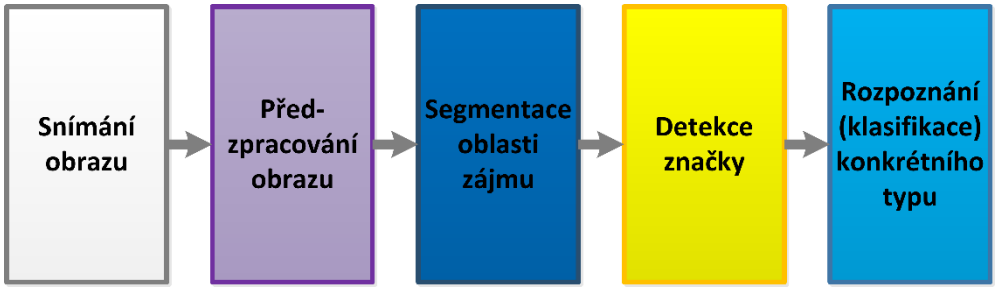
2. REŠERŠE

2.1. Úvod do problematiky rozpoznání značek

Problematikou rozpoznání dopravních značek se lidé začali zabývat ještě dříve, než se v automobilech objevily dostatečně výkonné elektronické systémy, které byly schopny převézt teorii do praxe. Tento systém byl poprvé použit v komerčním řešení v roce 2008 ve vozidle BMW řady 7, kde docházelo k rozpoznání dopravních značek omezení rychlosti. V brzké době se systém rozšířil do většiny vozů nejvyšší třídy a také došlo k rozšíření množiny rozpoznaných značek. V současné době si lze v konfiguraci koupit systém rozpoznání dopravních značek i u vozidel nižší střední třídy, například Škoda Octavia III. generace.

Samozřejmě všichni výrobci komerčních řešení tají specifikace použitých technologií a algoritmů. Naštěstí je toto odvětví velmi populární i na poli akademickém, respektive vědeckém. Za posledních 10 let bylo publikováno nespočet článků na toto téma. Z obecného pohledu lze rozlišit dva odlišné přístupy na základě použití. Prvním z nich jsou algoritmy zaměřené na co největší přesnost rozpoznání při zanedbání výpočetní náročnosti. Toto byl dlouhou dobu hlavní přístup, kdy se čekalo, až bude dostatečně výkonný hardware pro provádění analýzy obrazu v reálném čase. S tímto výkonným hardwarem se začal objevovat i praktický přístup zaměřený na funkční řešení rozpoznání dopravních značek, kdy je real-time použití hlavním cílem a tomu je obětována přesnost rozpoznání, které je následně dosaženo zkombinováním výsledků z několika snímků. Přesto se i v současné době objevují nové a přesnější algoritmy, které jsou však zatím příliš náročné pro použití v reálném čase.

Většina popsaných algoritmů detekce a rozpoznání dopravních značek z obrazových dat sleduje několik standardních kroků. Těmito kroky jsou: snímaní obrazu, předzpracování obrazu, segmentace zajímavých oblastí, detekce dopravní značky a rozpoznání konkrétního typu značky. Jednoduché schéma těchto kroků je na Obr. 2-1.



Obr. 2-1: Obecné schéma algoritmu detekce a rozpoznání dopravních značek

Některé tyto kroky platí i obecně v rámci oboru počítačového vidění, jiné jsou používány až pro specifické problémy, jako je právě detekce a rozpoznání dopravních

značek. V některých případech byly použity i přístupy odlišné [1], ale jinak se většina řešení liší jen v jednotlivých metodách, případně dochází ke sloučení dvou kroků do jednoho (např. detekce a rozpoznání u [2]). V následujících kapitolách jsou popsány nejpoužívanější metody v jednotlivých dílčích krocích.

2.1.1. Předzpracování obrazu

Předzpracování obrazu je standardní krok u většiny problémů v oboru počítačového vidění. Ve fázi předzpracování často dochází k úpravám obrazu, které mají překonat nedokonalosti snímacího zařízení, například odstranění šumu, úprava barev a podobně a přiblížit obraz přirozenému lidskému vjemu. Dále je cílem předzpracování provézt takové změny obrazu, které zvýší účinnost, nebo jsou přímo podmínkou následujících algoritmů. Mezi takové změny patří například zvýraznění kontrastu hran.

Tyto úpravy se provádí prostřednictvím různých typů filtrů. Problematika filtrace obrazu je velmi široký obor a pro konkrétní problém mohou být použity různé filtry. Přehled o základních typech lze nalézt v [3].

Ve studovaných pracích byl nejčastěji využit jednoduchý Gaussovský filtr pro odstranění šumu. Někdy bývá také použit náročnější mediánový filtr, který odstraňuje šum a přitom zachovává, případně i zvýrazňuje ostré hrany.

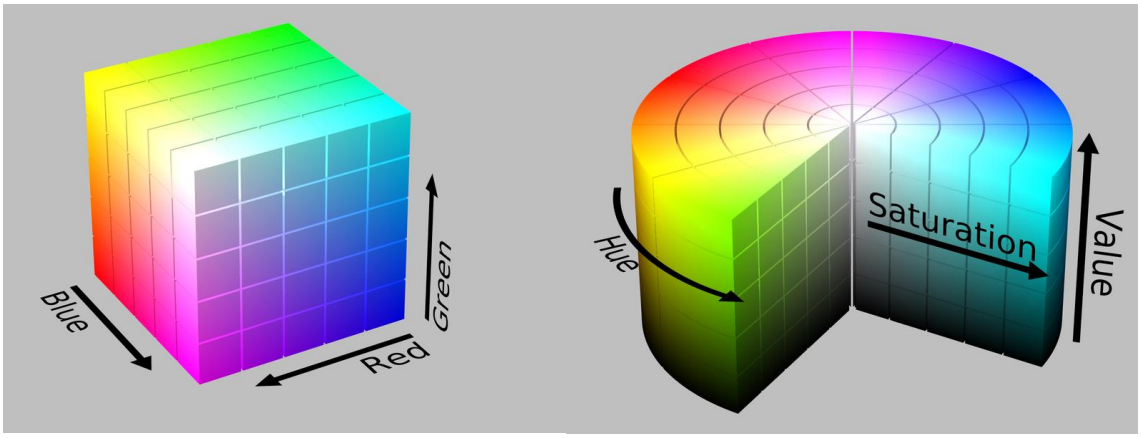
Vzhledem k tomu, že filtrace probíhají na celém obrazu a s různými velikostmi konvolučních jader, jde o velmi náročné operace. Proto u real-time problémů bývá snaha tyto operace co nejvíce omezit, případně zcela vynechat.

2.1.2. Segmentace

Dalším krokem při většině algoritmů počítačového vidění je segmentace, neboli rozdělení obrazu na oblasti zájmu a ostatní oblasti (například objekt – pozadí). Obecně se přístupy nejčastěji dělí na segmentaci na základě barevné informace a segmentaci na základě tvaru. V případě detekce dopravních značek jsou možné oba přístupy a oba jsou zjednodušeny tím, že u značek jde o standardizovanou barvu i tvar.

Segmentace na základě barev je častější přístup. Samozřejmě nejde o hledání jedné konkrétní standardizované barvy. Na výslednou barvu, jak se jeví na sejmutém obraze, má totiž vliv spousta faktorů. Nejvíce záleží na světelných podmínkách při získávání snímku a dále na reprezentaci barev zobrazovacího zařízení. Aby byly tyto vlivy potlačeny a výsledný algoritmus byl co nejuniverzálnější, využívají se pro reprezentaci barev různé barevné modely a segmentace probíhá na základě intervalu hledaných barev. Mezi často používané barevné modely patří standardní RGB, YCbCr, HSV, CIELab. V [4] lze nalézt podrobný popis modelů RGB, HSV a YCbCr, metod segmentace na základě těchto modelů a porovnání výsledků při detekci značek. Práce [5] je pak přímo zaměřena na porovnání jednotlivých barevných modelů při detekci značek

různých barev. V obou pracích je jako nejpřesnější metoda segmentace vyhodnocena metoda založena na HSV modelu. Tento model je blízký lidskému vnímání barev a jeho úspěch je tedy logický. Nevýhodou však je větší výpočetní náročnost při převodu obrázku do HSV barevného prostoru.



Obr. 2-2: Porovnání RGB a HSV modelu, převzato z [6]

Při segmentaci na základě tvaru dochází nejdříve k výpočtu hran objektu v obraze. K tomuto slouží různé algoritmy, například Cannyho hranový detektor [7] nebo Houghova transformace [8]. Na základě vypočtených hran jsou pak vyhledány oblasti, které odpovídají tvaru dopravní značky. Při tom je nutné myslet na perspektivní deformace značek, kdy například kruh se deformuje na elipsu a čtverec či obdélník na lichoběžník.

Vzhledem k výhodám a nevýhodám obou základních přístupů dochází často k jejich kombinaci, kdy je značka segmentována na základě barvy a konkrétní ohraničení je uskutečněno prostřednictvím detekce hran. Tímto způsobem vzniká ideálně segmentovaný obraz pro další krok, kterým je detekce.

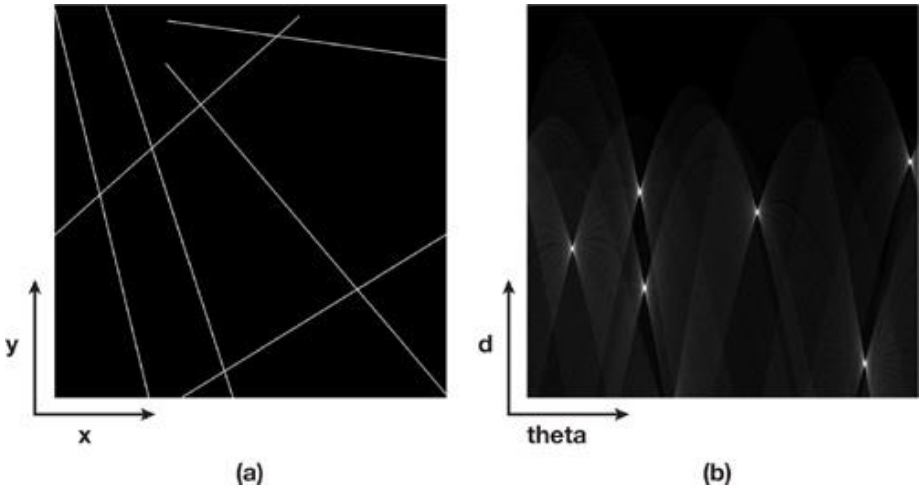
2.1.3. Detekce

Detekcí v této práci rozumíme proces nalezení konkrétní polohy dopravní značky v segmentovaném obraze a případně určení o jaký typ značky jde (kruhová, trojúhelníková atd.).

V případě, že je použito zvýraznění hran, jak bylo zmíněno výše, je standardním přístupem analýza těchto hran, na jejímž základě se rozhodne o detekovaném tvaru. K nejčastěji používaným metodám v tomto případě patří Houghova transformace. Tato metoda je založena na hledání parametrického vyjádření hledaného tvaru prostřednictvím výpočtu všech možných v parametrů pro pixely, jež do tvaru patří.

Konkrétně v nejjednodušším případě hledání přímky, jde o popis přímky dvěma parametry (vzdálenost od středu a úhel svírající s osou x). Pokud vykreslíme všechny

přímky procházející daným bodem obrazu v parametrické rovině (osy roviny jsou hledané parametry) dostaneme sinusoidu. Pokud je tento postup zopakován pro další body na hledané přímce, výsledné sinusoidy se protnou v bodě, který odpovídá parametrům přímky, na níž body leží. Stejný postup je funkční i u složitějších tvarů (kružnice, elipsa), zvyšuje se jen dimenze celého problému v závislosti na počtu hledaných parametrů. Více o Houghově transformaci lze nalézt v [9].



Obr. 2-3: Ukázka hledání přímek v parametrické rovině, převzato z [10]

Celkově jde o nejpoužívanější přístup detekcí hran v oboru počítačového vidění a to díky své relativní jednoduchosti a přesnosti (objekty jsou detekovány i v případě, že nejsou zcela kompletní). S velikostí obrazu a složitostí hledaného tvaru však dochází i k nárůstu výpočetní náročnosti.

Jiným častým přístupem je porovnání se vzorem. Neboli segmentovaná oblast je prostřednictvím vybrané váhové funkce porovnána se vzory představujícími různé značky a na základě největší podobnosti je prohlášena za konkrétní typ. Případně mohou být porovnány i další vlastnosti vzoru a obrazu (velikost, natočení atd.) – čímž dojde k získání dalších cenných informací. Tato metoda je přes svou jednoduchost velmi účinná a výpočetně ne příliš náročná a je tedy častým řešením především v real-time aplikacích.

2.1.4. Rozpoznání

Rozpoznáním, nebo klasifikací je míněno konkrétní určení detekované značky. K tomu je využito vnitřních piktogramů značky. V některých případech se tento krok provádí společně s detekcí, v jiných je na základě detekce jen omezena množina možných výsledků (u kruhových značek se uvažují jen příslušné piktogramy). Často je před provedením samotné klasifikace piktogram značky znova segmentován, případně jsou určeny další statistické vlastnosti.

Stejně jako u detekce se v tomto kroku hojně využívá porovnání se vzorem. Vlivem větší množiny vzorů, jejich složitosti a u některých i vzájemné podobnosti, je

často potřeba vytvořit složitější algoritmus porovnání, který tyto problémy řeší například použitím složitějších rozhodovacích stromů, nebo přidáním dalších měřítek do porovnání.

Podobným přístupem je porovnání na základě statistických údajů. Nejde o přímé porovnání obrazů, ale jejich vlastností, které jsou jak pro vzory tak následně pro obraz vypočteny. Jako tyto údaje se například používají poměr hlavní a vedlejší poloosy, kompaktnost, konvexnost, momentové invarianty a další. Nejčastější je použití více těchto údajů, které jsou následně vhodně zkombinovány (např. metodou AdaBoost [11]) pro výslednou klasifikaci. Výhodou tohoto přístupu je nezávislost na velikosti a natočení obrazu při použití normalizovaných veličin.

Mezi další tradiční přístupy rozpoznání patří umělé neuronové sítě. Tyto bývají natrénovány na velké množině testovacích značek (řádově stovky instancí jednoho typu značky) a následně využity pro rozpoznání. Úspěšnost v tomto případě závisí na konkrétním nastavení neuronové sítě a na univerzálnosti trénovací množiny. Celkově však jde o docela složitý postup s ne zcela jistým výsledkem [4].

Mezi dalšími algoritmy pak stojí za zmínku algoritmus FOSTS (Foveal System for Traffic Signs) [12]. Tento algoritmus má za cíl napodobit vnímání známých tvarů lidským zrakem. Funguje na principu určování orientace hran v okolí 27 přesně definovaných bodů na třech soustředných kružnicích. Autoři článku hodnotí metodu jako velmi přesnou (95 % rozpoznaných značek) a to i za různých světelných podmínek a také jako velmi rychlou.

Samozřejmě existují i další méně používané přístupy, nejen v klasifikaci, ale v celkovém řešení problému detekce a rozpoznání dopravních značek, ale popsat všechny by bylo nad rámec této práce.

2.2. Rešerše konkrétních řešení

Jak již bylo zmíněno výše, o této problematice bylo v posledních letech napsáno mnoho publikací. Ze všech studovaných zdrojů proto byly vybrány 3 práce, které jsou zaměřením a použitými metodami nejblíže konkrétnímu problému, který vyplývá ze zadání této práce. Ve všech třech případech jde o závěrečné práce studentů různých fakult Vysokého učení technického v Brně.

2.2.1. Detekce a rozpoznání dopravních značek – Pavel Číp

Pavel Číp ve své práci [4] řeší obecný problém detekce a rozpoznání dopravních značek. Na začátku předkládá velmi obsáhlý popis používaných metod v jednotlivých krocích algoritmu. Velká část ze zkoumaných metod je také v jeho práci implementována a zhodnocena a na základě stanovených cílů vybráno nejlepší řešení. Jako prostředí pro implementaci metod byl vybrán MATLAB a použit byl především Image Processing Toolbox.

Předzpracování obrazu je vynecháno. Z metod segmentace jsou popsány a implementovány metody na základě barevných modelů RGB, HSV a YCbCr. Nejlepší výsledky získává při HSV segmentaci, pro RGB model byly problematické žluté značky a YCbCr je dle jeho práce příliš závislý na snímacím zařízení.

Pro detekci bylo vyzkoušeno opět několik metod. Metoda radiometrických deskriptorů je hodnocena jako velmi přesná, ale také výpočetně náročná vlivem výpočtu statistických údajů pro segmentované tvary. Genetický algoritmus na vyhledávání kruhových značek se ukázal jako velmi výpočetně náročný a navíc ne příliš přesný. Aplikace detekce prostřednictvím Houghovy transformace byla vyhodnocena jako příliš výpočetně náročná a tento přístup byl použit jen v rámci normalizace tvarů. Jako nejpřesnější a zároveň nejrychlejší se ukázalo porovnání se vzorem, které bylo aplikováno ve finálním algoritmu.

Před rozpoznáním je popsán algoritmus normalizace velikosti založený na hledání vrcholů značek prostřednictvím Houghovy transformace a následné perspektivní transformaci.

Pro samotné rozpoznání je nejprve použita umělá neuronová síť, která nepřináší ideální výsledky. Další přístup je počítání momentových invariantů, které jsou pro klasifikaci využity opět v rámci neuronové sítě a také na základě rozhodovacího stromu. Tyto metody fungovaly dobře na vzorových datech, ale při použití reálných testovacích snímku docházelo k nepřesnostem. Jako nejlepší řešení se ukázalo opět porovnání se vzorem, které je provedeno na základě kvadratické odchylky od obrazu ve stupních šedi.

Výsledkem je prototypový program na detekci a rozpoznání dopravních značek v provozu. Pracuje s obrazy s rozlišení 640x480 pixelů. Dle autora je úspěšnost rozpoznání velmi vysoká. Nedosahuje však příliš dobrého času výpočtu pro použití v real-time aplikaci, neboť průměrný čas se pohybuje kolem 1,2 vteřiny.

Obr. 2-4: Ukázky výsledného řešení Pavla Čípa, převzato z [4]

2.2.2. Využití metod zpracování signálů pro zvýšení bezpečnosti automobilové dopravy – Radek Beneš

V práci Radka Beneše [13] je opět cílem detekovat a rozpoznat všechny používané dopravní značky. K tomu je přidán požadavek na vytvoření prototypového programu, který bude opravdu fungovat v provozu a tedy celý algoritmus provádět s dostatečnou rychlostí. To vede na časté upřednostnění rychlosti na úkor přesnosti. Celý program je vytvořen prostřednictvím jazyka C++ a knihovny OpenCV, která obsahuje funkce pro počítačové vidění.

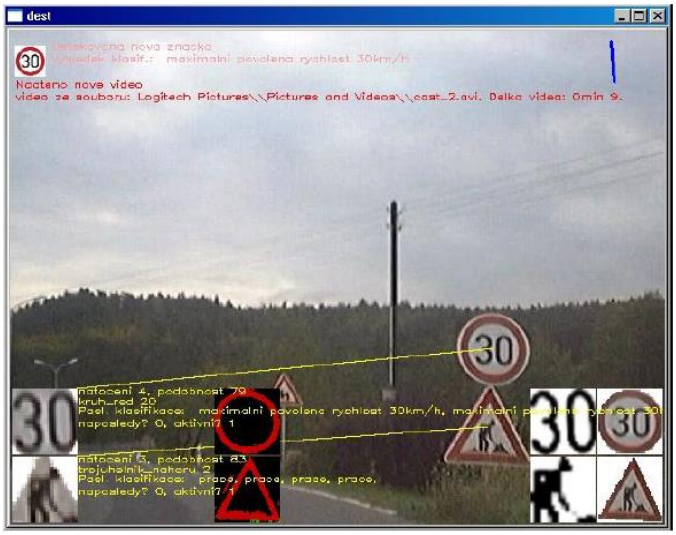
Po filtraci jednoduchým Gaussovým filtrem je obraz segmentován. Stejně jako v předchozí práci je vyzkoušena segmentace na základě modelů RGB a HSV. I v této práci je jako přesnější hodnocena metoda prostřednictvím HSV modelu, ale vlivem nižší náročnosti je vybrána RGB segmentace. Po segmentaci jsou ještě prostřednictvím Cannyho hranového detektoru vypočteny hrany objektu a zpřesněno ohraničení segmentovaných oblastí.

Detekce značky je řešena porovnáním se vzorem. Od každého typu značky jsou použity vzory pro natočení až ± 12 stupňů, což je využito i při normalizaci značek. Dále je už ve fázi detekce využito sledování značek v rámci videosekvence (viz dále), kdy například zmíněné natočení je zkoušeno jen v omezené míře na základě natočení v minulém snímku.

Rozpoznání vnitřního piktogramu značky je také provedeno porovnáním se vzorem. Kromě samotného obrazu piktogramu je v příznakovém vektoru i několik dalších informací, například o poloze piktogramu ve značce a velikosti. Pro každý piktogram je také vygenerováno několik vzorů, které mají omezit vliv nepřesného snímku, případně nepřesné segmentace.

Autor použil algoritmus sledování a predikce značek mezi snímky videosekvence. Tento algoritmus je založen na Kalmanově filtru a je využit k celkovému zvýšení úspěšnosti (rozhodování o rozpoznání značky na základě několika snímků) a snížení náročnosti (predikce polohy a typu značky v následujícím snímku).

Výsledný program má úspěšnost rozpoznání v testovacím videozáznamu 78 % a rychlost provádění algoritmu se pohybuje mezi 40 a 90 milisekundami v závislosti na složitosti scény, což je pro aplikaci rozpoznání v reálném provozu dostačující. Cíle práce tedy byly splněny.



Obr. 2-5: Ukázka výsledného programu Radka Beneše, převzato z [13]

2.2.3. Aplikace pro rozpoznání dopravních značek pro Windows Phone 7 – Marek Dvořák

Marek Dvořák [2] se zabývá aplikací pro mobilní platformu, která slouží k rozpoznání zákazových (tedy kruhových červených) dopravních značek. Cílem nebylo vytvořit opravdový asistenční systém, který by měl fungovat v reálném čase, ale pomůcku na rozpoznání vyfocených dopravních značek. Práce je zajímavá ze dvou důvodů. Zaprvé zde velmi záleží na výpočetní náročnosti, ale tentokrát z důvodu omezené výpočetní kapacity přenosného zařízení. A za druhé je zde algoritmus vyvíjen bez podpory knihoven počítačového vidění.

Na začátku je obsáhlá rešerše používaných metod. Na základě rešerše je vybrán pro segmentaci červené barvy HSV model. Po segmentaci červených objektů (značek) dochází k normalizaci na 100x100 pixelů. Následně je celá vnitřní část značky opět segmentována na základní barvy: černá, bílá, žlutá, zelená, modrá. K tomu jsou uvedeny meze pro jednotlivé barvy v HSV modelu.

Takto zcela segmentovaná značka je pixel po pixelu porovnána se stejně segmentovanými vzory a na základě největší shody rozpoznána.

Autor práce stanovil při testování úspěšnost na 82 %. Při práci s fotografiemi s rozlišením 1600x1200 pixelů pak rozpoznání zabere 1,5 sekundy.

Obr. 2-6: Ukázka výsledku práce Marka Dvořáka, převzato z [2]

2.3. Podpora počítačového vidění v programu MATLAB

2.3.1. Computer Vision System Toolbox

Pro podporu vývoje algoritmů počítačového vidění slouží v programu MATLAB Computer Vision System Toolbox. Tento toolbox obsahuje funkce a nástroje nejen pro většinu základních metod počítačového vidění, ale pokrývá také oblast složitějších algoritmů, například detekci obličeje nebo hledání a detekování SURF features. Více o tomto toolboxu lze nalézt na [14]

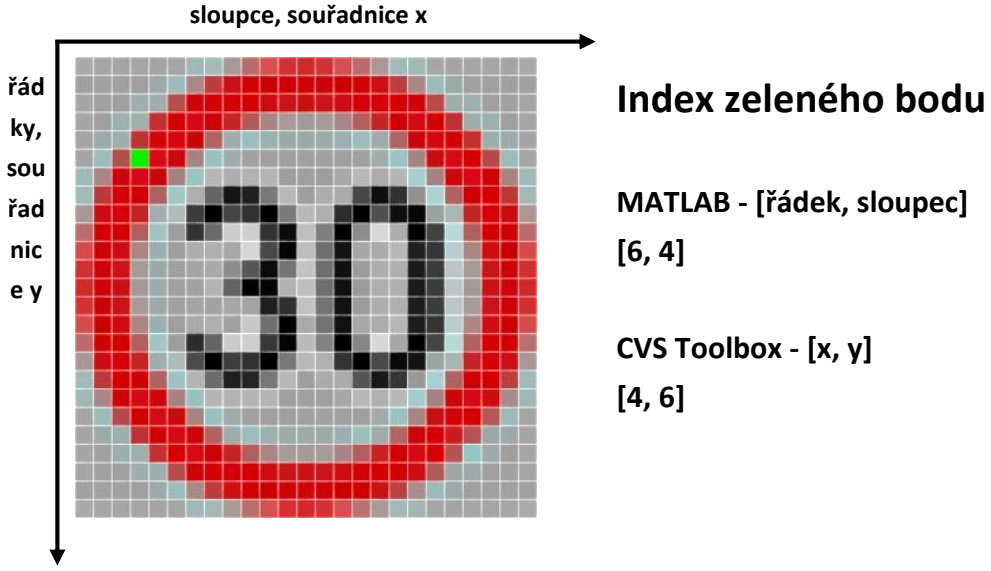
Většina složitějších algoritmů je v rámci CVS Toolboxu implementována jako tzv. *system object*. Tato třída funguje podobně jako objekty v objektově orientovaných jazycích. Má své atributy, neboli nastavení a metody, kterými provádí různé operace. Práce s třídou *system object* funguje následovně. Na začátek je potřeba objekt inicializovat na objekt dané třídy (například objekt perspektivní transformace) a nastavit mu příslušné atributy (velikost cílového obrazu). Pokud objekt se stejnými atributy voláme opakovaně (například v rámci opakující se funkce pro každý snímaný obraz), je dobré jej nastavit jako tzv. *persistent* proměnnou. Takto se objekt zachovává i mezi jednotlivými běhy funkce a náročná inicializace se prostřednictvím podmínky provede jen při prvním volání funkce. Samotné provedení operace je pak zajištěno voláním metody step daného objektu. Další důležitou metodou je metoda *release* pro změnu atributů.

System object odpovídá v prostředí Simulink jednotlivým blokům a atributy jsou nastavovány standardně prostřednictvím dialogových oken. Práce v tomto prostředí je pro uživatele intuitivnější.

2.3.2. Práce s obrazovými daty v prostředí MATLAB

Obrazová data jsou v prostředí MATLAB vyjádřena jako matice hodnot jednotlivých pixelů. V případě, že máme obrázek 640x480 pixelů v odstínech šedi vyjádřený v MATLABu, bude z něj matice o 480 řádcích a 640 sloupcích. V případě, že jde o obrázek barevný, je takto vyjádřen každý z kanálů RGB, výsledkem je pole tří takových matic (480x640x3). Vzhledem k zaměření programu MATLAB na práci s maticemi je toto vyjádření ideální a s obrazy jde pracovat jako s jakýmikoli jinými maticemi, včetně indexování a využití výhod vektorizace.

Standardním datovým typem je double (64 bitů), který v obrazových datech nabývá hodnot 0 až 1. Každý jednotlivý pixel je vyjádřen tímto typem. Vzhledem k ulehčení paměti je vhodné obraz vyjádřit datovým typem uint8, případně uint16, kdy je intenzita pixelu vyjádřena hodnotami 0 až maximální hodnota datového typu (255, respektive 65535).



Obr. 2-7: Ukázka rozdílu indexování obrazových dat

Při práci s obrazovými daty prostřednictvím standardních MATLAB funkcí a zároveň požitím funkcí CVS Toolboxu je potřeba mít na paměti rozdíl v indexování jednotlivých pixelů a tím pádem i oblastí obrazu. Zatímco v MATLABu je standardem indexování ve formě [řádek, sloupec], při práci s CVS Toolboxem se nejčastěji indexuje ve formě souřadnic obrazu [x, y], kde osa x je vodorovná a odpovídá šířce obrazu a y je svislá a odpovídá výšce.

Jde o opačný přístup než u standardních matic, což může při nepozornosti vést k těžko identifikovatelným problémům. Více o indexování obrázků lze nalézt v [15]. Pro přehlednost je problematika shrnuta na Obr. 2-7.

3. ŘEŠENÍ A VÝSLEDKY

3.1. Specifika práce

Z provedené rešerše vyplynulo několik směrů, kterými by bylo možné se při implementaci řešení detekce a rozpoznání omezení rychlosti z dopravních značek vydat. Nejdříve bylo potřeba si stanovit hlavní specifika práce oproti výše zmíněným řešením. Hlavním rozdílem jsou následující dva body:

- Zaměření pouze na značky omezující rychlost.
- Požadavek na rychlost algoritmu – možnost využití v provozu.

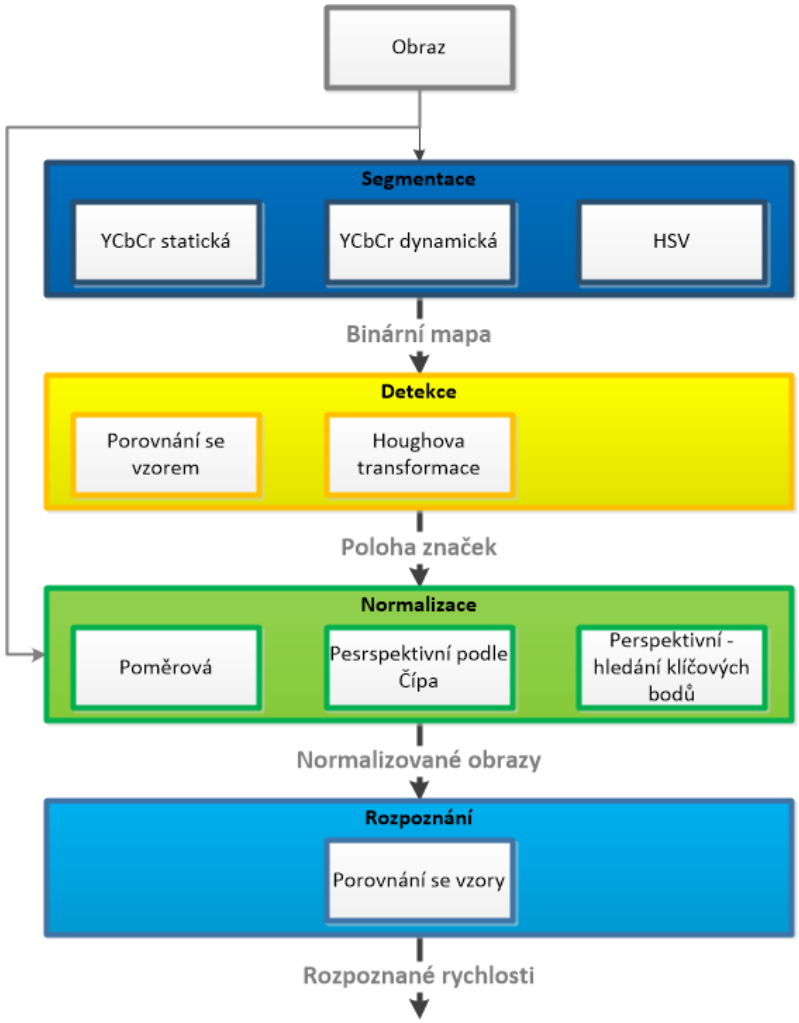
Tyto body se navzájem doplňují. Díky omezení množiny, kterou je potřeba rozpoznávat je možné sestavit štíhlejší algoritmus, který bude méně výpočetně náročný. Přesto bylo potřeba mít na paměti výpočetní náročnost po celou dobu vývoje algoritmu. Z tohoto důvodu také nebyly zvažovány metody detekce a rozpoznání, které měly sice skvělé výsledky z pohledu úspěšnosti, ale výpočetní náročnost neodpovídala potřebám real-time aplikace, alespoň tedy při použití standardního hardwaru.

3.2. Implementace vybraných řešení

Z výše uvedeného vyplývá, že implementace byla zaměřena především na méně komplexní algoritmy. Jelikož se práce zabývá jen značkami omezení rychlosti, nedalo se na základě rešerše zcela určit, které přístupy jsou pro tento účel nejvhodnější, protože práce většinou obsahovaly zhodnocení algoritmů jako celku pro celou množinu značek. Bylo tedy rozhodnuto o implementaci několika metod, a následném provedení testů, na základě kterých bude nejvhodnější řešení vybráno.

Celkový algoritmus byl rozdělen do částí, které jsou popsány v kapitole 2.1. Pro každý z bloků (segmentace, detekce, normalizace, rozpoznání) pak bylo implementováno několik metod. Při implementaci dílčích metod byl kladen důraz na modularitu těchto řešení. Tedy aby každý z těchto „modulů“ bylo možné použít v jakékoli kombinaci s ostatními moduly. Toho bylo dosaženo standardizováním vstupů a výstupů mezi jednotlivými bloky. Například mezi blokem segmentace a detekce je vždy předávána mapa logických hodnot odpovídající segmentovanému obrazu. Jednotlivé bloky, moduly a vstupy/výstupy je možné vidět na Obr. 3-1.

Co se týče výpočetní náročnosti, jako kritické se ukázaly operace, které se týkají celého obrazu, jako jsou například filtrace, prahování nebo samotné vykreslení obrázku. Např. při rozlišení 800x600 pixelů znamená každá operace na celém snímku 480 000 operací na jednotlivých pixelech, případně třikrát více pokud uvažujeme 3 kanály (RGB). Z tohoto důvodu byla při vývoji algoritmu snaha se těmto operacím na celém obrazu co nejvíce vyhnout, a proto také nebylo použito předzpracování obrazu.



Obr. 3-1: Schéma použitého modulárního řešení

Jako prostředí pro vytvoření jak jednotlivých algoritmů, tak celkového řešení programu bylo zvoleno prostředí MATLAB a jeho nástroj Simulink. Pro potřeby řešení konkrétního problému rozpoznání dopravních značek bylo potřeba, aby v MATLABu byly instalovány následující toolboxy:

- Computer Vision Systém Toolbox
- Image Acquisition Toolbox
- Image Processing Toolbox

Toto vývojové prostředí bylo zvoleno především kvůli uvedeným toolboxům, které obsahují funkce pro práci s obrazovými daty. Ať už jde o jednoduché funkce na ořez nebo transformaci obrazu, po složitější algoritmy jako jsou filtrace, nebo ekvalizace histogramu. Funkce, objekty a bloky v rámci Computer Vision System Toolboxu pak pokrývají většinu základních a spoustu pokročilých algoritmů využívaných v oboru počítačového vidění. Toto vše, ve spojení s jednoduchým vytvářením dataflow a

intuitivní prací s videem v Simulinku, vede k jednoduššímu a rychlejšímu vývoji celé aplikace, než při použití standardních jazyků jako jsou C nebo C++.

Z této jednoduchosti však plynou i nevýhody práce v MATLABu. Jde především o nemožnost vidět „dovnitř“ funkcí a určování nastavení namísto uživatele (např. určení datových typů). Vývojář tak nemá úplnou kontrolu nad vytvořeným algoritmem, nebo ji získává až dalším explicitním nastavením. To se v případě této práce dotýká především výsledné výpočetní náročnosti, kdy funkce, která by stačila v jednoduché podobě, v MATLABu obsahuje další „skryté“ příkazy, kvůli možnostem univerzálního použití a celkové robustnosti.

Při vytváření algoritmu byla snaha tyto nevýhody eliminovat explicitním nastavením a v některých případech byla místo použití vytvořené funkce napsaná vlastní funkce, která stejný problém řešila jednodušeji a „na míru“. Příkladem takové funkce je napsaná funkce na výřez části obrazu *imorez*.

3.2.1. Algoritmy segmentace obrazu

3.2.1.1. YCbCr segmentace se statickým prahem

Prvním z algoritmů na segmentaci obrazu, který byl naprogramován, byla segmentace založená na YCbCr barevném modelu. Výhodou tohoto modelu pro použití v detekci červených značek je fakt, že obsahuje Cr složku neboli červený chrominanční komponent. Při vykreslení této složky jsou jasně patrné červené oblasti v obrazu, včetně značek omezujících rychlost. Jako ideální přístup k segmentaci červených značek byla označena metoda založená na prahování jednotlivých složek YCbCr modelu i v [5].

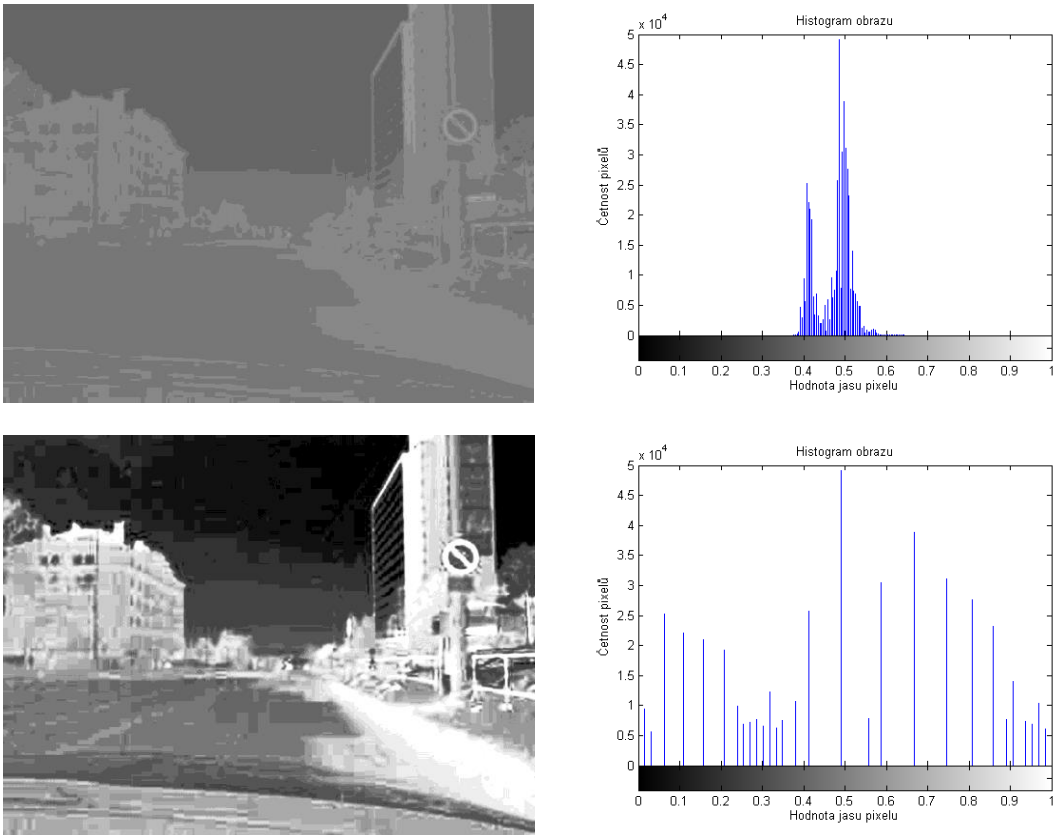
Samotná segmentace je založená na jednoduchém principu statického prahování. Neboli vše s menší hodnotou jasu než daný práh je označeno hodnotou 0 a vše větší nebo rovno prahu je označeno 1. Jako práh byla experimentálně zvolena hodnota 0,56, pokud hodnoty intenzity měříme v rozsahu 0 až 1. Při rozsahu 0-255 (pokud je obrázek vyjádřen datovým typem uint8) je hodnotou prahu hodnota 143. Příklad prahování lze vidět na Obr. 3-2.

Obr. 3-2: YCbCr prahování se statickým prahem 0,56. Zleva: RGB, Cr složka a binární mapa

Z obrázku je patrné, že dochází k dobré segmentaci červené barvy. Samozřejmě se často v obrázcích vyskytují i další části s odstíny červené barvy. Tyto oblasti by bylo možné eliminovat zavedením prahu i na další složky modelu a výslednou binární mapu vytvořit binárním součinem vzniklých map. Jelikož by toto řešení velmi zvyšovalo výpočetní náročnost, další oblasti zatím tolerujeme a eliminujeme je v algoritmu detekce.

Jak je vidět, výhodou metody je přesná segmentace a dále jelikož prahujeme jen jednu složku jen jedním statickým prahem, jde o nejméně výpočetně náročný způsob. Jako nevýhoda byly zjištěny problémy s prahováním při horších světelných podmínkách, kdy má Cr složka u značek nižší hodnoty.

Jako způsob, jak překonat tuto nevýhodu bylo navrženo použití ekvalizace histogramu na Cr složku. Od metody se očekávalo, že posune nižší hodnoty Cr složky, které odpovídají červeným značkám k vyšším hodnotám jasu a bude možné je prahovat stejným prahem jako fotografie s dobrým osvětlením (i když pravděpodobně jiným, než výše zvoleným). Bohužel se metoda neosvědčila, protože hodnoty odpovídající značkám a jím blízké příliš saturuje až na hraniční mez (především u značek s dobrými světelnými podmínkami) a není možné značky dobře segmentovat.



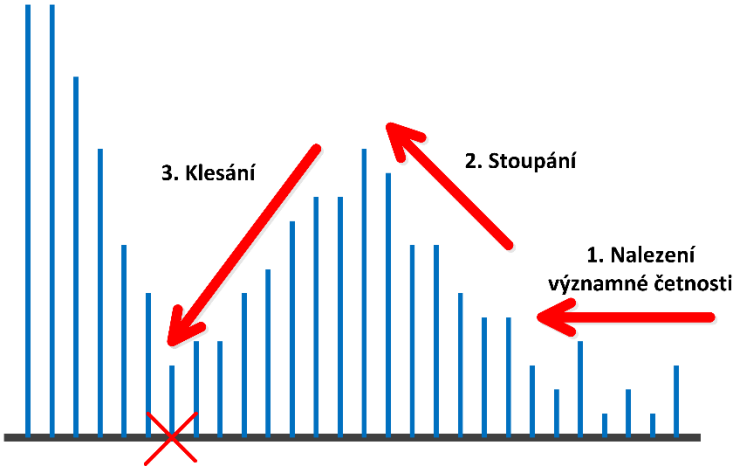
Obr. 3-3: Ukázka ekvalizace histogramu Cr složky obrazu

3.2.1.2. YCbCr segmentace s dynamickým prahem

Na základě neúspěchu s ekvalizací histogramu byl navržen další algoritmus, který je založený na histogramu, ale jde na to z druhé strany. Tento koncept vznikl na základě úvahy, že lepší, než upravovat celý obraz, abychom mohli použít stejný práh, bude lepší upravit práh a tedy dynamicky jej přizpůsobit danému obrazu.

Nejdříve bylo potřeba přijít na pravidlo, na základě kterého se dynamický práh vypočte. Byla sestavena množina obrázků dopravních značek, prozkoumány jejich histogramy a vyzkoušeny některé triviální metody (např. vybrat práh, který jako první zprava dosahuje vybrané četnosti). Jelikož triviální pravidla nedosáhla větší úspěšnosti, bylo na každém obrázku provedeno iterativní vyhledávání nejlepších mezí tak, aby následný algoritmus detekce (založeny na hledání kruhu Houghovou transformací, viz 3.2.2.3) správně detekoval značku. Po prozkoumání histogramů s nalezenými prahy bylo navrženo řešení založené na hledání významného lokálního minima zprava, neboť četnosti pixelů červených oblastí v histogramech vytvářely větší či menší vrcholy.

Samotný algoritmus pracuje ve 3 krocích. Nejdříve nalezne první významnější četnost zprava (aby byl omezen vliv šumu a ojedinělých pixelů), dále sleduje stoupající četnosti až po vrchol a následně klesá až do lokálního minima. Algoritmus je znázorněn na Obr. 3-4.

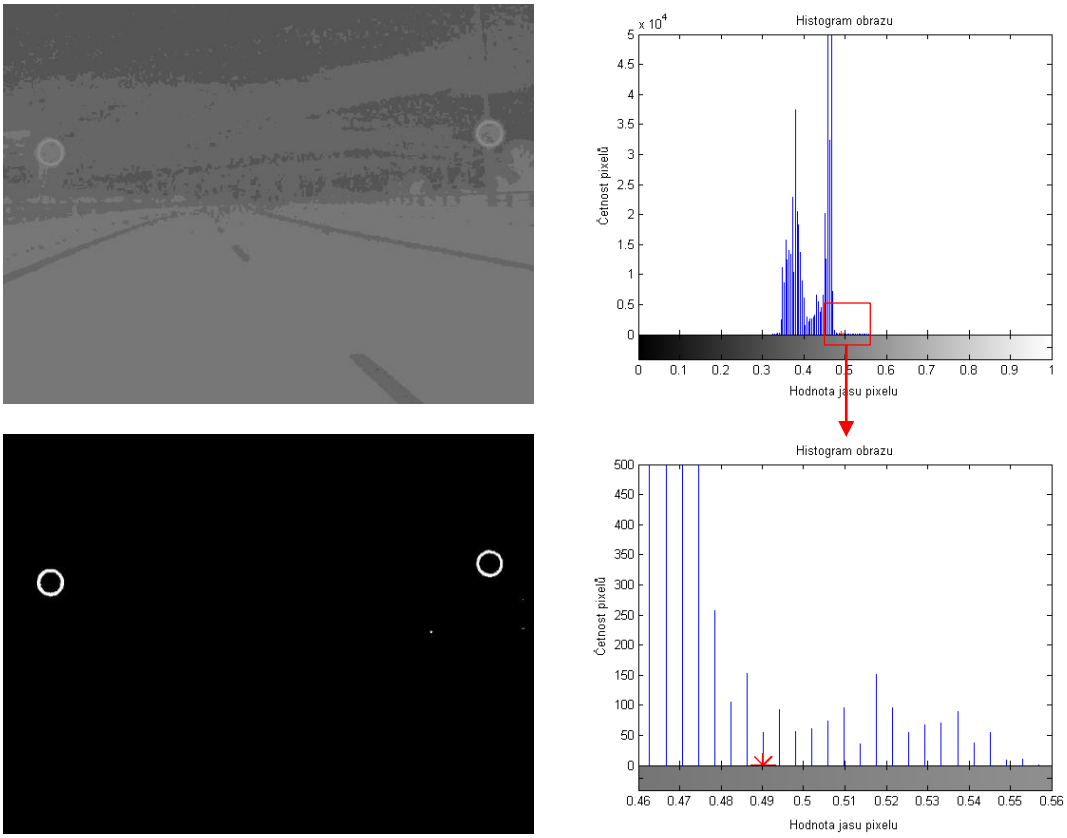


Obr. 3-4: Znázornění funkce hledání dynamického prahu

Základní verze prohledávání byla málo robustní a často se zasekla v bezvýznamném bodě, kde byla náhodně menší četnost, i když ve větším měřítku šlo stále o stoupání. Pro vyrovnání těchto „nerovností“ byl vyzkoušen medián filtr, od kterého se očekávalo „vyhlazení histogramu“. Ukázalo se však, že tento typ filtru příliš úspěšnost nezvýší.

Jako další úprava bylo implementováno prohledávání o 2 kroky vpřed, kdy jak při stoupání, tak při klesání algoritmus kontroluje dva následující kroky a pokud má alespoň

jeden z nich stoupající/klesající tendenci, pokračuje dále. Toto opatření potlačilo vliv ojedinělých děr a vrcholů a zlepšilo úspěšnost algoritmu. Aby byl ještě více potlačen vliv malých lokálních minim a maxim, byla zavedena podmínka minimálního rozdílu mezi jednotlivými četnostmi, které algoritmus bere při stoupání a klesání v potaz. Poslední problém, který se objevoval, a bylo jej potřeba vyřešit, byla neexistence významných lokálních minim u obrázků, kde bylo více červených ploch, kdy se algoritmus zastavoval např. až po překonání nejvyššího vrcholu celého histogramu. Tento byl vyřešen podmínkou zastavení algoritmu, v případě že dochází k přílišnému růstu četnosti. Příklad výsledku finálního algoritmu lze vidět na Obr. 3-5.



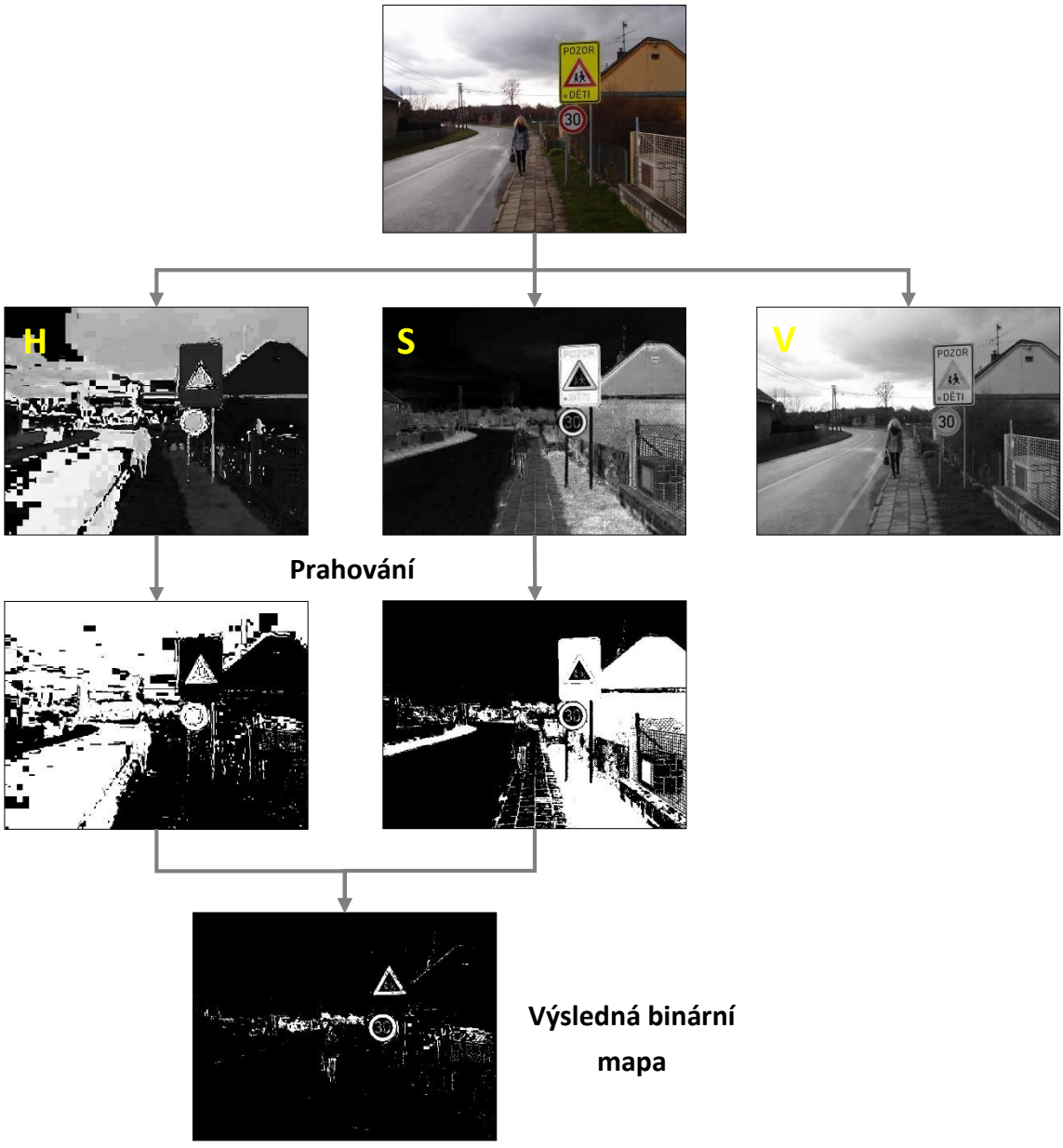
Obr. 3-5: Ukázka obrázku segmentovaného na základě dynamického prahu. Nalezený práh je v histogramu vyznačen červeně.

Výsledný algoritmus dosahoval velmi dobrých výsledků na testovací množině, která byla složena především ze snímků značek při horších světelných podmínkách. V tomto případě je dynamický algoritmus úspěšnější, než algoritmus se statickým prahem. Nevýhodou je lehce zvýšená výpočetní náročnost vlivem výpočtu histogramu a jeho následného prohledávání.

3.2.1.3. HSV segmentace

Posledním algoritmem segmentace, který byl implementován, je segmentace založená na HSV barevném modelu. Tento model je často používán, například v [16], protože se díky němu dají dobře segmentovat i další barvy dopravních značek a to nezávisle na světelných podmínkách.

V základní verzi algoritmu byl obrázek prahován ve všech 3 složkách a výsledný obraz vznikl binárním součinem. Základní meze byly přejaté z [2]. Následně došlo k ručnímu ladění mezí. Jelikož vliv prahování Value složky byl velmi malý, byla tato složka z důvodu sníženi náročnosti zanedbána. Výsledné meze pro Hue a Saturation složku jsou v Tab. 3-1. Ukázka segmentace HSV modelu je na Obr. 3-6.



Obr. 3-6: Ukázka HSV segmentace

Složka	Meze při rozsahu 0 - 1	Meze při rozsahu 0 - 255
Hue	0 – 0,018 0,66 - 1	0 – 4 169 - 255
Saturation	0,31 - 1	80 - 255
Value	-	-

Tab. 3-1: Meze HSV prahování

Mezi výhody tohoto řešení patří především přesná segmentace při dobrých světelných podmínkách. Použitím dvou kanálů obrazu se také často vyfiltruje vliv objektu v pozadí, který má například červený odstín, a tedy vejde se do mezí H složky, ale vlivem jiné hodnoty saturace je ve finální binární mapě vyfiltrován. Nevýhodou je vyšší výpočetní náročnost, neboť jde o tři postupné prahování plus jejich logickou kombinaci a to vše na celém obrazu.

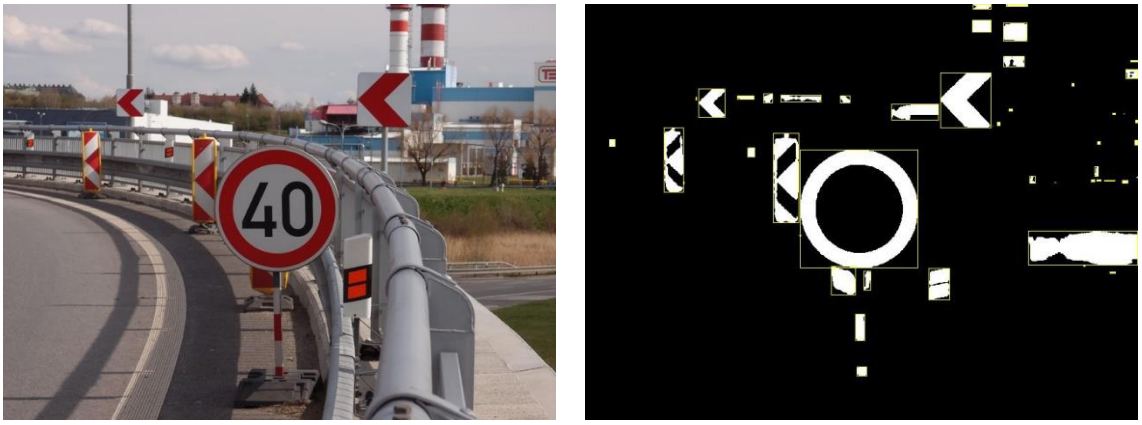
3.2.2. Algoritmy detekce značek

Jako další krok zpracování segmentovaného obrazu je detekce, neboli určení oblastí, které opravdu odpovídají kruhovým dopravním značkám. Tato detekce probíhá ve dvou krocích, nejdříve dojde k nalezení spojitých segmentovaných oblastí a k filtraci potenciálních dopravních značek na základě konkrétních rozměrů oblastí a v druhém kroku už ke konkrétnímu zjištění, zda se jedná o tvar odpovídající kruhové dopravní značce.

3.2.2.1. Analýza spojitých částí na binární mapě

Pro analýzu spojitých oblastí byl využit *system object* (a Simulink blok) *Blob Analysis* obsažený v Computer Vision Systém Toolboxu. Tento objekt slouží k vypočtení statistických údajů pro spojité oblasti binárních map. Je schopný vrátit např. polohu oblasti, rozměry, obvod, těžiště a další statistické údaje (více v [17]). Pro získání těchto údajů je potřeba správně nastavit vlastnosti objektu při jeho deklaraci.

Vzhledem k potřebám v dalších krocích algoritmu bylo jako výstup nastaveno jen ohraničení spojité oblasti – bounding box – ve formě vektoru souřadnic levého horního rohu a šířky a výšky: [x y šířka výška]. Dalším důležitým nastavením bylo nastavení konektivity na 4. Do spojité oblasti se zahrnují jen pixely ve 4-okolí a ne pixely sousedící s jinými jen diagonálně, což omezilo vliv šumu v okolí segmentovaných oblastí. Maximální počet spojitých oblastí byl nastaven na 500 a experimentálně bylo zjištěno, že jde o hranici s významnou rezervou. Ukázka funkce objektu *Blob Analysis* je na Obr. 3-7.



Obr. 3-7: Ukázka výstupu analýzy spojitých oblastí

Jak lze vidět i v obrázku, výstupem analýzy spojitých oblastí je několik různých bounding boxů, z nichž jen jeden odpovídá značce. Aby bylo co nejvíce „falešných“ oblastí vyřazeno, byl dále zařazen filtr, který nalezené bounding boxy filtroval podle rozměrů. Do dalšího kroku detekce se dostaly jen ty, které splňovaly následující podmínky:

- Oba rozměry (šířka, výška) jsou větší než 16 pixelů.
- Poměr stran (šířka/výška i výška/šířka) je minimálně 2/3.

Obě tyto podmínky byly experimentálně doladěny na základě schopností dalších částí algoritmů rozpoznat rychlost z takto omezených obrázků značek. Samozřejmě i přes tento filtr se dostaly oblasti, které neodpovídaly značkám, a bylo cílem dalších kroků tyto oblasti eliminovat.

3.2.2.2. Detekce na základě porovnání

První z algoritmů detekce je jednoduché porovnání. Funguje na principu přímého porovnání normalizovaného obrazu značky se vzorem odpovídajícím právě kruhové značce. Potenciální značka je nejprve vybrána z obrazu na základě bounding boxu, transformována na normalizovanou velikost 16x16 pixelů a následně vypočten koeficient podobnosti podle rovnice 1.1.

$$s = \frac{\sum |T - I|}{16 \times 16}, \tag{1.1}$$

kde

s	koeficient podobnosti,
T	matice vyjadřující vzor,
I	matice vyjadřující potenciální značku,
Σ	vyjadřuje součet všech prvků matice.

Koeficient podobnosti je díky dělení celkovým počtem pixelů v normalizované podobě, nabývá tedy hodnot 0-1. Samozřejmě vlivem segmentace, perspektivní deformace a jednoduché normalizace obrazu nemůžeme čekat stoprocentní shodu. Byl tedy stanoven práh pro prohlášení za kruhovou značky v hodnotě 0,75.

Obr. 3-8: Ukázka porovnání se vzorem. Výsledná podobnost 0,885.

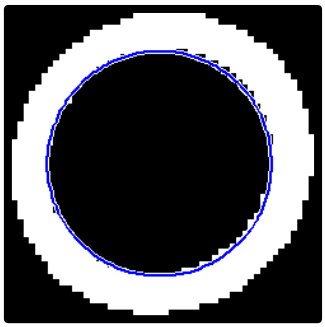
3.2.2.3. Detekce použitím Houghovy transformace pro kruhy

Druhým algoritmem detekce je použití Houghovy transformace pro kruhy. MATLAB obsahuje funkci pro tuto transformaci ve svém Image Processing Toolboxu: jde o funkci *imfindcircles*. Funkce hledá kruhové vzory v rámci obrazu na základě zadaného rozsahu poloměrů a je možné její nastavení dále specifikovat, výstupem jsou souřadnice středu a poloměry. Pro potřeby detekce kruhové značky bylo nastavení zvoleno následovně

- Rozsah poloměru: 15 – 28 pixelů
- Sensitivity: 0,95
- Object Polarity: „dark“

Parametr Sensitivity určuje, jak přesné kruhy požadujeme. Vysoká hodnota odpovídá méně přesným, i nekompletním kruhům. A Object Polarity určuje, na jakém rozhraní má funkce kruhy hledat. Nastavením na „dark“ hledá tmavé kruhové objekty, což v případě značek znamená vnitřní (bílý, po segmentaci tedy černý) kruh značky. Toto nastavení je důležité především z hlediska falešných detekci, neboť v přirozeném prostředí se často vyskytují červené kruhové objekty, které nejsou značkami, a které by byly při opačném nastavení detekovány. Červených „prstenců“ s vnitřním kruhem jiné barvy je však výrazně méně.

Algoritmus detekce začíná opět normalizací, tentokrát však na 50x50 pixelů vzhledem k minimálnímu poloměru, který funkce dokáže nalézt a po provedení hledání kruhových objektů je zkontrolován výstup funkce. V případě, že je kruh nalezen, je objekt považován za detekovanou značku, v opačném případě nikoliv. Příklad nalezeného kruhu je na Obr. 3-9.



Obr. 3-9: Kruh nalezený prostřednictvím Houghovy transformace

3.2.3. Algoritmy normalizace obrazu

Vzhledem k následujícímu kroku extrakce čísla ze značky pro rozpoznání, bylo potřeba perspektivně deformované obrazy normalizovat na jednotnou velikost a poměr 1:1. Jako normalizovaná velikost byla zvolena velikost 50x50 pixelů. Na tuto velikost normalizujeme výřez značky, detekované v předchozích krocích. Před samotnou normalizací ještě dochází k převedení tříkanálového RGB obrazu na jednokanálový obraz obsahující odstíny šedé barvy. Pro normalizaci byly vyzkoušeny následující tři algoritmy.

3.2.3.1. Základní metoda normalizace

Nejjednodušší metodou normalizace je poměrová (proporcionální) transformace. Obrázek je v tomto případě proporcionálně roztažen/zmenšen aby odpovídal normalizované velikosti. K výpočtu hodnoty jednotlivých pixelů je použita bilineární interpolace [18]. Ukázka funkce této metody je na Obr. 3-10.

Obr. 3-10: Ukázka funkce základní metody normalizace.
Zleva: Originální obraz, Normalizovaný na 50x50 pixelů (2x zvětšený pro názornost)

Tato jednoduchá metoda se osvědčila a až na velmi perspektivně deformované značky vedla ke správnému rozpoznání. Jelikož následující dvě metody nevedly v celkovém měřítku k ideálním výsledkům, byla metoda nakonec použita ve finálním řešení.

3.2.3.2. Normalizace podle Čípa

Ve své práci [4] popisuje Pavel Číp použité metody normalizace, pro dopravní značky, kdy na základě vyhledaných hran objektu určí souřadnice rohů značky a pomocí

perspektivní transformace je normalizuje do standardního tvaru. Na závěr zmiňuje metodu pro kruhové značky, kde využívá faktu, že kruh se v perspektivě deformuje na elipsu. Nalezne hlavní a vedlejší poloosy elipsy, pomocí vektorového počtu vyhledá odpovídající „vrcholy“ a perspektivně značku transformuje. Výsledek transformace je potřeba rotovat o úhel, o který byla vychýlena hlavní osa elipsy. Bohužel ve své práci nepopisuje detaily řešení, jako jsou metody vyhledání poloos ani teorii, ze které při této normalizaci vycházel.

Na základě popisu byl vytvořen algoritmus, který měl za cíl normalizaci zopakovat. Poloosy elipsy byly získány prostřednictvím volitelných výstupů funkce *Blob Analysis*: délky hlavní osy, délky vedlejší osy a orientace hlavní osy. Aby tyto odpovídaly přímo elipse vzniklé segmentací značky, bylo potřeba segmentovaný tvar, tedy prstenec, v binární mapě „vyplnit“, čehož bylo dosaženo funkcí *imfill*. Po získání os, byl jednoduchým vektorovým počtem vypočten obdélník opisující elipsu. Pomocí těchto bodů a bodů cílového normalizovaného obrazu byl proveden výpočet transformační matice a provedena perspektivní transformace. Vzniklý obraz byl dále rotován a opět ořezán na normalizovanou velikost. Postupné kroky normalizace je možné vidět na Obr. 3-11.

Obr. 3-11: Postupné kroky algoritmu podle Čípa.
Zleva: Nalezení elipsy a vrcholů, transformovaná značka, výsledek po natočení a oříznutí

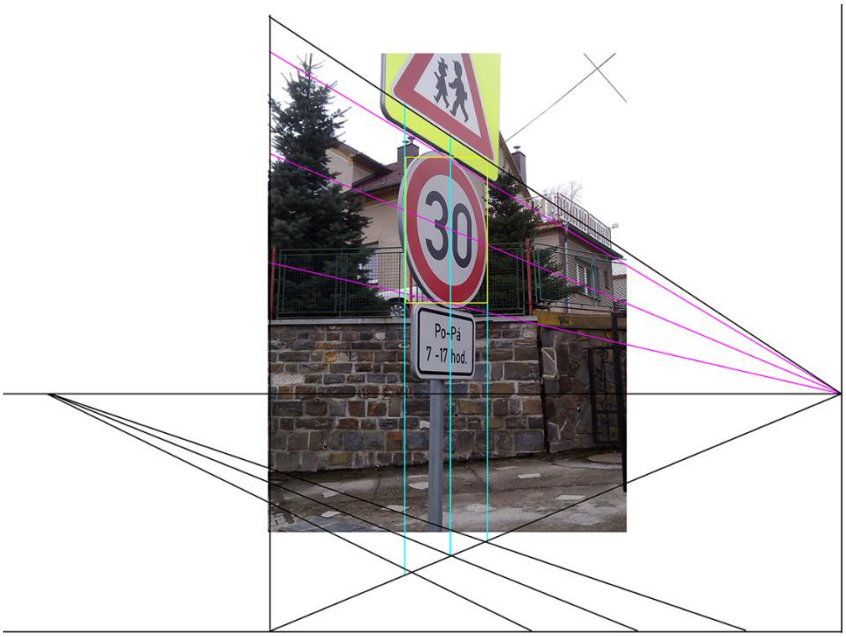
Po otestování normalizace na několika perspektivně deformovaných značkách se ukázalo, že transformace není ideální. U normalizované značky bylo patrné natočení, nebo špatná transformace. Není jasné, zda jde o problém přímo této metody, nebo její nepřesné implementace. Z těchto důvodů, a z důvodů nejasné teorie za tímto algoritmem bylo upuštěno od jeho použití.

3.2.3.3. Perspektivní transformace na základě hledání klíčových bodů

Vlivem nepřesnosti předchozí metody vznikla snaha vytvořit vlastní algoritmus perspektivní transformace pro kruhové značky. Cílem bylo nalézt klíčové body na segmentovaném perspektivně deformovaném obrazu, které by jasně odpovídaly bodům

na normalizované značce a následně prostřednictvím těchto bodu vypočíst transformační matici a provést transformaci. Pro výpočet transformační matice pro perspektivní transformaci je potřeba minimálně 4 dvojic vzájemně si odpovídajících bodů.

K výběru a nalezení klíčových bodů byly použity zásady lineární perspektivy. Po vynesení pomocných čar do obrázku reálné značky (Obr. 3-12) se ukázalo jako možné řešení nalezení těchto bodů: levý, pravý, horní a dolní krajní bod značky a její střed.



Obr. 3-12: Perspektiva dopravní značky

V MATLABu bylo nalezení klíčových bodů provedeno následovně. Nejprve byla potenciální značka oříznuta podle ohraničujícího obdélníku. Následně bylo provedeno prohledání levého okraje obrazu na bílé pixely segmentované značky a vrácen střed nalezené oblasti bílých pixelů. Pro větší robustnost algoritmus prohledával několik pixelů dopředu, aby byl omezen vliv ojedinělých pixelů a děr. V případě malého počtu pixelů na okraji se pak celý algoritmus posunul o sloupec vedle. Stejný postup byl proveden i na pravém okraji. Tím byly získány souřadnice levého a pravého okraje značky.

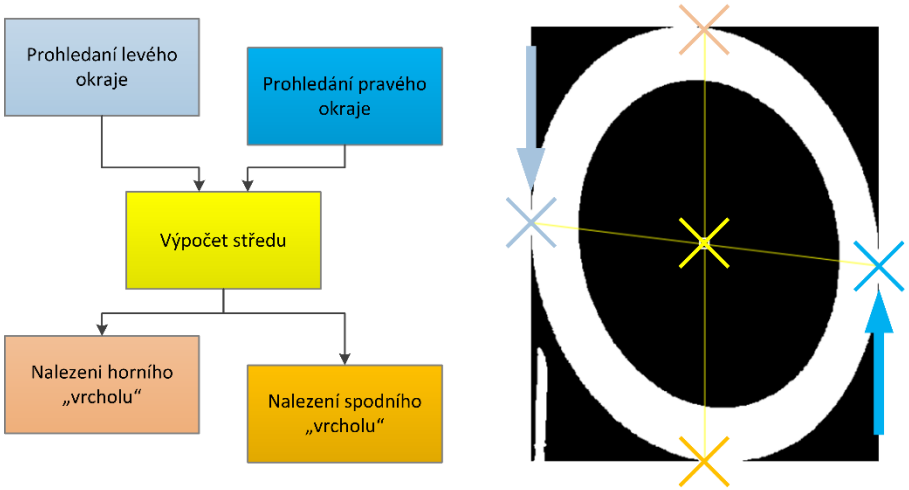
Pomocí těchto bodů byly vypočteny souřadnice středu. Ty byly získány na základě úvahy, že v příčném směru nedochází k perspektivní deformaci a střed se nachází v polovině výšky obrazu značky. Byl zanedbán vliv vrcholů výsledné elipsy, které se vlivem podélné perspektivní deformace nacházejí lehce nad bodem odpovídajícím reálnému „vrcholu“ značky. Tím byla získána y-ová souřadnice středu a x-ová byla dopočtena na základě rovnice 1.2 vycházející ze základních rovnic analytické geometrie.

$$x_S = x_L + \frac{(x_P - x_L)}{(y_P - y_L)} \cdot \left(\frac{y_{max}}{2} - y_L\right) \qquad (1.2)$$

kde

x_S	x-ová souřadnice středu,
x_P, y_P	souřadnice pravého krajního bodu,
x_L, y_L	souřadnice levého krajního bodu,
y_{max}	počet pixelů obrazu ve směru y.

Po nalezení souřadnic středu proběhlo nalezení horního a spodního okraje značky, čehož bylo dosaženo prohledáváním obrazu od horního, respektive spodního, okraje na úrovni středu až po nalezení prvního bílého pixelu. Celý algoritmus hledání klíčových bodů je znázorněn na Obr. 3-13.

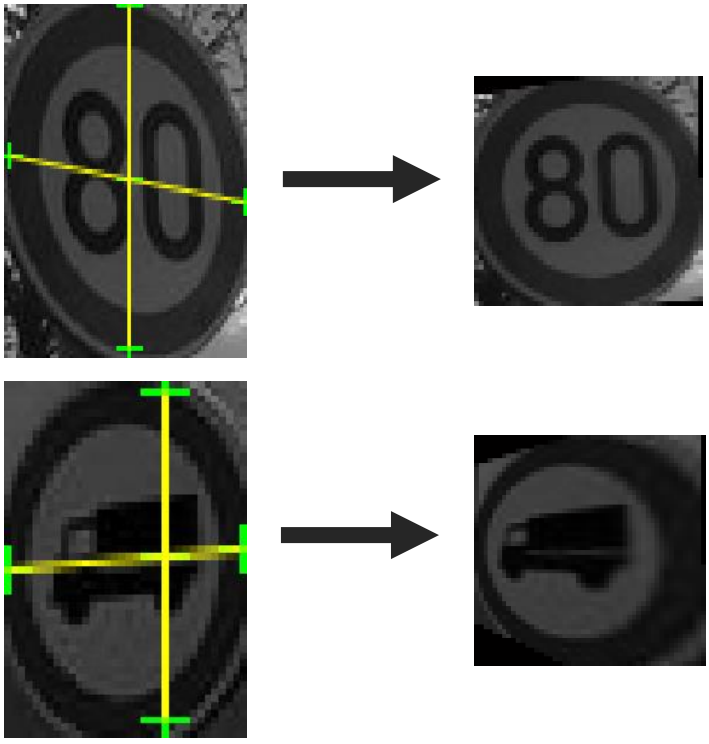


Obr. 3-13: Schéma hledání klíčových bodů

Z takto získaných klíčových bodů a jim odpovídajícím bodům na normalizované značce byla vypočtena transformační matice a provedena perspektivní transformace.

Při testování tohoto řešení se objevil problém v případě falešných detekcí, kdy pokud binární mapa neodpovídá kruhové značce, může prohledávání skončit v podstatě kdekoli. Špatně nalezené body neznamenaly až takový problém, kromě nesmyslných obrazů po transformaci, ale pokud se prohledávání dostalo mimo obraz, vracel MATLAB chybu. Toto bylo ošetřeno vnitřním systémem chyb, kdy pokud toto nastalo, funkce vrátila chybu a nadřazená funkce pak značku považovala za falešnou detekci.

Výsledky algoritmu závisely především na velikosti značky, případně na kvalitě segmentace. U značek, které měly na výšku přibližně 100 pixelů a výše byly výsledky hledání klíčových bodů a z toho plynoucí normalizace velmi dobré. U menších obrazů značek však algoritmus často selhával. To bylo způsobeno především velkou diskretizací značek, kdy nalezené klíčové body zcela neodpovídaly reálným krajním bodům, případně byl nepřesný vypočtený střed, což vedlo i k špatným souřadnicím horního a spodního krajního bodu. Příklady takto normalizovaných značek jsou na Obr. 3-14.



Obr. 3-14: Ukázka dobré a špatné normalizace na základě hledání klíčových bodů

Z důvodu nejistých výsledků, především u menších značek, nebyl ani tento algoritmus ve výsledném řešení použit a veškeré normalizace jsou prováděny jednoduchou proporcionální metodou.

3.2.4. Rozpoznání rychlosti

Různými metodami rozpoznání vnitřního piktogramu značky, od umělých neuronových sítí po porovnání, se zabývaly všechny práce, které byly zmíněny v kapitole 2.2. A ve všech bylo na závěr použito právě porovnání se vzorem, neboť jde o jednoduchý algoritmus, který přinášel nejlepší výsledky. Z tohoto důvodu bylo porovnání se vzorem použito i v této práci.

3.2.4.1. Porovnání se vzorem

Prvním krokem porovnání se vzorem je extrakce piktogramu ze značky. Vzhledem k zaměření na značky omezující rychlost bylo rozhodnuto o omezení extrakce pouze na první číslo, například u značky s číslem 20 extrahujeme pouze oblast s číslici 2, protože ve většině případů by 0 byla vždy shodná a tím pádem nám zvyšovala podobnost mezi jednotlivými rychlostmi a tím pravděpodobnost špatného rozpoznání. Vybraná oblast u čísel nad 100 znamenala extrakci 1 a části následujícího čísla, což bylo považováno pro porovnání za dostačující. Celková oblast, kterou extrahujeme z normované (50x50 pixelů) značky má souřadnice levého horního rohu [11, 13] a šířku 15 a výšku 26 pixelů.

Na základě zvolené oblasti byla vytvořena sada vzorů pro porovnání, extrakci ze vzorových značek a prahováním, čímž byly získány binární vzory pro značky odpovídající 10 – 130. Poté byly vzory ještě ručně upraveny na základě zkoušky porovnání na sadě obrázků odpovídajících dané značce.

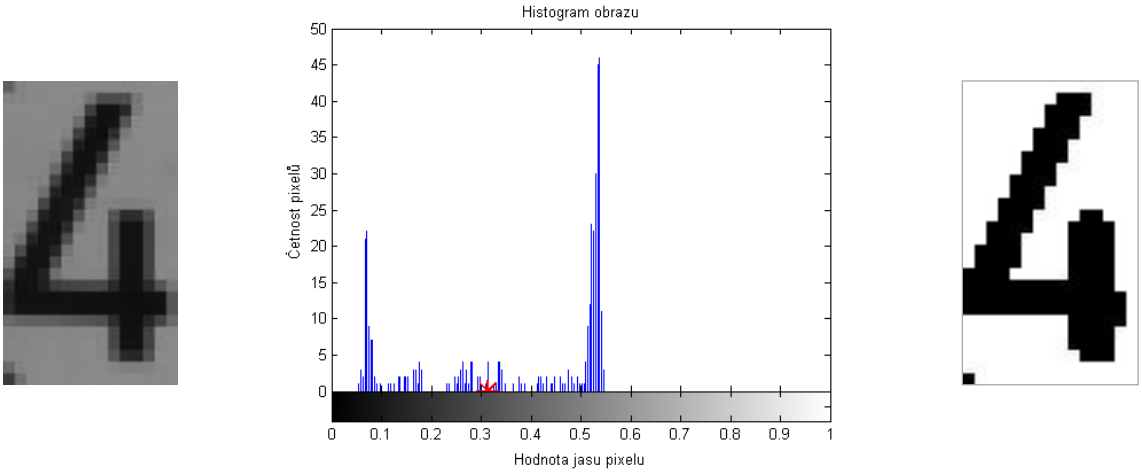
Vzhledem k častému výskytu dalších zákazových značek, které byly algoritmem detekovány bylo rozhodnuto o rozšíření vzorů nad rámec značek omezujících rychlost. Bylo vytvořeno sedm dalších vzorů pro nejčastěji se vyskytující (především v městském prostředí) zákazové značky: Zákaz vjezdu všech vozidel, Zákaz vjezdu nákladních automobilů, Zákaz stání, Zákaz zastavení, Zákaz odbočování vlevo, Zákaz odbočování vpravo a Zákaz předjíždění. Tímto přidáním dalších vzorů jsme kromě rozšířené funkčnosti celého algoritmu detekce snížili pravděpodobnost falešné detekce rychlosti u těchto značek.

Výsledná sada vzorů je na Obr. 3-15. V rámci programu se vyskytuje v souboru *Vzory_vse.mat* a k jejímu načtení dojde při inicializaci.

Z důvodu zjištění případných problémů při rozpoznání - záměny jedné značky za druhou - bylo provedeno porovnání všech vzorů mezi sebou na vzájemnou podobnost. Jako velmi podobné se ukázaly vzory pro značky omezení rychlosti na 100, 120 a 130, při testování ale k záměně nedošlo a nebyly vytvořeny žádné opatření proti záměně.

Obr. 3-15: Použité vzory k porovnání se vzorem

Algoritmus rozpoznání pracuje tak, že nejdříve vybere oblast, která obsahuje číslo (podle daných souřadnic) a poté následuje prahování vybraného obrazu. Jako metoda prahování byla zvolena Otsuho metoda, která spočívá v nalezení optimálního prahu na základě histogramu obrazu. Otsuho metoda pracuje s předpokladem, že obraz obsahuje 2 třídy pixelů, jedny pro objekt a druhý pro pozadí a na základě minimalizace rozptylu uvnitř třídy vypočte ideální práh pro rozdělení těchto tříd [19]. V případě rozpoznání piktogramu jde o ideální metodu, kdy číslo představuje objekt a bílé (respektive šedé) okolí je považováno za pozadí. V prostředí MATLAB je Otsuho metoda použita v rámci objektu *Autothresholder*. Ukázka funkce je na Obr. 3-16.



Obr. 3-16: Ukázka prahování Otsuho metodou.
Zleva: Obraz před prahováním, Jeho histogram s vyznačeným prahem, Výsledný binární obraz

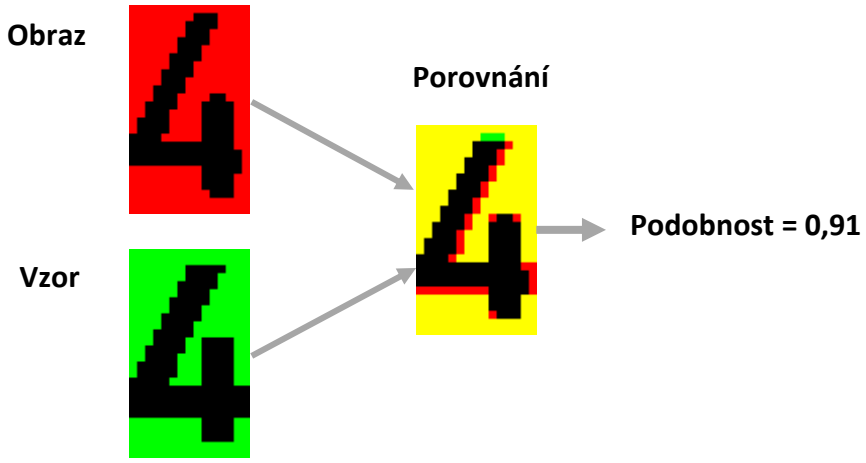
Vzniklý binární obraz čísla je poté porovnán se vzory, na základě obdobného vztahu jako u detekce porovnáním:

$$s = \frac{\sum |T - I|}{15 \times 26}, \tag{1.3}$$

kde

s	koeficient podobnosti,
T	matice vyjadřující vzor,
I	matice vyjadřující obraz pro rozpoznání,
Σ	vyjadřuje součet všech prvků matice.

Podobnost je ve smyčce vypočtena pro všechny dostupné vzory a následně je vybrán vzor s největší mírou shody. Pokud shoda překoná mez 0,76, je značka považována za rozpoznanou a je k ní přiřazeno odpovídající číslo. O zvolení této meze pojednává kapitola 3.3.3.2. Jedinou výjimkou je, pokud je vzorem s největší shodou vzor odpovídající značce Zákaz vjezdu všech vozidel. Díky velké míře shody s dalšími zákazovými značkami, které mají jen drobný piktogram, případně se značkou upravující rychlost, která je přísněji prahována, je u této značky nastaven limit podobnosti na 0,95. V případě, že tento limit není překonán, je za nejpodobnější značku považována druhá v pořadí.



Obr. 3-17: Porovnání extrahovaného čísla se vzorem

V případě, že není žádným vzorem překonána mez, dochází ještě k rozšířenému porovnání, protože často dojde k situaci, kdy je obraz vlivem jednoduché normalizace jen lehce posunut v rámci vybrané oblasti. V takovém případě dochází často ke „správnému“ rozpoznání, kdy největší podobnost má ten správný vzor, ale nedojde k překonání minimální meze podobnosti. Rozšířené porovnání spočívá v posouvání vzoru o jeden pixel ve všech 8 nejbližších směrech a vypočtení podobnosti po každém posunutí. Jakmile dojde k překonání meze podobnosti, je značka považována za rozpoznanou.

Obr. 3-18: Rozšířené porovnání – posouvání vzorů

Posledním krokem algoritmu rozpoznání je přiřazení číselného kódu k rozpoznaným a nerozpoznaným značkám. Pro značky omezující rychlost kód odpovídá rychlosti, pro ostatní značky lze vidět v Tab. 3-2.

Kód 250 označuje „jinou zákazovou značku“ neboli značku, který byla detekována, ale nebyla rozpoznána a to ani rozšířeným porovnáním. Na základě testů byl kód 250 přiřazen i rozpoznaným značkám Zákaz stání a Zákaz zastavení, protože úspěšnost jejich rozpoznání byla malá. To bylo způsobeno tím, že místo bílého obsahují pozadí modré, které nebylo možné spolehlivě rozlišit Otsuho metodou. V případě cílení i na tyto značky by bylo potřeba použít jinou metodu.

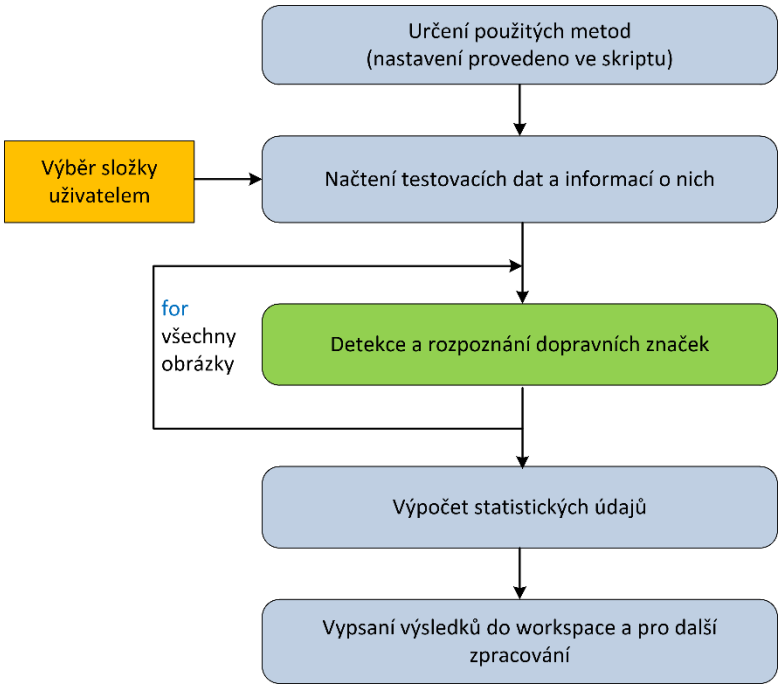
Rozpoznaná značka	Kód rozpoznání
Nejvyšší povolená rychlost (10 – 130)	10 - 130
Zákaz vjezdu všech vozidel	140
Zákaz vjezdu nákladních automobilů	150
Zákaz odbočení vlevo	180
Zákaz odbočení vpravo	190
Zákaz předjíždění	200
Jiná zákazová značka (nerozpoznáno)	250

Tab. 3-2: Číselné kódy rozpoznaných značek

3.3. Ladění a testování algoritmů

3.3.1. Testovací skript

Pro testování vyvinutých řešení byl vytvořen testovací skript, ve kterém jsou implementovány kromě samotných algoritmů i další funkce potřebné pro uživatelsky přívětivé testování. Schéma testovacího skriptu je na Obr. 3-19.



Obr. 3-19: Schéma testovacího skriptu

Skript nejprve obsahuje nastavení jednotlivých modulů, které mají být použity v celkovém algoritmu, pro testování jednotlivých řešení. V rozšířené verzi je pak toto řešeno smyčkou, která projede všechna relevantní nastavení. Následuje výběr složky obsahující testovací množinu, ke které dochází volbou uživatele. Aby skript dokázal

vypočítat statistické údaje, musí být testovací obrázky pojmenovány podle následujícího vzoru:

kódZnačky1_kódZnačky2 libovolný text.jpg

Například tedy „60_250 test.jpg“, nebo „90 noc.jpg“ v případě jedné značky na obrázku.

Poté je pro každý obrázek z množiny volána funkce *Detekce_a_rozpoznání*, obsahující v rámci jednotlivých podfunkcí všechny moduly, která provede celý algoritmus podle nastavení a vrací rozpoznanou rychlost. Pro každý běh funkce je metodou *tic-toc* měřen čas jejího prováděni z důvodu měření výpočetní náročnosti.

Po provedení detekce a rozpoznání dopravních značek na všech obrázcích množiny, jsou vypočteny následující statistické údaje:

- Průměrná rychlost výpočtu
- Počet testovacích značek
- Počet správně detekovaných značek
- Počet správně rozpoznaných značek
- Počet falešně detekovaných značek
- Počet falešně rozpoznaných značek
- Úspěšnost detekce v %
- Úspěšnost rozpoznání z celkového počtu značek v %
- Úspěšnost rozpoznání z detekovaných značek v %

Úspěšnosti v procentech jsou vypočteny podle vztahu 1.4.

$$Úspěšnost = \frac{Počet\ správně\ \ldots}{Celkový\ počet} \cdot 100, \qquad (1.4)$$

Posledním krokem skriptu je výpis nastavení a výsledků do workspace MATLABu a do proměnné *VysledkyExcel*, která slouží k jednoduchému kopírování do programu MS Excel z důvodu dalšího zpracování.

3.3.2. Testovací množina

Pro testování funkčnosti algoritmů bylo vytvořeno několik množin obrázků. Tyto se skládaly z fotografií značek omezujících rychlost a dalších kruhových zákazových značek. Značky byly většinou pořízeny přímo z jedoucího automobilu, aby byly relevantní pro zamýšlené použití. Všechny testovací fotografie byly pořízeny fotoaparátem Olympus XZ-1, který má následující základní parametry, více v [20]:

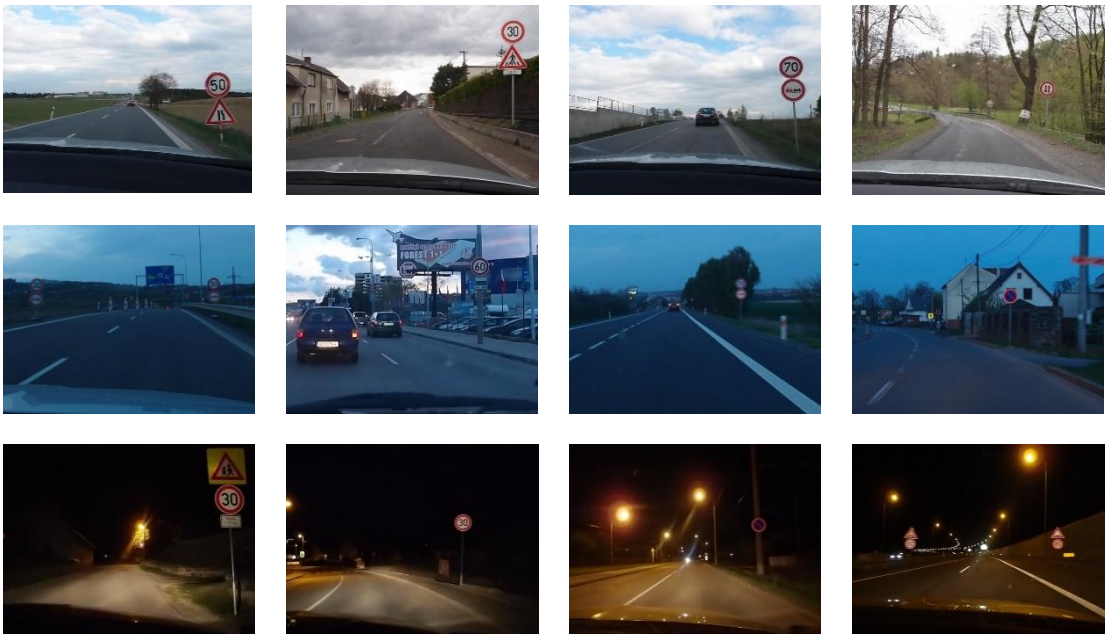
- Formát snímače: 1/1,63“
- Rozlišení: 10 MPx

- Typ snímače: CCD
- Základní rozlišení: 2560x1920 Px

Kromě fotografií pro testování výpočetní náročnosti v závislosti na rozlišení byly všechny fotografie zmenšeny na velikost 800x600 pixelů.

Pro testování spolehlivosti detekce v závislosti na světelných podmínkách byly pořízené fotografie rozřazeny do tří množin, ukázky fotografií z jednotlivých množin jsou na Obr. 3-20:

- Den – obsahuje fotografie pořízené za dobrých světelných podmínek
- Šero – obsahuje fotografie pořízené v různých fázích stmívání
- Noc – obsahuje fotografie pořízené při umělém osvětlení světlomety automobilu



Obr. 3-20: Příklady fotografií v jednotlivých testovacích množinách.
Shora po řádcích jsou množiny den, šero a noc

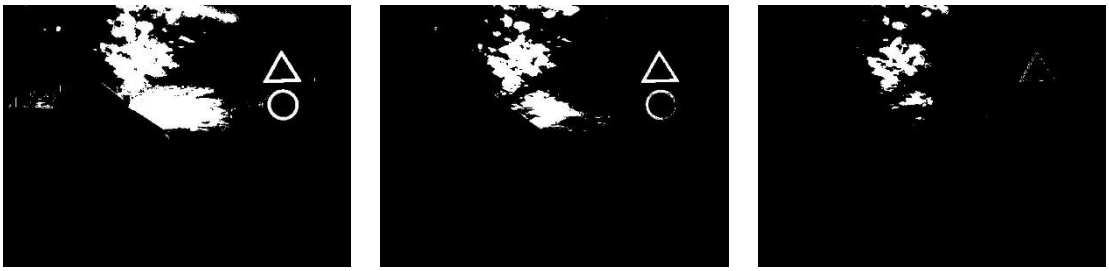
3.3.3. Ladění parametrů algoritmu

Během práce na algoritmu se ukázalo, že úspěšnost detekce a rozpoznání často závisí na co nejpřesnějším nastavení několika parametrů, jako jsou například meze pro prahování. Jelikož při zcela manuálním ladění byla složitá kontrola správnosti řešení, bylo nakonec ladění parametrů vyřešeno metodou prohledávání. Byla sestavena reprezentativní množina pro ladění, a použitím testovacího algoritmu byly ozkoušeny parametry v blízkosti pravděpodobné ideální hodnoty. Následně byl vybrán parametr, který nejvíce splňoval požadavky, což většinou znamenalo maximalizaci správných

výsledků (detekce, rozpoznání) při udržení falešných pozitivních nálezů na minimální míře.

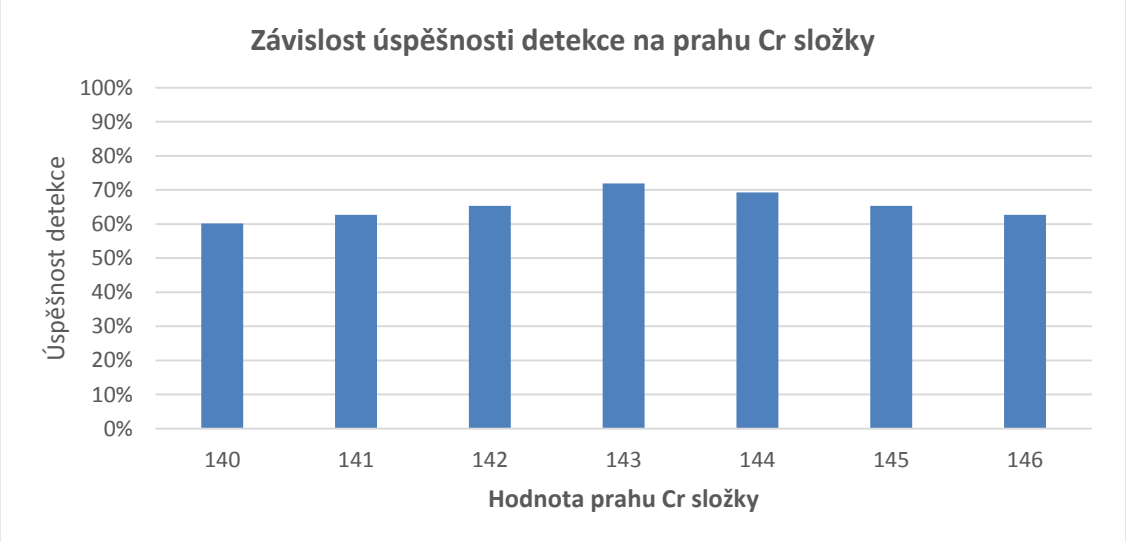
3.3.3.1. Ladění prahu pro YCbCr segmentaci

Při segmentaci podle statického prahu v Cr složce veškerý úspěch závisí právě na zvolené hodnotě prahu, proto je důležité, aby byla určena co nejlépe. Na Obr. 3-21 lze vidět, jak ovlivňují i malé odchylky celkovou segmentaci.



Obr. 3-21: Stejná fotografie segmentována prahy s hodnotami 138, 143 a 148

Proto bylo provedeno ladění prahu v okolí hodnoty 140, aby bylo nalezeno co nejlepší řešení. Výstupem byla celková úspěšnost detekce. Výsledky lze vidět na Obr. 3-22.

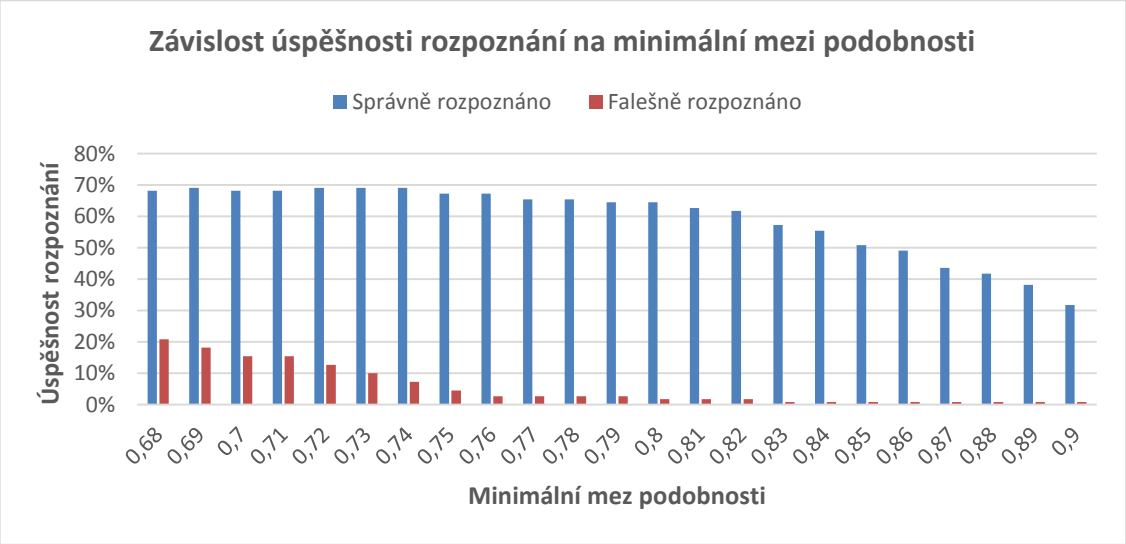


Obr. 3-22: Graf závislosti detekce na hodnotě prahu Cr složky

Nejúspěšnější je z pohledu detekovaných značek hodnota 143 (0,56), a tato hodnota byla implementována do finálního algoritmu. Za povšimnutí stojí fakt, že při změně prahu pouze o 3 hodnoty úspěšnost detekce klesá o 10%.

3.3.3.2. Ladění limitu pro rozpoznání značky

Další parametr, který byl nastaven na základě prohledávání, byl limit pro rozpoznání značky, neboli minimální hodnota podobnosti, kterou musí vzor splňovat, aby byl prohlášen za správné řešení. V tomto případě bylo velmi důležité sledovat počty falešných rozpoznání. Výsledky pro jednotlivé limity jsou na Obr. 3-23.



Obr. 3-23: Graf závislosti rozpoznání na hodnotě minimální meze podobnosti

Na základě těchto výsledků byla jako minimální hodnota podobnosti zvolena 0,76, protože vykazuje vysoké hodnoty úspěšného rozpoznání při velmi nízkém počtu falešných rozpoznání (pod 3%).

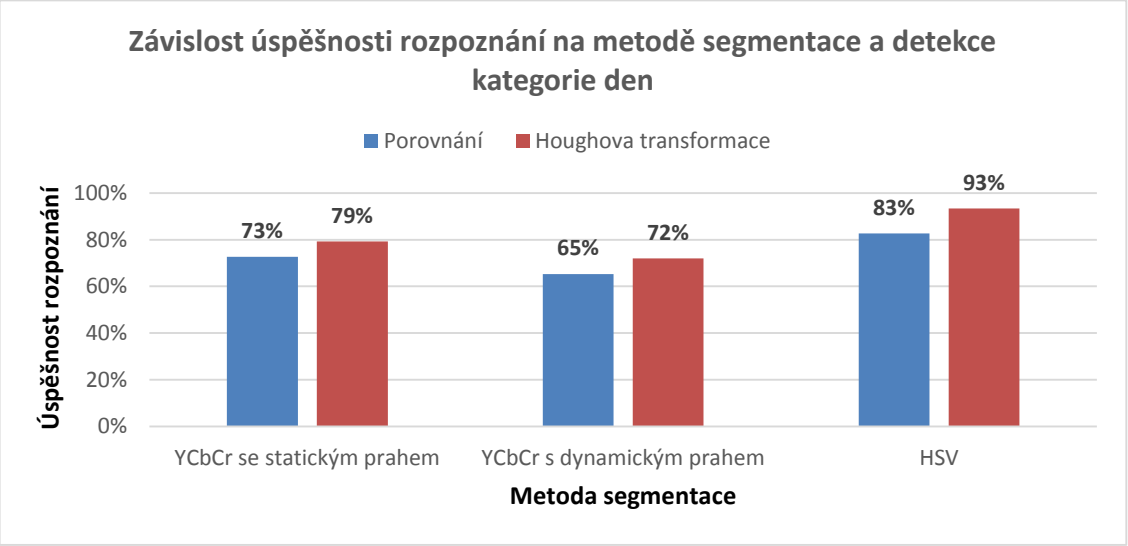
3.3.4. Test segmentace a detekce značky

3.3.4.1. Úspěšnost metod v závislosti na světelných podmínkách

Pro výběr nejlepší metody segmentace a detekce značky byly postupně provedeny testy na výše zmíněných množinách fotografií (den, šero, noc). Testy byly provedeny pro všechny tři metody segmentace (YCbCr s pevným prahem, YCbCr s dynamickým prahem a HSV) vždy v kombinaci s jednou ze dvou metod detekce (porovnání a Houghova transformace). Hlavními sledovanými parametry byly úspěšnost rozpoznání a rychlost výpočtu u jednotlivých kombinací. Úspěšnost rozpoznání před úspěšnosti detekce byla zvolena na základě předpokladu, že u správně detekované značky proběhne i správné rozpoznání. Byly také sledovány hodnoty falešných detekcí a rozpoznání, ale u všech metod se držely na nízkých hodnotách a nebyly proto brány příliš v úvahu. Výsledky veškerých testů obsahují Příloha 5 - Příloha 8. Testy byly prováděny na starším počítači, jehož parametry jsou v Tab. 3-3:

Výrobce	ASUS
Model	K50AB
Procesor	AMD Athlon™ X2 Dual-Core QL-65
Frekvence	2.10 GHz
RAM	4 GB
Op. Systém	64 bit Windows 8 Pro
MATLAB	Verze 2013a

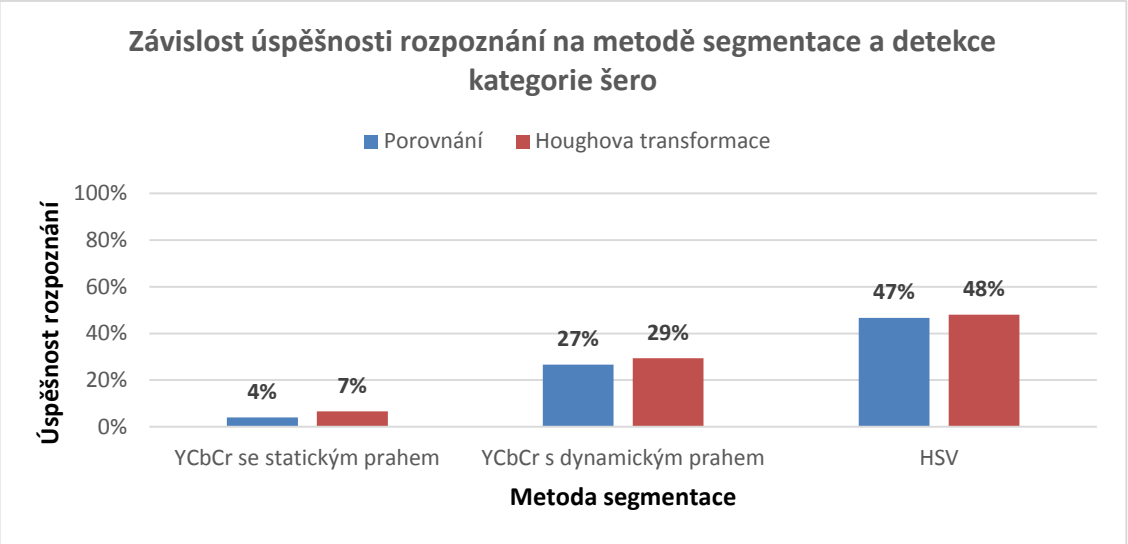
Tab. 3-3: Parametry zkušebního počítače



Obr. 3-24: Graf závislosti úspěšnosti rozpoznání na použité metodě, kategorie den

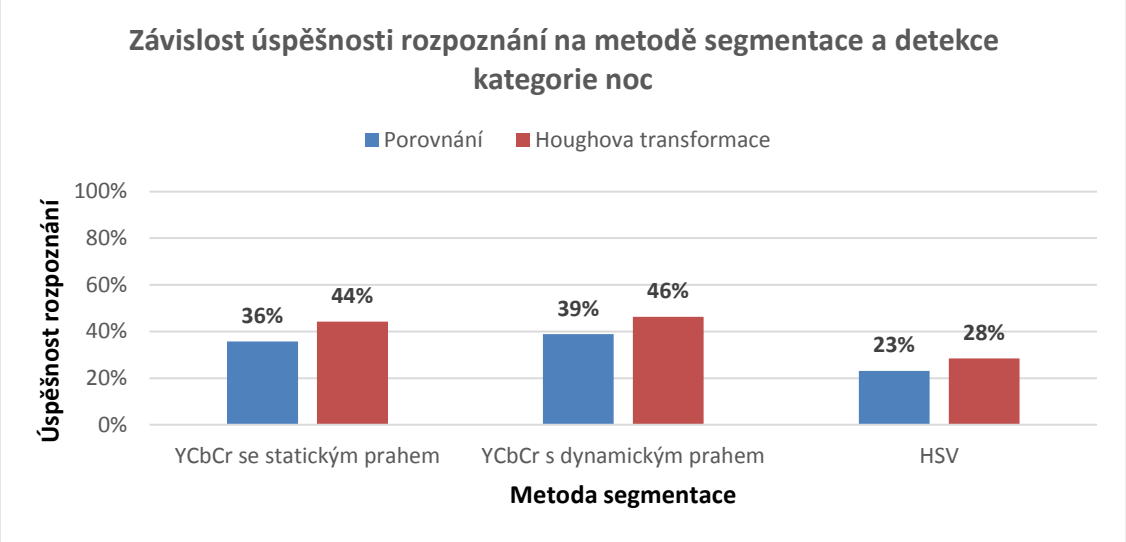
Na Obr. 3-24 lze vidět výsledky pro testovací množinu den. Nejlepší výsledky má metoda s HSV segmentací a detekcí pomocí Houghovy transformace, která dosahuje úspěšnosti 93 %. Obecně jde vidět lepší výsledky této metody detekce, před detekcí porovnáním. To může být způsobeno špatnými výsledky metody porovnání u značek Zákaz stání a především zastavení, u kterých vnitřní červené pruhy jsou segmentovány a zhoršují výsledek porovnání.

Úspěšnost u metody YCbCr se statickým prahem se pohybuje kolem 75 %, což je dobrý výsledek a je možné s touto metodou dále pracovat. Metoda s dynamickým prahem dosáhla horších výsledků, což je způsobeno tím, že její vývoj byl od začátku zaměřen na značky s horšími světelnými podmínkami.



Obr. 3-25: Graf závislosti úspěšnosti rozpoznání na použité metodě, kategorie šero

Na množině fotografií pořízených za šera si opět nejlépe vedlo prahování na základě barevného modelu HSV, úspěšnost je mnohem menší, než u snímků značek pořízených za dobrého osvětlení. Na těchto fotografiích se již projevila výhoda dynamického prahu u YCbCr segmentace, oprati statickému prahu, kdy první zmíněná metoda má úspěch v téměř třetině případů. Naopak metoda se statickým prahem nedokáže detekovat značky při sníženém osvětlení, neboť hodnota prahu byla naladěna na příkladech dobře osvětlených značek. Celkově jde v této kategorii vidět značný pokles. Ten je způsoben naladěním hodnot na denní světlo, ale je možné, že některé z testovaných metod by nebylo možné naladit a musely by se použít jiné úpravy obrazu.

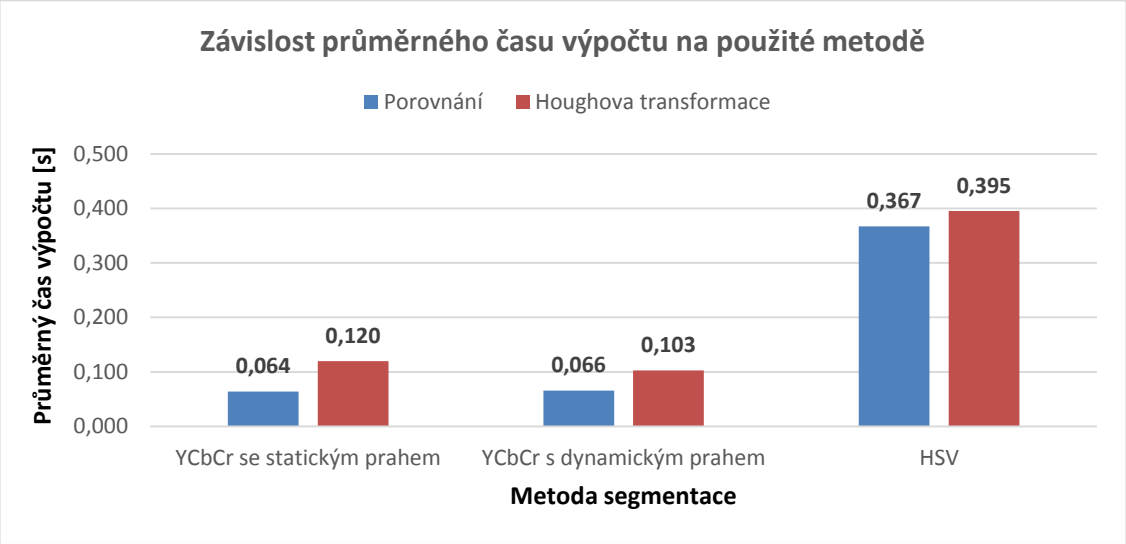


Obr. 3-26: Graf závislosti úspěšnosti rozpoznání na použité metodě, kategorie noc

Na Obr. 3-26 jsou výsledky testu na množině fotografií pořízených v noci pod umělým osvětlením světlometů automobilu. Vlivem výrazné červené barvy díky reflexní vrstvě dopravních značek, jsou výsledky prahování prostřednictvím YCbCr modelu lepší než v případě šera. Ale špatný vliv má z pohledu prahování oranžové světlo pouličního osvětlení, které také nabývá výšších hodnot v Cr kanálu. Co se týče porovnání obou metod pracujících s YCbCr modelem, jsou výsledky téměř shodné, s lehkou převahou dynamického prahu. U nočních fotek však selhává HSV prahování. To je způsobeno vysokými hodnotami kanálu saturace, které jsou způsobeny právě intenzivním osvětlením světlomety, a také zde se projevuje oranžové světlo pouličních lamp. Z těchto důvodů by bylo pro přesnou segmentaci v noci potřeba provézt výraznější úpravy těchto metod, případně zvážit metody jiné.

3.3.4.2. Výpočetní náročnost jednotlivých metod

Velmi důležitým parametrem pro hodnocení jednotlivých metod je jejich výpočetní náročnost. Pro porovnání rychlosti výpočtu byly vybrány průměrné rychlosti výpočtu při testu kategorie den. Tato kategorie byla zvolena z důvodu největšího počtu fotografií v množině a pak také kvůli vysokému počtu detekovaných značek u všech metod, neboť detekované značky by se také měly projevit v celkové výpočetní náročnosti vlivem následných funkcí pro normalizaci a rozpoznání. Výsledky porovnání jsou znázorněny na Obr. 3-27.



Obr. 3-27: Graf závislosti průměrného času výpočtu na použité metodě

Na výsledcích tohoto testu jdou vidět obrovské rozdíly mezi výpočetní náročností segmentace použitím YCbCr modelu a HSV. Tento rozdíl je způsoben třemi faktory. Zaprvé je potřeba si uvědomit, že segmentace probíhá na celém obrazu, v podstatě pixel po pixelu, a jde o nejnáročnější operaci. U YCbCr segmentací jde o porovnání jednoho kanálu s jedním prahem, tedy v podstatě jeden „běh“ celým obrazem. Naproti tomu u

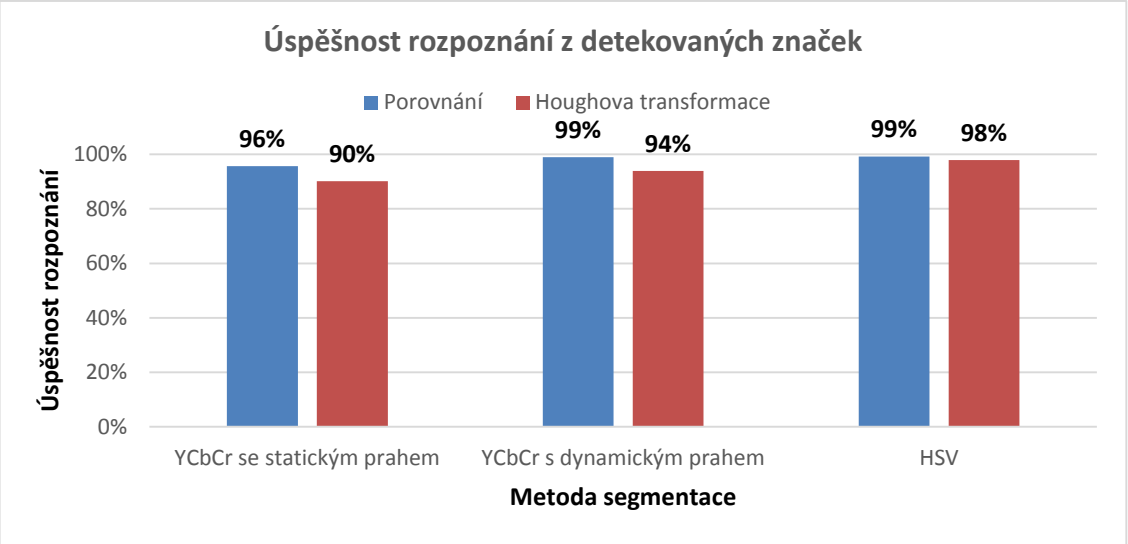
HSV segmentace dochází k porovnání dvou kanálů (H a S) dohromady se třemi prahy (spodní a horní práh u H složky a spodní u S složky) a tyto výsledky jsou ještě zkombinovány na základě logických operátorů, jde tedy o násobně složitější operaci. Zadruhé má jistý vliv převádění z RGB na HSV model, kdy nejde o lineární kombinaci jednotlivých složek jako je tomu u převodu na YCbCr model. A třetí faktor je jistá neflexibilita programu MATLAB, konkrétně funkce na převod z RGB do HSV prostoru. Tato funkce totiž vrací převedený obraz vždy jako datový typ double (tzn. každý pixel je vyjádřen typem double) bez ohledu na datový typ vstupního obrazu. Obecně platí, že operace s „většími“ datovými typy je náročnější než operace s menšími. Toto bylo potvrzeno i při optimalizaci programu, kdy explicitní deklarací datového typu uint8 pro veškerá obrazová data u algoritmu s YCbCr segmentací byl výpočetní čas snížen na polovinu oproti implicitně používanému datovému typu double. Pro optimalizaci problému s převodem na HSV model a následnou segmentací bylo vyzkoušeno několik metod (přetypování, segmentace použitím „double“ mezí atd.), ale žádná metoda neznamenala výrazné zlepšení. Nakonec bylo použito přetypování celého HSV obrazu na datový typ uint8 před segmentací, což sice ulehčí segmentaci, ale přidává náročnost operací přetypování.

Dalším zajímavým výsledkem tohoto testu je rozdíl náročnosti mezi použitými metodami detekce. Za předpokladu, že jinak jsou algoritmy téměř shodné a vliv vyššího počtu detekovaných značek v případě Houghovy transformace (průměrně 10 %) má jen malý podíl na celkovém rozdílu, je průměrný čas potřebný pro provedení Houghovy transformace 40 ms. Jde tedy o relativně náročnou operaci.

Za zmínku u tohoto testu stojí nejlepší výsledek, kterého dosáhl algoritmus využívající YCbCr segmentaci se statickým prahem a detekci porovnáním. Výsledný čas – 64 ms – odpovídá snímkovací frekvenci videa 15 fps (frames per second), což je pro potřeby rozpoznání dopravních značek skvělý výsledek.

3.3.5. Test rozpoznání

Pro zhodnocení úspěšnosti algoritmu rozpoznání vnitřního piktogramu byla pro každý z algoritmů vypočtena hodnota procentuální úspěšnosti rozpoznání, kdy jako základní hodnota byl použit počet správně detekovaných značek. Opět byly použity hodnoty z testovací množiny den. Výsledky jsou na Obr. 3-28.



Obr. 3-28: Graf závislosti úspěšnosti rozpoznání z detekovaných značek na metodě segmentace

Z grafu je jasně patrné, že v případě, že je značka detekovaná, je úspěšnost algoritmu rozpoznání velmi vysoká, v některých případech dokonce rovna 99 %. Zajímavým výsledkem je, že hodnoty rozpoznání jsou vyšší u detekce porovnáním oproti Houghově transformaci. Také počet falešně rozpoznaných značek se drží na velmi nízké mezi pod 3 %. Zvolenou metodu rozpoznání lze označit za velmi spolehlivou.

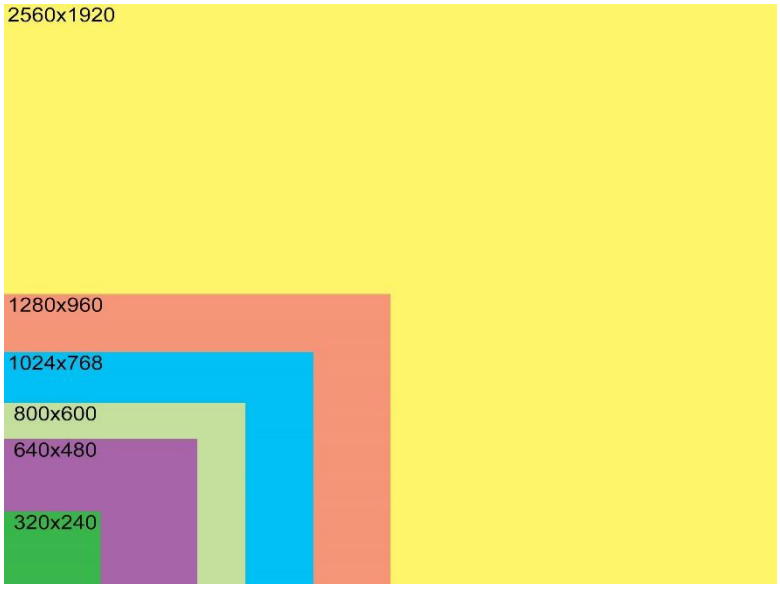
V rámci testu spolehlivosti rozpoznání byl ještě vyzkoušen vliv použití Otsuho metody oproti pevnému prahu při prahování oblasti vnitřního piktogramu. Při použití pevného prahu byla úspěšnost v průměru o 10 % nižší a na celkové výpočetní náročnosti nebylo patrné žádné výrazné zlepšení. Není tedy důvod nepoužít pro prahování Otsuho metodu.

3.3.5.1. Výpočetní náročnost v závislosti na rozlišení obrazu

Pro test výpočetní náročnosti v závislosti na rozlišení byla vytvořena množina 10 fotografií napříč různými značkami. Fotografie byly vybrány tak, aby měl algoritmus velkou pravděpodobnost určení značky, aby se nepromítl vliv obrázků bez detekovaných značek. Z této základní množiny pak byly vytvořeny množiny pro jednotlivé rozlišení zmenšením fotografií v programu Picasa. Rozlišení pro test byly zvoleny na základě standardních rozlišení pro formát fotografií a videa v poměru 4:3. Testovaná rozlišení lze najít v Tab. 3-4 a porovnání velikosti jednotlivých rozlišení na Obr. 3-29.

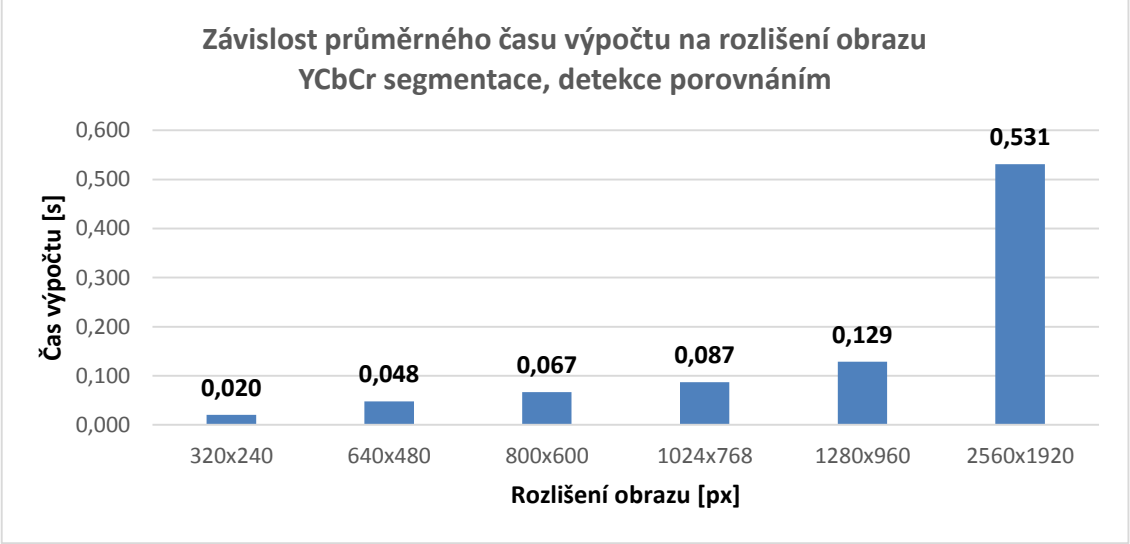
Zkratka	QVGA	VGA	SVGA	XGA	SXGA-	Originál
Šířka [px]	320	640	800	1 024	1 280	2 560
Výška [px]	240	482	600	768	960	1 920
Počet pixelů	76 800	307 200	480 000	786 432	1 228 800	4 915 200

Tab. 3-4: Seznam testovacích rozlišení



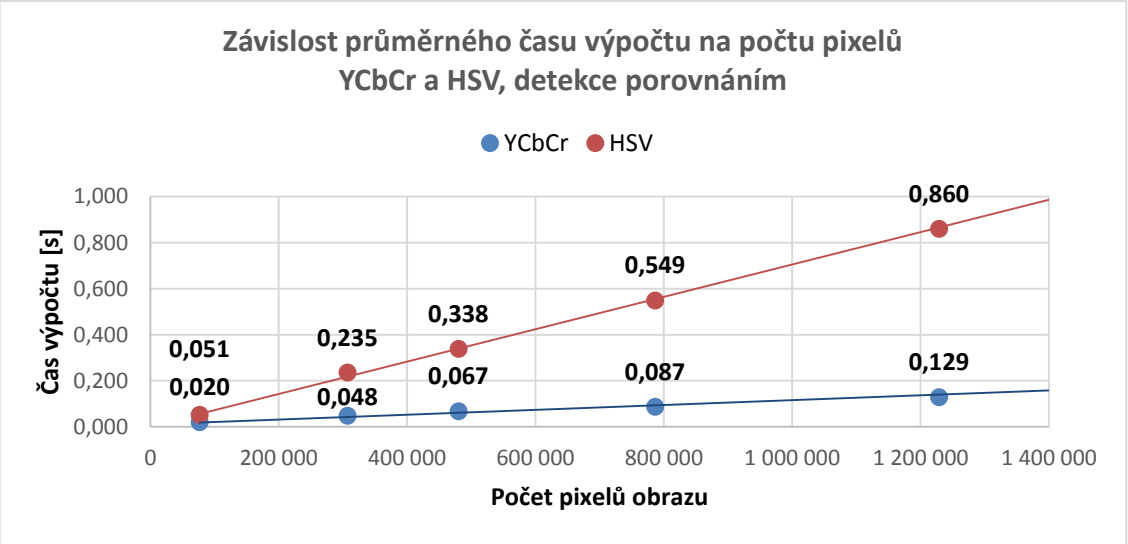
Obr. 3-29: Porovnání jednotlivých testovacích rozlišení

Pro tato rozlišení bylo opět provedeno testování na všechny kombinace metod segmentace a detekce. Jako ukázka výsledku je na Obr. 3-30 uvedena výpočetní náročnost YCbCr segmentace s detekcí porovnáním.



Obr. 3-30: Graf závislosti času výpočtu na rozlišení

Z grafu je patrné, že podle předpokladu rozlišení obrazu velmi ovlivňuje výpočetní náročnost. Pro lepší představu samotné závislosti byla vykreslena závislost výpočetního času na počtu pixelů. Pro porovnání byla přidána i HSV segmentace a výřez grafu omezen na zajímavou oblast, tedy počet pixelů maximálně odpovídající rozlišení 1280x960 pixelů. Výsledek lze vidět na Obr. 3-31



Obr. 3-31: Graf závislosti času výpočtu na celkovém počtu pixelů obrazu

Na tomto grafu je jasně zřetelné, že závislost na počtu pixelů je u obou typů segmentace lineární. Jen u HSV je sklon lineární charakteristiky 5krát strmější. Obecně pro potřeby rozpoznání dopravních značek v provozu bychom měli uvažovat o snímkovací frekvenci alespoň 10 fps, na základě této úvahy je tedy potřeba volit rozlišení, jež je algoritmus schopen provést pod 0,1 sekundy. V případě segmentace YCbCr pak jde konkrétně o obraz, jež obsahuje maximálně 1 000 000 pixelů.

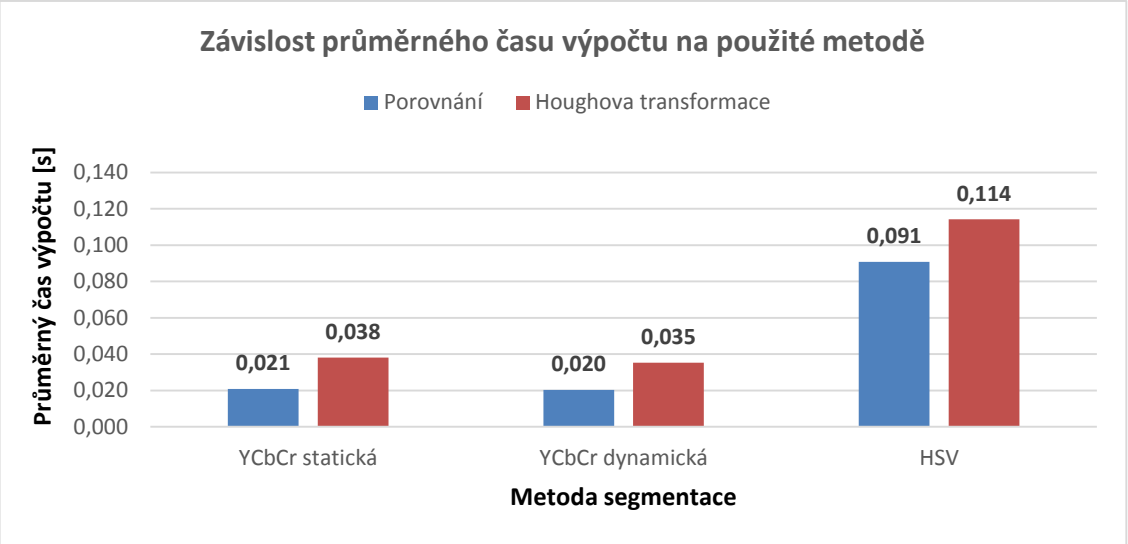
3.3.6. Rychlost výpočtu na výkonném počítači

Pro porovnání byly všechny testy provedeny ještě na výkonnějším počítači s parametry v Tab. 3-5.

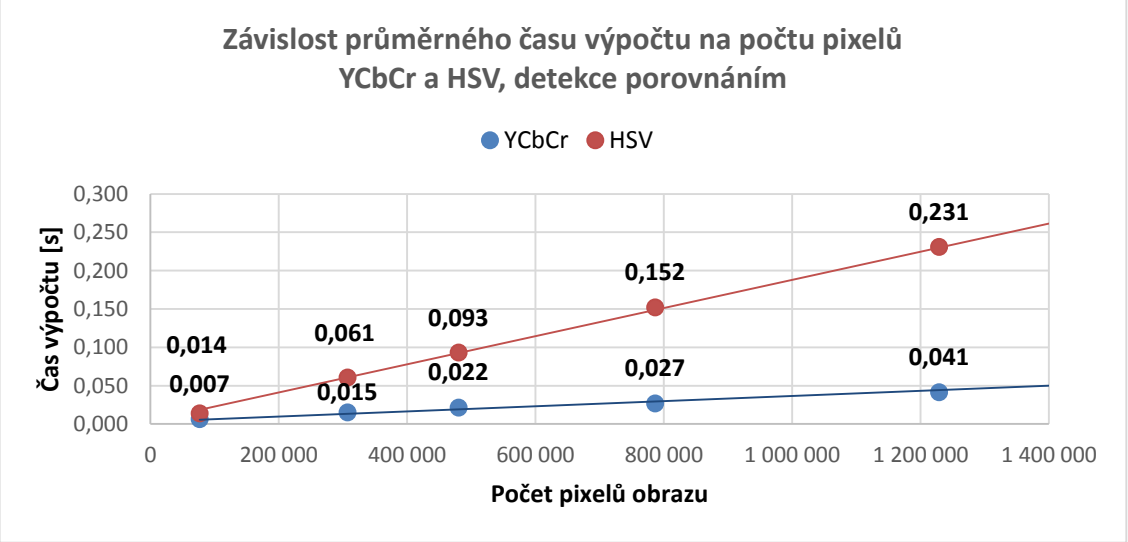
Výrobce	Lenovo
Model	ThinkPad T430 2344
Procesor	Intel® Core™ i7-3520M CPU
Frekvence	2.90 GHz
RAM	8 GB
Op. Systém	64 bit Windows 8.1 Pro
MATLAB	2013a

Tab. 3-5: Parametry výkonného zkušebního počítače

Samozřejmě na stejné testovací množině byly výsledky úspěšnosti algoritmů stejné. Relevantní změny jsou jen ve výpočetní rychlosti. Z tohoto důvodu byl vykreslen graf pro průměrnou rychlost při rozlišení 800x600 v závislosti na metodě (Obr. 3-32), který odpovídá Obr. 3-27; a dále graf závislosti času výpočtu na počtu pixelů (Obr. 3-33) odpovídající Obr. 3-31.



Obr. 3-32: Graf závislosti průměrného času výpočtu na použité metodě – výkonný počítač



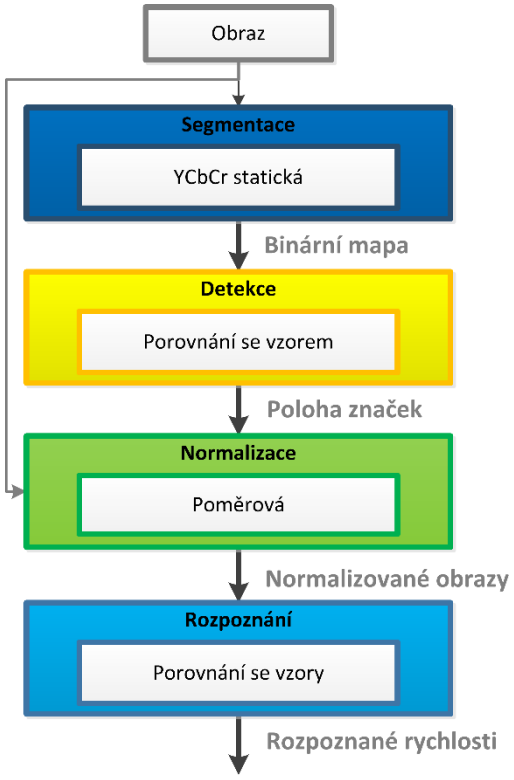
Obr. 3-33: Graf závislosti času výpočtu na celkovém počtu pixelů obrazu – výkonný počítač

Na základě těchto výsledků je patrné, že výkonný počítač dokáže provést vyvinuté algoritmy přibližně třikrát rychleji, než standardní přístroj. V případě, že by vývoj probíhal cíleně pro zařízení s takovým výkonem, bylo by možné aplikovat i složitější algoritmy detekce a rozpoznání. V současné podobě algoritmu je na počítači s vyšším výkonem doporučeno využít výpočetní výkon zvýšením použitého rozlišení, ale jen do té míry aby snímkovací frekvence neklesla pod 15 fps, případně ji nechat i na vyšší úrovni. Toto by mělo v celkovém měřítku zvýšit úspěšnost rozpoznání rychlosti v reálném provozu (viz kapitola 3.4).

3.3.7. Výběr výsledného algoritmu

Na základě všech provedených testů byl vybrán výsledný algoritmus pro implementaci do modelu v prostředí Simulink, který slouží k rozpoznání značek v rámci videa, a k použití ve výsledném prototypovém řešení.

Vlivem obecně špatných výsledků při omezených a umělých světelných podmínkách byla metoda volena na základě výsledků při dobrých podmínkách, tedy v rámci zkušební množiny den. Klíčovým při výběru finální metody byl faktor času, respektive výpočetní náročnosti. Z těchto důvodů byla vybrána metoda prahování založená na statickém prahu v Cr složce YCbCr barevného modelu a jako metoda detekce bylo zvoleno porovnání se vzorem kruhové značky. Schéma vybraného algoritmu je na Obr. 3-34.



Obr. 3-34: Blokové schéma vybraných metod

Metoda YCbCr se statickým prahem byla zvolena právě díky vynikajícím výsledkům, co se týče průměrného času výpočtu, a vysoké úrovni celkové úspěšnosti rozpoznání. Z pohledu úspěšnosti byla sice lepší metoda segmentace v HSV prostoru, ale vybraná metoda má pětinásobně menší výpočetní náročnost. Při reálném použití tedy dokážeme analyzovat 5 sekvenčních snímků touto vybranou metodou, zatímco HSV metoda by analyzovala jen jeden. Z tohoto důvodu je předpokládáno, že výsledná celková úspěšnost bude u vybrané metody vyšší. Stejně proběhla i úvaha u metody detekce, kdy

porovnání má také výrazně nižší výpočetní náročnost, při téměř stejné úspěšnosti detekce a dokonce vyšší úspěšnosti rozpoznání u takto detekovaných značek.

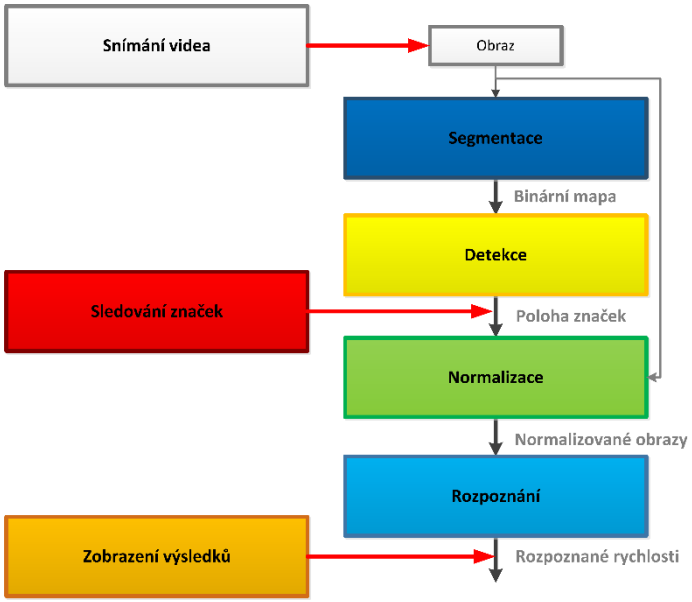
Co se týče výběru finálního rozlišení, pro standardní počítač bylo vybráno rozlišení 800x600 pixelů, které by mělo odpovídat snímkovací frekvenci 15 fps. Toto rozlišení by mělo být dostatečně detailní pro detekci značek standardně umístěných na okraji silnice. Ale celý algoritmus je vytvořen tak, aby fungoval se vstupními daty jakéhokoli rozlišení, je tedy možné jednoduše vstupní rozlišení upravit v rámci ladění výsledného řešení. V případě použití výkonného počítače bylo vybráno rozlišení 1280x960 pixelů.

3.4. Implementace řešení pro data ve formě videa

Rozpoznávat dopravní značky omezující rychlost ve videu (ať už záznamu, nebo snímaném v reálném čase) a celkově tvorba uživatelsky funkčního řešení s sebou přináší několik dalších problémů, které musí být vyřešeny. Nabízí ale také jednu výhodu, oproti rozpoznání na jednotlivých fotografiích. Mezi problémy patří např. samotný proces snímání videa a jeho softwarové řešení, nebo naopak výstup algoritmu, neboli upozornění řidiče na rozpoznané značky.

Výhodou, o které byla řeč, je fakt, že v podstatě získáváme dimenzi pro rozpoznání navíc. Lze předpokládat, že každou značku v reálném provozu sejmeme v rámci videosekvence několikrát. Už tedy nejde jen o rozpoznání na jedné fotografii s výsledkem rozpoznáno/nerozpoznáno, ale v případě neúspěchu na jednom snímku je stále možnost rozpoznat značku na snímcích dalších. Případně je možné využít více snímků pro zvýšení jistoty ohledně správného rozpoznání. Tato výhoda se samozřejmě týká především úspěšnosti rozpoznání. Co se týče vytváření algoritmu, znamená řešení dalších problémů, především sledování polohy značky mezi snímky a vytvoření a implementace celkové logiky zvýšení úspěšnosti rozpoznání.

Pro potřeby rozpoznání dopravních značek ve videu bylo potřeba přidat do architektury algoritmu několik dalších bloků (viz Obr. 3-34), které slouží k řešení výše zmíněných problémů, a také několik logických signálů mezi jednotlivými bloky (kapitola 3.4.2).



Obr. 3-35: Přidané bloky algoritmu pro rozpoznání ve videu

Při vytváření modelu v Simulinku byl algoritmus segmentace sestaven z bloků pro jednotlivé operace. Ostatní částí algoritmu (detekce, normalizace a rozpoznání) byly implementovány v podobě kódu vloženého do bloku *MATLAB Function*, jelikož jde o složitější funkce a sestavovat je přímo v Simulinku by nebylo příliš efektivní. Takto stačilo již vyvinutý kód pro jednotlivé metody zkopírovat a případně jen lehce upravit.

Většina signálů mezi bloky je vícerozměrná, a to nejen z důvodu informace, kterou přenášejí (např. poloha značek – přenáší 4 rozměry bounding boxu), ale i proto, že algoritmus je dimenzován na více značek v jednom snímku. Konkrétní maximální počet značek je nastaven na 6, počínaje blokem sledování značek mají tedy všechny signály v nejvyšší dimenzi 6 hodnot. Mezi bloky detekce a sledování značek je pak prostor pro 10 různých detekovaných značek (správně nebo falešně).

Blok snímání videa, případně získání videa ze záznamu, je vyřešen přímo pomocí příslušných bloků v Simulinku (*From Video Device* a *From Multimedia File*), nebylo tedy potřeba se tímto problémem více zabývat. Tyto bloky nabízí spoustu možnosti nastavení, ať už jde o nastavení snímkovací frekvence, nebo datového typu výstupu. Ostatní nově přidané algoritmy jsou popsány dále.

3.4.1. Algoritmus sledování značek mezi snímky

Pro sledování značek mezi snímky byl použit algoritmus založený na distanční matici, který byl částečně převzat z [21]. Tento algoritmus pracuje tak, že ke každému vstupu (bounding box detekované značky v současném snímku) vypočte hodnotu distance od všech bounding boxů, které byly výstupem v minulém běhu funkce, podle rovnice 1.5.

$$d = \frac{|2\times(y_{k-1}-y_k)+v_{k-1}-v_k|}{v_{k-1}-v_k} + \frac{|2\times(x_{k-1}-x_k)+s_{k-1}-s_k|}{s_{k-1}-s_k}, \qquad (1.5)$$

kde d hodnota distance

y y souřadnice levého horního rohu,

x x souřadnice levého horního rohu,

v výška bounding boxu [px] ,

s šířka bounding boxu [px],

k vyjadřuje hodnoty proměnných v současném snímku,

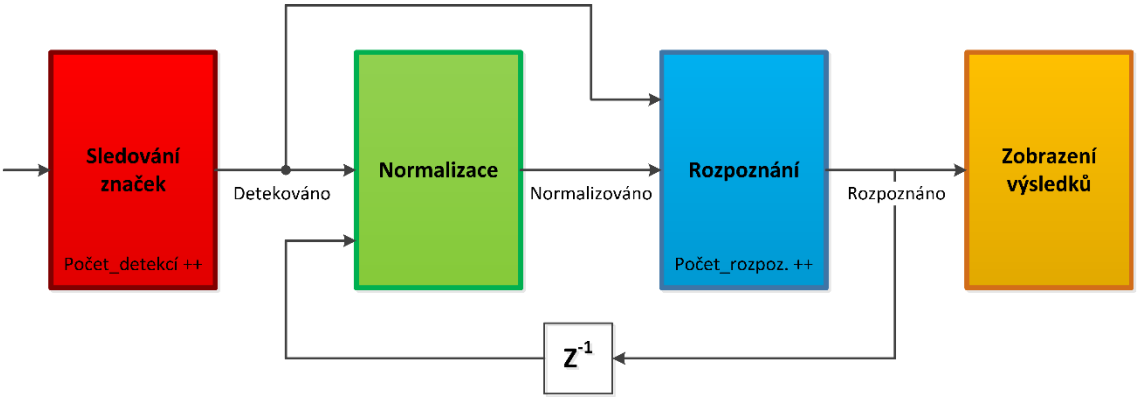
$k-1$ vyjadřuje hodnoty proměnných v minulém snímku.

Následně vždy pro danou detekovanou značku vyhledá nejmenší hodnotu distance od jednotlivých značek detekovaných v minulém snímku. Pokud je hodnota menší než maximální mez distance, prohlásí nově detekovanou značku za odpovídající značce v minulém snímku a dosadí nové hodnoty na stejné místo na výstupu, přitom se zvýší čítač počítající počet konsekutivních detekcí stejné značky. Dosazení na stejné místo na výstupu se dále propaguje všemi bloky a je klíčové pro správnou funkci dalších algoritmů. Pokud se žádná hodnota distance nedostane pod mezní hodnotu, je značka prohlášena za nově detekovanou značku a je ji přiděleno prázdné místo na výstupu. V rámci bloku sledování pak ještě dochází k porovnání čítače s konstantní hodnotou 2, která vyjadřuje, že až po dvou po sobě následujících detekcích je značka prostřednictvím logického signálu označena za detekovanou. Toto opatření slouží k odfiltrování falešných detekcí.

Tento víceméně jednoduchý postup sledování objektů v jednotlivých snímcích videa se ukázal jako dostatečně účinný pro potřeby sledování značek.

3.4.2. Logický algoritmus pro zvýšení úspěšnosti a snížení náročnosti

Díky implementaci algoritmu sledování značky v konsekutivních snímcích videa bylo možné využít výhody videa, které byly zmíněny výše. Z tohoto důvodu bylo zavedeno několik logických signálů mezi bloky sledování, normalizace, rozpoznání a zobrazení a také bylo nutné implementovat čítače detekcí a rozpoznání jednotlivých značek. Propojení jednotlivých bloků těmito signály je možné vidět na Obr. 3-36.



Obr. 3-36: Schéma propojení bloků logickými signály

Vyšší úspěšnost rozpoznání by měla být zajištěna dvěma čítači. První je ve fázi sledování a jeho funkce už byla popsána výše. Druhý je implementován v bloku rozpoznání, a jeho hodnota se zvyšuje v případě, že je stejná značka rozpoznána stejně. Pokud se uskuteční dvě stejná rozpoznání po sobě, je značka prohlášena za rozpoznanou a na výstupu je logická 1. Výjimkou je případ, pokud je značce přiřazen kód 250 (jiná zákazová značka); v tomto případě musí dojít k „nerozpoznání“ 8 krát, aby měl algoritmus větší šanci upřesnit rozpoznání.

Logika algoritmu je založena na pravidle, aby se žádná část neprováděla zbytečně, čím by měla být snížena celková výpočetní náročnost. Proto blok normalizace probíhá pouze v případě, že je značka detekována, ale ještě není rozpoznána a tento výsledek se přenáší i do bloku rozpoznání. Ten po splnění podmínky rozpoznání už jen propaguje dále již rozpoznanou hodnotu, dokud je stále sledována stejná značka, což je dáno jedničkou na logickém signálu *detekováno*. Jakmile signál spadne na 0, všechny hodnoty se v podstatě resetují a může se začít s rozpoznáváním další značky.

3.4.3. Zobrazení výsledků

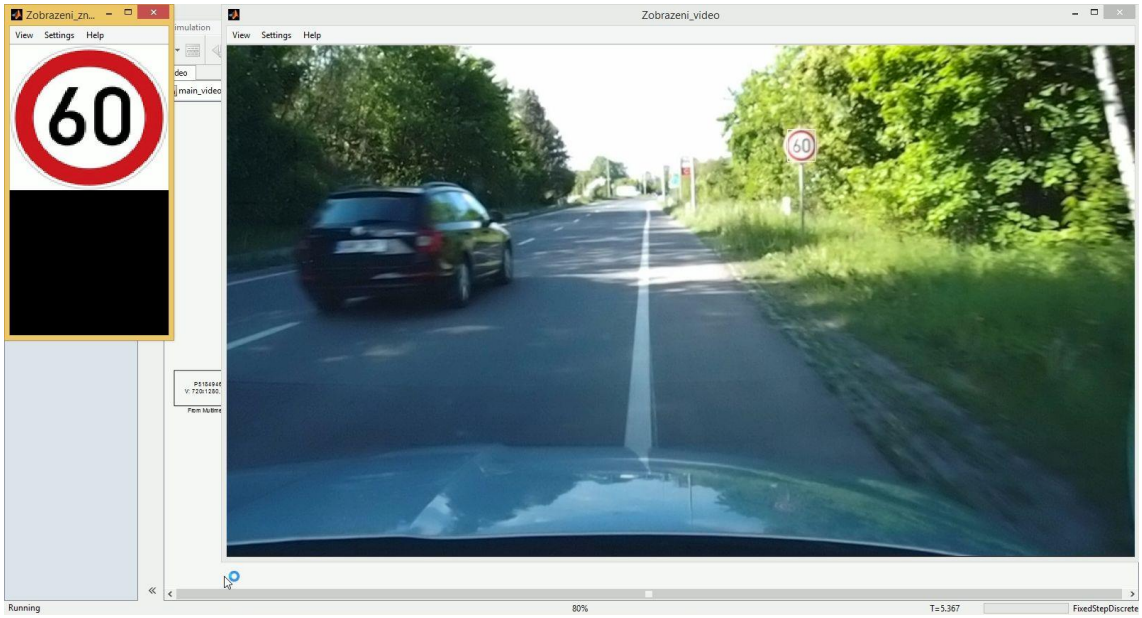
Pro finální zobrazení výsledků algoritmu rozpoznání byly zvažovány různé možnosti. Nakonec bylo vybráno řešení, které se řídí zásadami jednoduchosti a funkčnosti. Pro reálné použití v provozu není důležité zobrazovat co nejvíce informací, které z algoritmu dostáváme, toto je důležité jen pro testování, debuggování a ladění programu. Pro řidiče automobilu s tímto asistenčním systémem je důležitá jen jedna informace a to konkrétní značka, kterou systém rozpoznal a které si řidič například nevšiml. Takto bývá řešen tento systém i v komerčních řešeních použitých u současných automobilů.

Podobný algoritmus byl implementován i v tomto případě, kdy na základě výstupu z bloku rozpoznání (rozpoznaná rychlost a status rozpoznání) algoritmus zobrazí ve výstupním okně barevný vzor detekované značky. Značky jsou přitom řazeny do dvou kategorií: značky omezení rychlosti a ostatní značky. Jelikož tato práce byla zaměřena

především na značky omezení rychlosti, a také proto, že jsou tyto značky mnohdy pro řidiče důležitější, zobrazují se ve výstupním okně nahoře. Ostatní rozpoznané značky se nezávisle na nich zobrazují pod nimi. U všech značek je implementována jistá setrvačnost zobrazení, kdy pokud není rozpoznána jiná značka, zobrazuje se rozpoznaná značka ještě nějaký čas poté, co zmizela ze zorného pole kamery a už není detekována. Toto by mělo zajistit právě upozornění řidiče na značku, které si nevšiml. Barevné vzory, jež algoritmus zobrazuje, a tedy i značky, které systém rozpoznává lze vidět na Obr. 3-37.

Obr. 3-37: Vzory značek k zobrazení

Jako ukázka funkčnosti algoritmu bylo implementováno ještě jednoduché zobrazení detekce značky přímo ve videu: v okně videa se přímo do obrazu z kamery vykreslují obdélníky kolem detekovaných značek. Toto bylo provedeno použitím dedikovaného bloku CVS Toolboxu *Draw Shapes*, na který byl přiveden signál z bloku sledování. Ukázka výsledného zobrazení je na Obr. 3-38.



Obr. 3-38: Výsledné zobrazení výsledků

Za zmínku ještě stojí bloky použité pro samotné zobrazení. Tím jsou opět bloky z CSV Toolboxu *To Video Display*, jež slouží k jednoduchému zobrazení videa, ale navíc podporují hardwarovou akceleraci a generování kódu.

3.4.4. Optimalizace programu

Po vytvoření celkového modelu v prostředí Simulink byla snaha program co nejvíce optimalizovat pro snížení výpočetní náročnosti. To bylo provedeno jak standardními úpravami algoritmu, tak použitím zabudovaného nástroje Profiler v prostředí Simulink.

Co se týče první z úprav, jako příklad může být uvedeno opětovné zkontrolování a úprava datových typů, které především u práce s obrazem hrají velkou roli ve výsledné výpočetní náročnosti. Celkově byla snaha používat pro daný účel datové typy s minimální velikostí. Hlavní použité datové typy jsou následující:

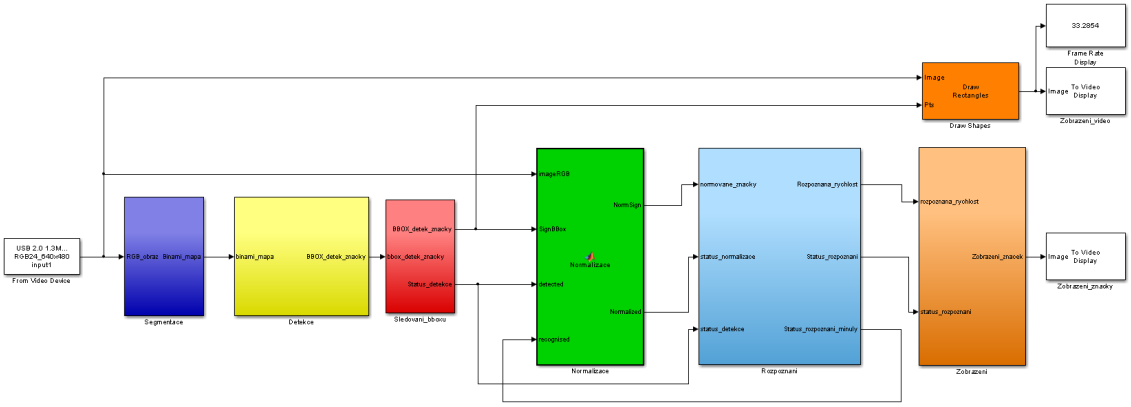
- uint8 pro veškeré obrazové data
- uint16 pro souřadnice v obrazu (bounding box)
- boolean pro logické signály a binární mapy obrazu

Výhodou práce v Simulinku je možnost zobrazit si datové typy u všech signálů, čímž se kontrola velmi zjednoduší, ale je potřeba dodržovat tyto zásady i v rámci implementovaných funkcí. V případě, že byl změněn datový typ na vstupu funkce, bylo dále potřeba toto přenastavit i v nástroji Ports and Data Manager, jinak Simulink při kompilací hlásil chybu.

Pro hledání problematických míst algoritmu byl použit nástroj Profiler. Pokud je tento nástroj aktivován, vytváří po každém běhu modelu zprávu ohledně výkonu jednotlivých částí algoritmu (funkce, bloky), což je velmi užitečné pro diagnostiku programu a hledání problematických míst. Tato zpráva obsahuje následující informace:

- časové a procentuální vyjádření celkového času provádění funkce,
- počet volání funkce,
- čas jednoho provedení,
- tzv. Self time, tedy čas „strávený“ přímo ve funkci, bez času provádění použitých subfunkcí.

Jako příklad problému nalezeného prostřednictvím tohoto nástroje, lze uvést zjištění výrazné výpočetní náročnosti bloku *Relational Operator*, který porovnával Cr složku obrazu s konstantním prahem při prahování obrazu. Na základě tohoto zjištění byl blok nahrazen blokem *Compare to Constant* a náročnost operace se snížila řádově. Později bylo zjištěno, že hlavní problém spočíval ve špatném nastavení datových typů a přetypování konstanty v každém běhu funkce.



Obr. 3-39: Náhled výsledného modelu v prostředí Simulink, detailní pohled obsahuje Příloha 1

3.4.5. Zhodnocení výsledků rozpoznání značek ve videu

Pro zhodnocení výsledků jsou klíčové dva parametry, stejně jako při jakémkoli předchozím testování: rychlost výpočtu a celková úspěšnost rozpoznání. Rychlost výpočtu byla tentokrát měřena přímo v počtu snímků, které je algoritmus schopný zpracovat za sekundu, tedy v hodnotě fps. K tomu slouží blok *Frame Rate*.

Při použití testovacího videa s rozlišením 1280x720 pixelů se snímkovací frekvence na výkonném počítači pohybovala okolo hodnoty 30 fps, což je celkově vynikající výsledek. Co se týče úspěšnosti rozpoznání, algoritmus rozpoznal většinu rychlostních značek v záznamu, úspěšnost lze hodnotit jako vysokou. Jako problematická se ukázala detekce značek v protisvětle, kdy dochází k velkému zkreslení červené barvy. Ve výsledku jde ale o funkční řešení, jež rozpoznává dopravní značky omezující

povolenou rychlost. Záznam obrazovky při rozpoznávání v testovacím videu lze vidět na Obr. 3-38 a další příklady obsahuje Příloha 2.

3.5. Test prototypového zařízení v provozu

Vzhledem ke stanoveným cílům této práce byl výsledný algoritmus vyzkoušen v reálném provozu.

3.5.1. Popis použitého hardware

Díky univerzalitě algoritmu, co se týče možností použitého zařízení, nebyla výběru hardware pro prototypové zkoušky v provozu věnovaná příliš velká pozornost. Obecně lze říci, že výsledný program může fungovat na jakémkoli počítači s dostatečným výpočetním výkonem a nainstalovaným programem MATLAB s použitými toolboxy. A dále je už potřeba jen mít připojenou kameru s dostatečným rozlišením a reálnou reprezentací barev.

Pro testování v provozu byla použita tato kombinace, kterou je možné vidět i na Obr. 3-40:

- Počítač Lenovo ThinkPad T430 2344 (specifikace viz Tab. 3-1)
- Kamera Logitech HD Webcam C270 (specifikace viz [22])

Obr. 3-40: Hardware použitý pro testování v provozu

Jelikož byl použit výkonný počítač, bylo tomu přizpůsobeno i nastavení rozlišení kamery, kdy bylo použito rozlišení 1280x960 pixelů. Kamera byla umístěna ihned za čelním sklem na straně spolujezdce a namířena přímo ve směru jízdy.

3.5.2. Zhodnocení výsledků

Při praktické zkoušce výsledného programu v reálném provozu bylo dosaženo výborných výsledků. Testovací jízda trvala cca 30 minut a probíhala v městském prostředí. Rychlost automobilu se pohybovala nejčastěji kolem 50 km/h v jednom úseku dosáhla 80 km/h. Během této jízdy byly rozpoznány všechny značky, které algoritmus rozpoznává a které byly při jízdě míjeny. Většina detekcí probíhala ve vzdálenosti cca 15 metrů od značky. Některé značky ve větší vzdálenosti (např. za křižovatkou) nebyly detekovány, ale to bylo způsobeno použitou kamerou, která má kratší ohniskovou vzdálenost a objekty ve větší vzdálenosti byly barevně zkresleny.

Mezi detekovanými byly i další zákazové značky, algoritmus je správně vyhodnotil jako jiné zákazové značky a nebyly zobrazeny na výstupu. Během jízdy byla spatřena jen jedna falešná detekce, kdy šlo o červené kulaté logo společnosti.

Celkově se snímkovací frekvence pohybovala kolem 20 fps, což se ukázalo jako dostatečná frekvence pro správnou detekci a rozpoznání a také pro sledování značek i při značných rázových vibracích kamery při jízdě.

Na základě této testovací jízdy lze hodnotit výsledný program jako funkční řešení problému detekce a rozpoznání omezení rychlosti z dopravních značek.

Obr. 3-41: Ukázka rozpoznaných značek při testu v provozu

4. ZÁVĚR

Cílem diplomové práce bylo vyvinout algoritmus na detekci a rozpoznání omezení rychlosti z dopravních značek.

V úvodní části práce je uvedena rešerše současného stavu této problematiky. Proces rozpoznání dopravních značek je rozdělen do několika bloků a ke každému z těchto bloků jsou uvedeny nejčastěji používané metody s odkazy na literaturu. Na závěr rešeršní části jsou popsány tři práce, které jsou relevantní především svým zaměřením a rozsahem. Následuje ještě krátké pojednání o specifikách řešení tohoto problému v prostředí MATLAB.

Popis řešení stanoveného problému začíná definicí specifik práce. Následuje přehled všech metod, které byly implementovány a otestovány. V rámci segmentace obrazu byly vyzkoušeny dvě metody založeny na YCbCr barevném modelu, metoda se statickým a metoda s dynamickým prahem, a jedna metoda segmentace na základě HSV barevného modelu. Pro detekci kruhové značky bylo naprogramováno porovnání se vzorem a detekce použitím Houghovy transformace pro kruhy. Detekované značky bylo potřeba transformovat na normalizovanou velikost 50x50 pixelů, čehož bylo dosaženo třemi různými metodami: proporcionální transformace, normalizace podle nalezené elipsy podle Čípa a perspektivní transformace na základě vytvořeného nalezení klíčových bodů. Další dvě metody nepřinesly očekávané výsledky, a proto nebyly dále použity. Finální krok rozpoznání rychlosti byl proveden porovnáním se vzorem vnitřního piktogramu.

Pro zhodnocení metod je popsáno rozsáhlé testování na reálných fotografiích z provozu, které bylo provedeno prostřednictvím vytvořeného skriptu. Testy byly vyhodnoceny z hlediska spolehlivosti detekce a výpočetní náročnosti, při uvažování vlivu světelných podmínek a rozlišení fotografií. Na základě těchto testů byla vybrána metoda pro implementaci do prototypového zařízení: segmentace YCbCr se statickým prahem, detekce porovnáním, proporcionální normalizace a rozpoznání porovnáním vnitřního piktogramu se vzorem.

Pro účely rozpoznání značek ve videosekvenci je dále popsán algoritmus sledování značek mezi snímky a využití tohoto sledování pro zlepšení úspěšnosti. Poslední částí algoritmu je zobrazení výsledků – upozornění řidiče na rozpoznané dopravní značky.

Finální řešení bylo otestováno nejdříve na videozáznamu a následně v reálném provozu. Ukázalo se, že výsledný algoritmus je funkčním řešením detekce a rozpoznání dopravních značek omezujících rychlost (a několika dalších). Všechny cíle práce byly tedy splněny.

5. POUŽITÁ LITERATURA

[1] AZAD, Reza. View-Independent Traffic Speed Sign Detection and Recognition System for Intelligent Driving System. In: *International Journal of Modern Education and Computer Science* [online]. 2014, s. 31-37 [cit. 2015-05-19]. DOI: 10.5815/ijmecs.2014.03.04. ISSN 20750161. Dostupné také z: http://www.mecs-press.org/ijmecs/ijmecs-v6-n3/v6n3-4.html

[2] DVOŘÁK, Marek. *Aplikace pro rozpoznání dopravních značek pro Windows Phone 7*. Brno, 2013. Bakalářská práce.

[3] FISHER, Robert, Simon PERKINS, Ashley WALKER a Erik WOLFART. Digital Filters. *The University of Edinburg, School of Informatics* [online]. 2003 [cit. 2015-05-19]. Dostupné také z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm

[4] ČÍP, Pavel. *Detekce a rozpoznávání dopravních značek*. Brno, 2009. Dostupné také z: https://www.vutbr.cz/studium/zaverecne-prace?zp_id=22225. Diplomová práce.

[5] ZAKIR, U., A.N.J. LEONCE a E.A. EDIRISINGHE Road Sign Segmentation Based on Colour Spaces: A Comparative Study. In: *Proceedings of the 11th IASTED International Conference Computer Graphics and Imaging (CGIM 2010)*. 2010. Dostupné také z: https://www.academia.edu/388864/Road_Sign_Segmentation_Based_on_Colour_Spaces_A_Comparative_Study

[6] HSL and HSV. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2015-05-19]. Dostupné také z: http://en.wikipedia.org/wiki/HSL_and_HSV

[7] CANNY, John. A Computational Approach to Edge Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 1986, s. 679-698 [cit. 2015-05-19]. DOI: 10.1109/TPAMI.1986.4767851. ISSN 0162-8828. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4767851

[8] HART, Peter. How the Hough transform was invented [DSP History. In: *IEEE Signal Processing Magazine* [online]. 2009, s. 18-22 [cit. 2015-05-19]. DOI: 10.1109/MSP.2009.934181. ISSN 1053-5888. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5230799

[9] FISHER, Robert, Simon PERKINS, Ashley WALKER a Erik WOLFART. Hough Transform. *The University of Edinburg, School of Informatics* [online]. 2003 [cit. 2015-05-19]. Dostupné také z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[10] *GPU gems 3* [online]. Upper Saddle River: Addison-Wesley, 2008, l, 942 s. [cit. 2015-05-19]. ISBN 978-0-321-51526-1. Dostupné také z: https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_pref01.html

[11] BAHLMANN, C., Y. ZHU, V. RAMESH, M. PELLKOFER a T. KOEHLER A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005* [online]. IEEE, 2005, s. 255-260 [cit. 2015-05-19]. DOI: 10.1109/IVS.2005.1505111. ISBN 0-7803-8961-1. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1505111

[12] GAO, X.W., L. PODLADCHIKOVA, D. SHAPOSHNIKOV, K. HONG a N. SHEVTSOVA Recognition of traffic signs based on their colour and shape features extracted using human vision models. In: *Journal of Visual Communication and Image Representation* [online]. 2006, s. 675-685 [cit. 2015-05-19]. DOI: 10.1016/j.jvcir.2005.10.003. ISSN 10473203. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/S1047320305000994

[13] BENEŠ, Radek. *Využití metod zpracování signálů pro zvýšení bezpečnosti automobilové dopravy*. Brno, 2009. Dostupné také z: https://www.vutbr.cz/studium/zaverecne-prace?zp_id=21737. Diplomová práce.

[14] THE MATHWORKS, INC. Computer Vision System Toolbox. *MathWorks* [online]. 1994 [cit. 2015-05-19]. Dostupné také z: http://www.mathworks.com/products/computer-vision

[15] THE MATHWORKS, INC. Expressing Image Locations. *MathWorks* [online]. 1994 [cit. 2015-05-19]. Dostupné také z: http://www.mathworks.com/help/images/image-coordinate-systems.html

[16] CHEN, Yixin, Yi XIE a Yulin WANG. Detection and Recognition of Traffic Signs Based on HSV Vision Model and Shape features. In: *Journal of Computers* [online]. 2013, s. - [cit. 2015-05-19]. DOI: 10.4304/jcp.8.5.1366-1370. ISSN 1796-203x. Dostupné také z: http://ojs.academypublisher.com/index.php/jcp/article/view/9570

[17] THE MATHWORKS, INC. Documentation: All Products. *MathWorks* [online]. 1994 [cit. 2015-05-19]. Dostupné také z: http://www.mathworks.com/help/index.html

[18] Understanding image-interpolation techniques. *Vision Systems Design* [online]. 2007 [cit. 2015-05-19]. Dostupné také z: http://www.vision-systems.com/articles/print/volume-12/issue-10/departments/wilsons-websites/understanding-image-interpolation-techniques.html

[19] OTSU, Nobuyuki. A Threshold Selection Method from Gray-Level Histograms. In: *IEEE Transactions on Systems, Man, and Cybernetics* [online]. 1979, s. 62-66 [cit. 2015-05-19]. DOI: 10.1109/TSMC.1979.4310076. ISSN 0018-9472.

Dostupné také z:
http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4310076

[20] MEGAPIXEL S.R.O. Recenze Olympus XZ-1. *MEGAPIXEL* [online]. 2001 [cit. 2015-05-19]. Dostupné také z: http://www.megapixel.cz/olympus-xz-1/recenze

[21] THE MATHWORKS, INC. Traffic Warning Sign Recognition. *MathWorks* [online]. 1994 [cit. 2015-05-19]. Dostupné také z:
http://www.mathworks.com/help/vision/examples/traffic-warning-sign-recognition.html

[22] LOGITECH. Logitech HD Webcam C270. *Logitech* [online]. 2015 [cit. 2015-05-19]. Dostupné také z: http://www.logitech.com/en-hk/product/hd-webcam-c270

6. SEZNAM SYMBOLŮ A ZKRATEK

ADAS	Advanced Driver Assistance Systems, pokročilé asistenční systémy
RGB	Red, Green, Blue, zkratka barevného modelu
YCbCr	Luminance, modrý a červený chrominanční komponent, zkratka barevného modelu
Cr	červený chrominanční komponent v YCbCr modelu
HSV	Hue, Saturation, Value, zkratka barevného modelu
CIELab	zkratka barevného modelu definovaného Mezinárodní komisí pro osvětlování (Commission internationale de l'éclairage)
FOSTS	Foveal System for Traffic Signs, metoda rozpoznání dopravních značek
C++	programovací jazyk
OpenCV	Open Source Computer Vision, knihovna pro práci s obrazem
SURF	Speeded Up Robust Features, metoda počítačového vidění
CVS	Computer Vision System, název MATLAB Toolboxu
uint8	8bitové neznaménkové celé číslo, datový typ
uint16	16bitové neznaménkové celé číslo, datový typ
boolean	logický (1bitový) datový typ
double	64bitový datový typ s plovoucí čárkou
s	koeficient podobnosti
T	matice vyjadřující vzor
I	matice vyjadřující obraz
Σ	vyjadřuje součet všech prvků matice
x_S	x-ová souřadnice středu
x_P, y_P	souřadnice pravého krajního bodu
x_L, y_L	souřadnice levého krajního bodu
y_{max}	počet pixelů obrazu ve směru y
jpg	Joint Photographic Experts Group, zkratka obrazového formátu
“	palec, jednotka délky
Px	pixel
MPx	megapixel

CCD	Charge-coupled Device, typ snímače obrazu
GHz	gigahertz, jednotka frekvence
GB	gigabyte, jednotka množství dat
s	sekunda, jednotka času
ms	milisekunda, jednotka času
fps	frames per second (snímky za sekundu), jednotka snímkovací frekvence
d	hodnota distance
y	y souřadnice levého horního rohu
x	x souřadnice levého horního rohu
v	výška bounding boxu [px]
s	šířka bounding boxu [px]
k	vyjadřuje hodnoty proměnných v současném snímku
$k-1$	vyjadřuje hodnoty proměnných v minulém snímku
km/h	kilometr za hodinu, jednotka rychlosti

7. SEZNAM OBRÁZKŮ, GRAFŮ A TABULEK

8. SEZNAM PŘÍLOH

Příloha 1: Výsledný model v prostředí Simulink

Příloha 2: Záznam obrazovky z testování na videozáznamu

Příloha 3: Fotografie z testování v provozu

Příloha 4: Ukázka kódu - funkce normalizace

Příloha 5: Výsledky testu detekce a rozpoznání – standardní počítač

Příloha 6: Výsledky testu rozlišení – standardní počítač

Příloha 7: Výsledky testu detekce a rozpoznání – výkonný počítač

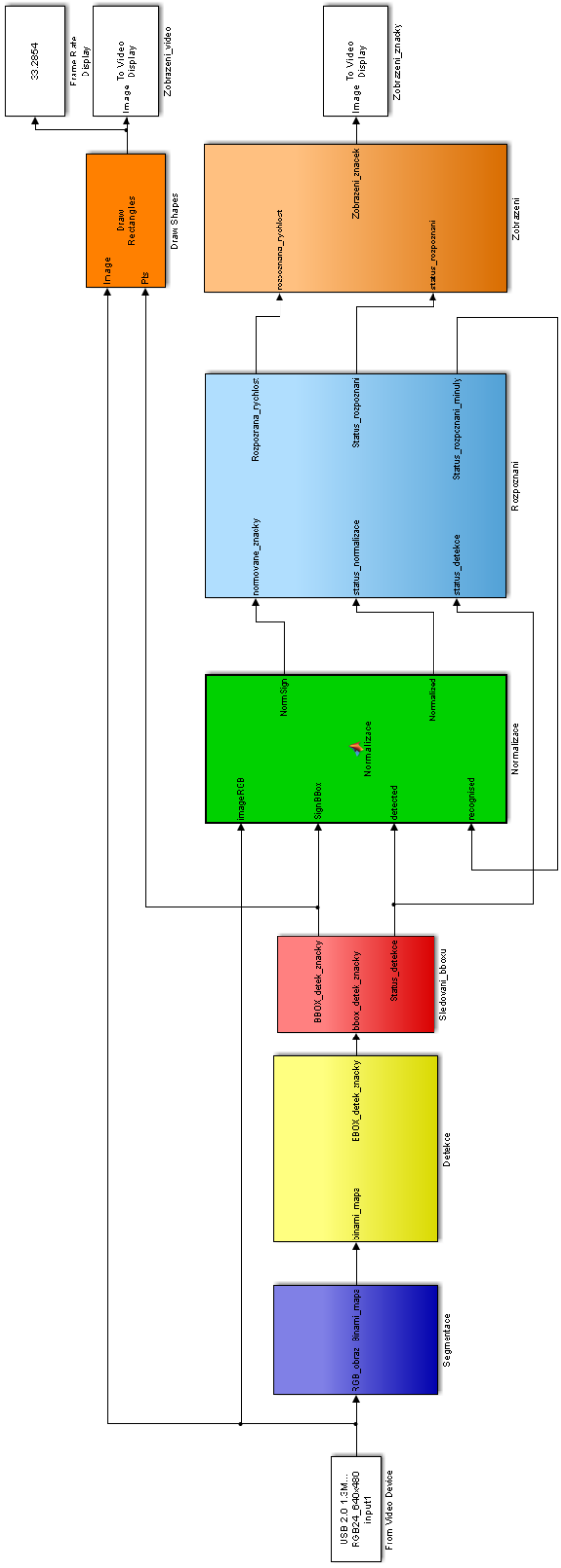
Příloha 8: Výsledky testu rozlišení – výkonný počítač

ELEKTRONICKÉ PŘÍLOHY

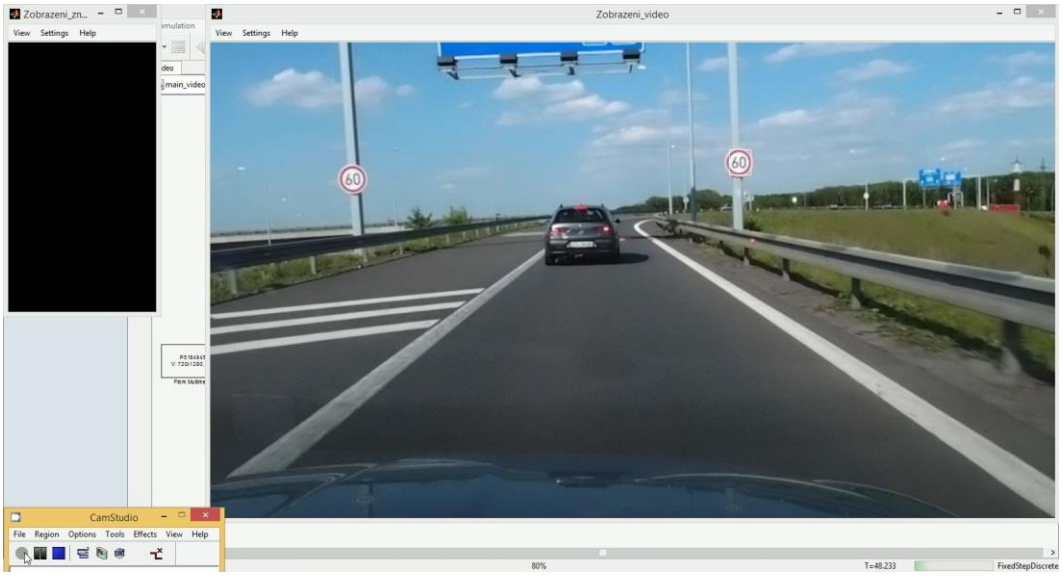
- Model pro real-time rozpoznání značek
- Model pro rozpoznání značek ve videu
- Testovací skript včetně všech funkcí
- Výsledky testování ve formátu xlsx
- Ukázkové video

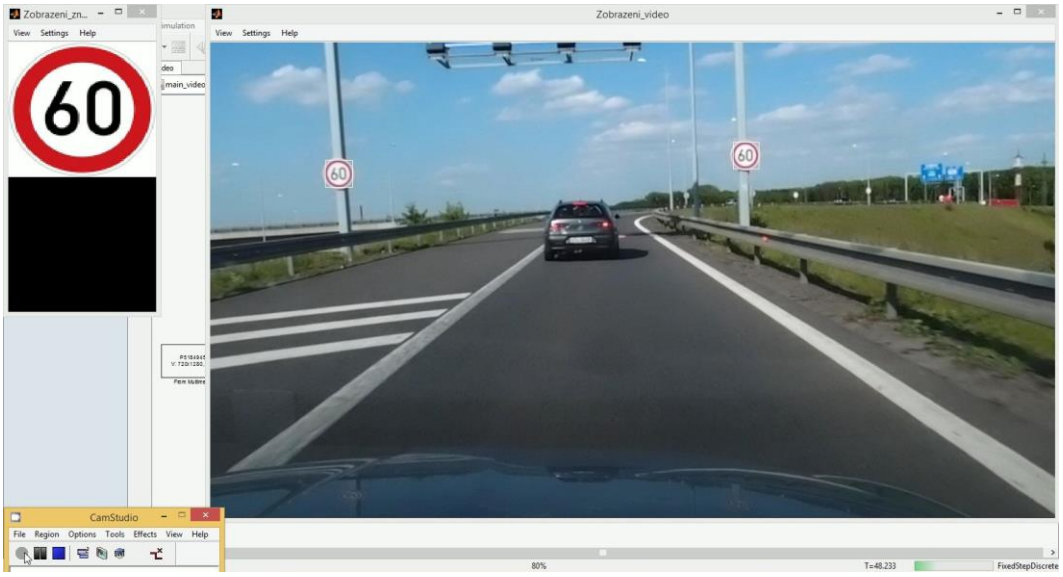
PŘÍLOHY

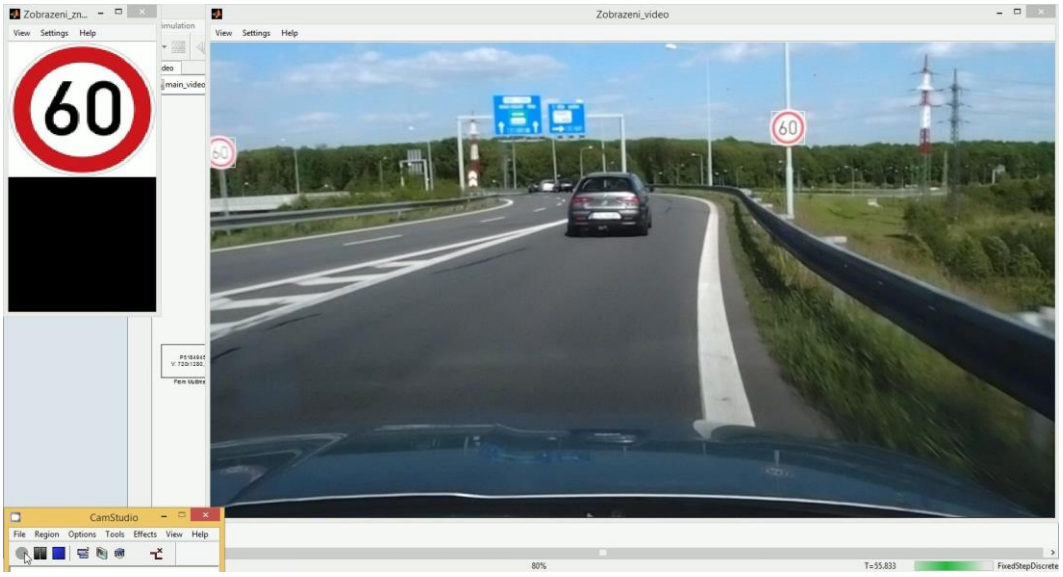
Příloha 1: Výsledný model v prostředí Simulink



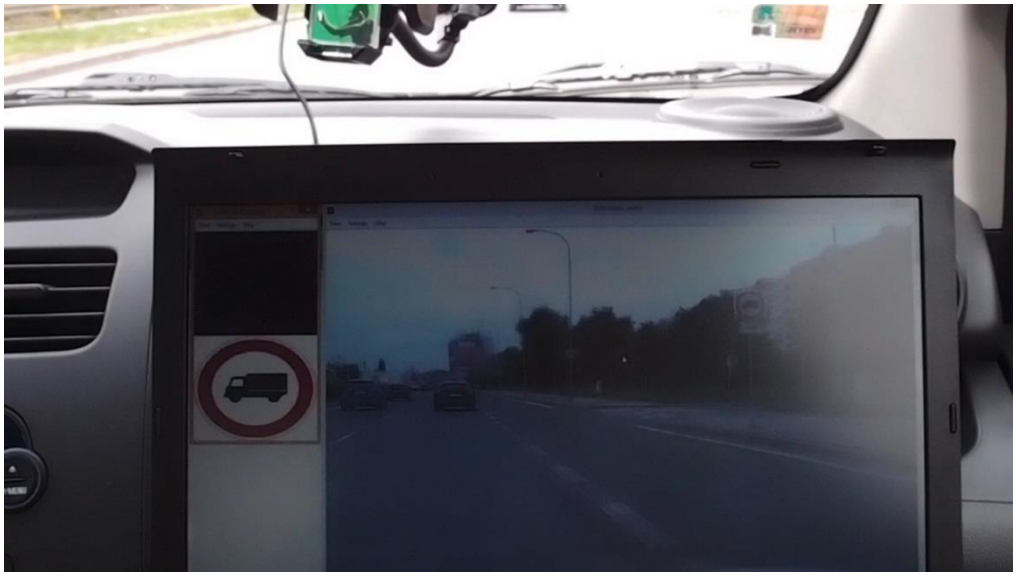
Příloha 2: Záznam obrazovky z testování na videozáznamu

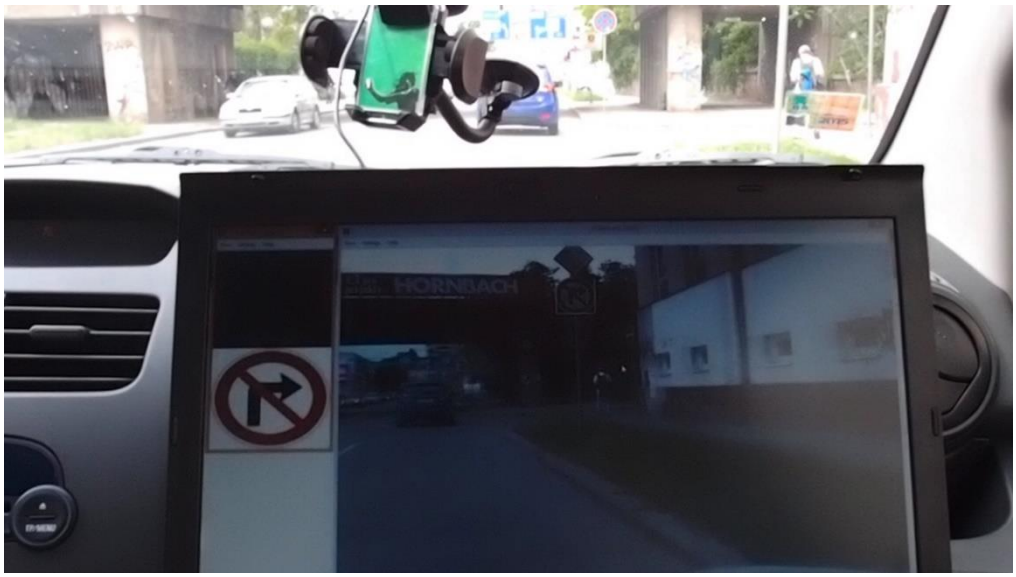

Zobrazeni_zn...
View Settings Help
Zobrazeni_video
View Settings Help
60
60
CamStudio
File Region Options Tools Effects View Help
80%
T=48.233
FixedStepDiscrete


Zobrazeni_zn...
View Settings Help
60
Zobrazeni_video
View Settings Help
60
60
CamStudio
File Region Options Tools Effects View Help
80%
T=48.233
FixedStepDiscrete


Zobrazeni_zn...
View Settings Help
60
Zobrazeni_video
View Settings Help
60
60
CamStudio
File Region Options Tools Effects View Help
80%
T=55.833
FixedStepDiscrete

Příloha 3: Fotografie z testování v provozu





Příloha 4: Ukázka kódu - funkce normalizace

```
function [NormSign, Normalized]  = Normalizace(imageRGB, SignBBox, detected,
recognised) %#codegen
%Funkce pro normalizaci detekovanych znacek na velikost 50x50 px v sede skale
(veikost pro rozpoznani piktogramu). K normalizaci pouziva jednoduchy rescale
%Inputs:    imageRGB - zakladni barevny obrazek
%           SignBBox - matice BBoxu, ktere oznacuji detekovane znacky
%           detected - vektor log hodnot udavajici, zda jiz byly znacky
%           detekovany (podminka detekce)
%           recognised - vektor log hodnot udavajici, zda jiz byly znacky
%           rozpoznany (podminka rozpoznani)
%Outputs:   NormSign - Array 50x50px gray obrazku (double), ktere predstavuji
%           normalizovane znacky
%           Normalized - vektor logickych hodnot urcujici na kterych
%           pozicich pole je normalizovana znacka


%Prealokace pole pro vysledne obrazky
NormSign=zeros(50,50,size(SignBBox,1), 'uint8');
Normalized=false(size(SignBBox,1),1);

%deklarace system objectu jako persistent
persistent colorConvSO
persistent RescaleSO

%inicializace vsech SO (pouze pokud dosud nejsou)
if isempty(colorConvSO)
%vytvoreni System objectu na barevnou konverzi
colorConvSO = vision.ColorSpaceConverter('Conversion','RGB to intensity');
end
if isempty(RescaleSO)
%inicializace GeometricRescaler szstem object
RescaleSO=vision.GeometricScaler('SizeMethod','Number of output rows and
columns', 'Size', [50, 50]);
end

%transformace obrazu na sede tony
imageGray=step(colorConvSO, imageRGB);

% vse je ve smycce pro jednotlive detekovane znacky
for Idx=1:size(SignBBox,1)
%podminka normalizace znacek, ktere uz byly 2 krat detekovany a jeste
%nebyly 2 krat rozpoznany a nejedna se o prazdny bbox
    if SignBBox(Idx,1)~=0 && detected(Idx)==1 && recognised(Idx) ==0

        %provedeni jednoduche normalizace - rescale na 50x50px
        NormSign(:,:,Idx)=step(RescaleSO,imorez(imageGray, SignBBox(Idx,:)));
        Normalized(Idx)=true;

    else
        Normalized(Idx)=false;
    end
end
end
```

Příloha 5: Výsledky testu detekce a rozpoznání – standardní počítač

		Nastavení				Rozlišení		Celkové počty					Úspěšnost v %		
Číslo testu	TEST sada	Segmentace	Detekce	Prahování	Sada vzorů	Šířka	Výška	Průměrná rychlost výpočtu	Testovacích značek	Správně detekovaných	Správně rozpoznaných	Falešně detekovaných	Falešně rozpoznaných	Detekce	Rozpoznání celková	Rozpoznání z detekce
1	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0640	150	114	109		1	3	76,00	72,67	95,61
2	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,1197	150	132	119	9	4	88,00	79,33	90,15
3	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0656	150	99	98	0	0	66,00	65,33	98,99
4	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,1026	150	115	108	4	2	76,67	72,00	93,91
5	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,3670	150	125	124	0	1	83,33	82,67	99,20
6	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,3954	150	143	140	9	2	95,33	93,33	97,90
7	Sero	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0620	75	3	3	0	0	4,00	4,00	100,00
8	Sero	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0856	75	8	5	0	1	10,67	6,67	62,50
9	Sero	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0624	75	29	20	4	5	38,67	26,67	68,97
10	Sero	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0885	75	36	22	4	5	48,00	29,33	61,11
11	Sero	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,3395	75	43	35	3	6	57,33	46,67	81,40
12	Sero	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,3698	75	50	36	5	8	66,67	48,00	72,00
13	Noc	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0666	95	40	34	0	5	42,11	35,79	85,00
14	Noc	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,1181	95	52	42	0	5	54,74	44,21	80,77
15	Noc	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0633	95	45	37	1	5	47,37	38,95	82,22
16	Noc	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0944	95	63	44	9	7	66,32	46,32	69,84
17	Noc	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,3547	95	23	22	0	1	24,21	23,16	95,65
18	Noc	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,4557	95	47	27	6	4	49,47	28,42	57,45

Příloha 6: Výsledky testu rozlišení – standardní počítač

		Nastavení				Rozlišení			Celkové počty					Úspěšnost v %		
Číslo testu	TEST sada	Segmentace	Detekce	Prahování	Sada vzorů	Šířka	Výška	Průměrná rychlost výpočtu	Testovacích značek	Správně detekovaných	Správně rozpoznaných	Falešně detekovaných	Falešně rozpoznaných	Detekce	Rozpoznání celková	Rozpoznání z detekce
1	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0204	10	10	10	10	0	100,00	100,00	100,00
2	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0501	10	10	10	1	0	100,00	100,00	100,00
3	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0149	10	9	7	0	1	90,00	70,00	77,78
4	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0320	10	9	7	0	1	90,00	70,00	77,78
5	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0511	10	9	9	0	0	90,00	90,00	100,00
6	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0755	10	9	9	0	0	90,00	90,00	100,00
7	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0478	10	10	10	0	0	100,00	100,00	100,00
8	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0841	10	10	10	0	0	100,00	100,00	100,00
9	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0533	10	7	7	0	0	70,00	70,00	100,00
10	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0656	10	7	7	0	0	70,00	70,00	100,00
11	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,2347	10	10	10	0	0	100,00	100,00	100,00
12	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,2505	10	10	10	0	0	100,00	100,00	100,00
13	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0667	10	10	10	0	0	100,00	100,00	100,00
14	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0950	10	10	10	2	0	100,00	100,00	100,00
15	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0641	10	10	10	0	0	100,00	100,00	100,00
16	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0932	10	10	10	0	0	100,00	100,00	100,00
17	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,3377	10	10	10	0	0	100,00	100,00	100,00
18	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,3652	10	10	10	0	0	100,00	100,00	100,00
19	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,0865	10	10	10	0	0	100,00	100,00	100,00
20	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,1255	10	10	10	1	0	100,00	100,00	100,00
21	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,0840	10	9	9	0	0	90,00	90,00	100,00
22	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,1139	10	9	9	0	0	90,00	90,00	100,00
23	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,5489	10	9	9	0	0	90,00	90,00	100,00
24	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,5840	10	9	9	0	0	90,00	90,00	100,00
25	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,1286	10	10	10	0	0	100,00	100,00	100,00
26	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,1612	10	10	10	0	0	100,00	100,00	100,00
27	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,1207	10	9	8	0	0	90,00	80,00	88,89
28	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,1661	10	9	8	0	0	90,00	80,00	88,89
29	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,8604	10	9	9	0	0	90,00	90,00	100,00
30	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,9047	10	9	9	2	0	90,00	90,00	100,00
31	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,5311	10	10	10	0	0	100,00	100,00	100,00
32	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,6446	10	10	10	2	0	100,00	100,00	100,00
33	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,5361	10	8	7	0	0	80,00	70,00	87,50
34	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,5883	10	8	7	1	0	80,00	70,00	87,50
35	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	3,4601	10	7	7	0	0	70,00	70,00	100,00
36	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	3,5342	10	7	7	0	0	70,00	70,00	100,00

Příloha 7: Výsledky testu detekce a rozpoznání – výkonný počítač

		Nastavení				Rozlišení		Celkové počty						Úspěšnost v %		
Číslo testu	TEST sada	Segmentace	Detekce	Prahování	Sada vzorů	Šířka	Výška	Průměrná rychlost výpočtu	Testovacích značek	Správně detekovaných	Správně rozpoznaných	Falešně detekovaných	Falešně rozpoznaných	Detekce	Rozpoznání celková	Rozpoznání z detekce
1	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0209	150	114	109	1	3	76,00	72,67	95,61
2	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0382	150	132	119	9	4	88,00	79,33	90,15
3	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0204	150	99	98	0	0	66,00	65,33	98,99
4	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0353	150	115	108	4	2	76,67	72,00	93,91
5	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0908	150	123	122	0	1	82,00	81,33	99,19
6	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,1143	150	144	140	8	1	96,00	93,33	97,22
7	Sero	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0209	75	3	3	0	0	4,00	4,00	100,00
8	Sero	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0294	75	8	5	0	1	10,67	6,67	62,50
9	Sero	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0197	75	29	20	4	5	38,67	26,67	68,97
10	Sero	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0325	75	36	22	4	5	48,00	29,33	61,11
11	Sero	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,0872	75	43	35	3	6	57,33	46,67	81,40
12	Sero	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	601	0,1091	75	50	36	5	8	66,67	48,00	72,00
13	Noc	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0213	95	40	34	0	5	42,11	35,79	85,00
14	Noc	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0435	95	52	42	0	5	54,74	44,21	80,77
15	Noc	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0202	95	45	37	1	5	47,37	38,95	82,22
16	Noc	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0353	95	63	44	9	7	66,32	46,32	69,84
17	Noc	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0922	95	23	22	0	1	24,21	23,16	95,65
18	Noc	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,1404	95	47	27	6	4	49,47	28,42	57,45

Příloha 8: Výsledky testu rozlišení – výkonný počítač

		Nastavení				Rozlišení			Celkové počty					Úspěšnost v %	
Číslo testu	TEST sada	Segmentace	Detekce	Prahování	Sada vzorů	Šířka	Výška	Průměrná rychlost výpočtu	Testovacích značek	Správně detekovaných	Správně rozpoznaných	Falešně detekovaných	Falešně rozpoznaných	Detekce	Rozpoznání celková	Rozpoznání z detekce
1	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0065	10	10	10	10	0	100,00	100,00	100,00
2	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0198	10	10	10	1	0	100,00	100,00	100,00
3	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0053	10	9	7	0	1	90,00	70,00	77,78
4	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0126	10	9	7	0	1	90,00	70,00	77,78
5	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0141	10	9	9	0	0	90,00	90,00	100,00
6	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	320	240	0,0236	10	9	9	0	0	90,00	90,00	100,00
7	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0152	10	10	10	0	0	100,00	100,00	100,00
8	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0301	10	10	10	0	0	100,00	100,00	100,00
9	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0163	10	7	7	0	0	70,00	70,00	100,00
10	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0216	10	7	7	0	0	70,00	70,00	100,00
11	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0606	10	10	10	0	0	100,00	100,00	100,00
12	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	640	480	0,0785	10	10	10	0	0	100,00	100,00	100,00
13	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0216	10	10	10	0	0	100,00	100,00	100,00
14	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0355	10	10	10	2	0	100,00	100,00	100,00
15	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0209	10	10	10	0	0	100,00	100,00	100,00
16	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0362	10	10	10	0	0	100,00	100,00	100,00
17	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,0931	10	10	10	0	0	100,00	100,00	100,00
18	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	800	600	0,1162	10	10	10	0	0	100,00	100,00	100,00
19	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,0267	10	10	10	0	0	100,00	100,00	100,00
20	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,0482	10	10	10	1	0	100,00	100,00	100,00
21	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,0281	10	9	9	0	0	90,00	90,00	100,00
22	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,0427	10	9	9	0	0	90,00	90,00	100,00
23	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,1521	10	9	9	0	0	90,00	90,00	100,00
24	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1024	768	0,1751	10	9	9	0	0	90,00	90,00	100,00
25	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,0414	10	10	10	0	0	100,00	100,00	100,00
26	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,0675	10	10	10	0	0	100,00	100,00	100,00
27	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,0419	10	9	8	0	0	90,00	80,00	88,89
28	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,0619	10	9	8	0	0	90,00	80,00	88,89
29	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,2309	10	9	9	0	0	90,00	90,00	100,00
30	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	1280	960	0,2562	10	9	9	2	0	90,00	90,00	100,00
31	Den	'YCbCr statická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,1688	10	10	10	0	0	100,00	100,00	100,00
32	Den	'YCbCr statická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,2081	10	10	10	2	0	100,00	100,00	100,00
33	Den	'YCbCr dynamická'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,1563	10	8	7	0	0	80,00	70,00	87,50
34	Den	'YCbCr dynamická'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,2014	10	8	7	1	0	80,00	70,00	87,50
35	Den	'HSV'	'Porovnání'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,9052	10	7	7	0	0	70,00	70,00	100,00
36	Den	'HSV'	'Houghova transformace'	'Otsuho metoda'	'Rozšířená sada'	2560	1920	0,9514	10	7	7	0	0	70,00	70,00	100,00