

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Rozpoznávání vodorovného dopravního značení
pomocí algoritmů zpracování obrazu

Petr Mizera

Diplomová práce
2012

zadání

zadání

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména ze skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 15. 5. 2012

Petr Mizera

Poděkování

Děkuji vedoucímu diplomové práce Ing. Martinu Dobrovolnému, PhD. za jeho cenné rady a připomínky při tvorbě práce. Dále bych chtěl poděkovat panu Šíldovi za upevnění LCD displeje na vývojovém kitu. V neposlední řadě děkuji mé rodině.

Anotace

Diplomová práce se věnuje v současné době velmi aktuální problematice rozpoznávání vodorovného dopravního značení, která nalézá uplatnění především v bezpečnostních asistenčních systémech u automobilů. V práci je uveden kompletní návrh, optimalizace a implementace algoritmu rozpoznávání vodorovného značení.

V první části práce je uveden rozbor problematiky. Je provedena analýza současných komerčních řešení a řešeních algoritmů rozpoznávání vodorovného dopravního značení. Druhá část práce uvádí návrh vlastního algoritmu v systému MATLAB, ve kterém je vytvořeno pomocné grafické interaktivní prostředí, které bylo využito při návrhu a optimalizaci algoritmu. Dále je zde uveden proces pořizování reálných obrazových dat. Třetí část je zaměřená na implementaci navrženého algoritmu do vestavěného systému Olimex SAM-L9261. Poslední část uvádí provedené experimenty a testování navrženého algoritmu na silničních úsecích, které splňují požadavky doporučení ISO 17361:2007 a FMCSA-MCRR-05-005.

Klíčová slova

Detekce jízdního pruhu, LDW, Houghova transformace, Linux, OpenCV, GTK+

Title

The design of detection system for traffic lane recognition based on image processing.

Annotation

This thesis deals with the very current topical issue of Recognition of road markings, which in particular finds its application in security assistance systems for cars. The thesis presents a complete design, optimization and implementation of a road markings recognition algorithm.

The first part contains an analysis of the issue. An analysis of existing commercial solutions and solutions of road markings recognition algorithms is made. The second part introduces a design of a custom algorithm created in the MATLAB system, in which an auxiliary graphic interactive environment is created and which was used during the design and optimization of the algorithm. It also shows the process of acquiring the real image data. The third part focuses on the implementation of the designed algorithm on the embedded Olimex SAM-L9261 system. The last part shows performed experiments and testing of the designed algorithm on road segments which meet the recommendations of ISO 17361:2007 and FMCSA-MCRR-05-005.

Keywords

Lane Marking detection, LDW, Hough transform, Linux, OpenCV, GTK+

Obsah

Seznam zkratk.....	8
Seznam obrázků.....	9
Seznam tabulek.....	10
1 Úvod	11
2 Rozbor problematiky	12
2.1 Počátek vývoje systémů LDW	12
2.2 Legislativa	13
2.3 Přehled současných řešení	15
2.4 Používané technologie snímačů.....	16
2.5 Stanovení cílů práce.....	18
3 Problematika algoritmů zpracování obrazu	19
3.1 Proces digitalizace	19
3.2 Metody předzpracování obrazu	20
3.2.1 Bodové operace	20
3.2.2 Detekce hran	21
3.2.3 Matematická morfologie.....	25
3.3 Segmentace obrazu	27
3.3.1 Metoda vycházející z detekce hran.....	27
3.3.2 Metoda orientovaná na regiony	29
3.3.3 Metoda barevné segmentace.....	30
3.4 Přehled stávajících algoritmů pro rozpoznávání vod. dop. značení	32
4 Pořizování podkladových obrazových dat	34
4.1 Specifikace snímacích zařízení.....	34
4.2 Umístění kamery	34
5 Popis vlastního řešení v prostředí MATLAB.....	36
5.1 Interaktivní grafické prostředí	43
5.1.1 Popis pracovního dialogového okna.....	44
5.1.2 Načtení a zobrazení vstupních dat	44
5.1.3 Práce s obrazovými daty.....	45
6 Realizace systému rozpoznávání jízdního pruhu	49
6.1 Hardwarový modul	49
6.2 Operační systém GNU/Linux	51

6.2.1	Vytváření aplikací pro platformu ARM	52
6.3	Použité open-source knihovny.....	54
6.4	Přístup ke kameře	57
7	Implementace	59
7.1	Transformace navrženého algoritmu do jazyka C.....	59
7.2	Vytvoření grafického uživatelského rozhraní.....	63
8	Zhodnocení funkčnosti navrženého systému	65
9	Závěr	68
	Literatura	69
	Příloha A.....	69
	Příloha B.....	75
	Příloha C.....	77
	Příloha D.....	78

Seznam zkratek

LDW	Lane Departure Warning
WHO	Světová Zdravotnická Organizace
NHTSA	National Highway Traffic Safety Administration
IIHS	Insurance Institute for Highway Safety
FMCSA's	Federal Motor Carrier Safety Administration's
CCD	Charge – Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
GPS	Global Positioning System
MATLAB	MATrix LABoratory
LOG	Laplacian of Gaussian
CIE	Commission International de l'Eclairage
DCT	Diskrétní kosinova transformace
ROI	Region of Interest
ARM	Advanced RISC Machine
MMU	Memory Management Unit
USB	Universal Serial Bus
OpenCV	Open Source Computer Vision
BSL	Boost Software License
LCD	Liquid Crystal Display
HT	Hough transformace
IEEE	Institute of Electrical and Electronics Engineers

Seznam obrázků

Obr. č. 1 – Vývoj celosvětové úmrtnosti dle WHO [23]	12
Obr. č. 2 – Varovné prahy dle doporučení FMCSA-MCRR-05-005 pro LDW [8]	14
Obr. č. 3 – Umístění kamery v LDW [33]	16
Obr. č. 4 – Umístění infračervených senzorů [29]	16
Obr. č. 5 – Využití infračervených senzorů v LDW [29]	17
Obr. č. 6 – Využití magnetických prvků [31]	17
Obr. č. 7 – Přejížděvací charakteristiky	20
Obr. č. 8 – Histogram monochromatického obrazu + úrovněvý výřez a binární obraz	20
Obr. č. 9 – Masky pro okolí velikosti 2x2 obrazové body	22
Obr. č. 10 – Aplikace konvoluční masky na obraz [31]	22
Obr. č. 11 – Gradientní obraz a binární obraz	23
Obr. č. 13 – Masky aproximující první derivaci	23
Obr. č. 14 – Průběh v řádku: hodnot jasu, první derivace, druhé derivace	23
Obr. č. 15 – Masky aproximující první derivaci	24
Obr. č. 16 – Strukturní elementy	25
Obr. č. 17 – Použitý strukturní element	26
Obr. č. 18 – Binární obraz po aplikaci dilatace	26
Obr. č. 19 – Zobrazení přímky v kartézských souřadnicích a v parametrickém prostoru ...	27
Obr. č. 20 – Proces detekce jízdního pruhu využitím HT	28
Obr. č. 21 – Binární obraz	29
Obr. č. 22 – Detekce jízdního pruhu	29
Obr. č. 23 – Znázornění RGB modelu pomocí krychle [38]	30
Obr. č. 24 – Kónická reprezentace HSV modelu [24]	31
Obr. č. 25 – Detekce vozovky v modelu HSV	31
Obr. č. 26 – Blokové schéma algoritmu LDW	32
Obr. č. 27 – Blokové schéma algoritmu LDW	33
Obr. č. 28 – Genius Face CAM 1320	34
Obr. č. 29 – Hercules webcam classic	34
Obr. č. 30 – Umístění kamery	35
Obr. č. 31 – Umístění kamery na příčnici	35
Obr. č. 32 – Blokové schéma navrženého algoritmu	36
Obr. č. 33 – ROI	37
Obr. č. 34 – Testování hranových detektorů	38
Obr. č. 35 – Strukturní element	39
Obr. č. 36 – Binární obraz (detekované hrany)	39
Obr. č. 37 – Výsledky morfologických transformací	39
Obr. č. 38 – Význam parametrů HT	40
Obr. č. 39 – Parametrický prostor	41
Obr. č. 40 – Blokové schéma navrženého algoritmu	42
Obr. č. 41 – Výstup z navrženého algoritmu v systému MATLAB	42
Obr. č. 42 – Vytvořené grafické prostředí	44

Obr. č. 43 – Načtení vstupních dat	44
Obr. č. 44 – Vstupní obraz.....	45
Obr. č. 46 – Histogram šedé stupnice.....	45
Obr. č. 45 – Monochromatický vstupní obraz	45
Obr. č. 47 – Histogram po ekvalizaci obraz	45
Obr. č. 48 – Monochromatický obraz po ekvalizaci histogramu.....	46
Obr. č. 49 – Vliv prahování na monochromatický obraz	46
Obr. č. 50 – Aplikace hranového detektoru na monochromatický obraz	47
Obr. č. 51 – Aplikace morfologické operace dilatace, typ struk. element line na obraz	47
Obr. č. 52 – Parametrický prostor.....	48
Obr. č. 54 – Detekované přímky pomocí hough transformace pod úhlem 30° - 60°	48
Obr. č. 53 – Zobrazení detekovaných maxim.....	48
Obr. č. 55 – Blokové schéma hardwarového modulu.....	50
Obr. č. 56 – Vývojová deska Olimex SAM-L9261	50
Obr. č. 57 – Prostředí GNOME distribuce Ubuntu 12.04	51
Obr. č. 58 – Vytváření aplikací pro x86 a ARM	53
Obr. č. 59 – Blokové schéma procesu implementace.....	60
Obr. č. 60 – Blokové schéma algoritmu	62
Obr. č. 61 – Testovací stanoviště.....	65
Obr. č. 62 – Detekované vodorovné dopravní značení.....	66
Obr. č. 63 – Aplikace využívající funkce knihovny GTK+	77
Obr. č. 64 – Aplikace využívající funkce GTK+ a OpenCV	80

Seznam tabulek

Tab. č. 1 – Přehled současných systémů[4].....	15
--	----

1 Úvod

Rozpoznávání vodorovného dopravního značení s cílem varovat řidiče v případě nebezpečí, kdy automobil začne opouštět jízdní pruh, je v současné době u automobilů v asistenčních bezpečnostních systémech velmi aktuální problematika. Tyto systémy jsou označovány LDW (Lane Departure Warning), neboli varovné systémy při neúmyslném opuštění jízdního pruhu. Systémy LDW se stávají postupně součástí nových moderních vozidel a postupem času je předpokládáno, že budou standardním vybavením každého vozu.

Cílem diplomové práce je vytvoření systému rozpoznávání vodorovného dopravního značení, který bude umožňovat předávání této informace řidiči prostřednictvím vizualizačního systému.

V počátku diplomové práce je proveden rozbor problematiky, ve kterém jsou analyzována současná komerční řešení. Další část se věnuje problematice číslicového zpracování obrazu a přehledu současných algoritmů pro rozpoznávání vodorovného dopravního značení. Čtvrtá kapitola uvádí provedená měření, při kterých byla pořízena obrazová data. Tyto počáteční části se staly základem při návrhu vlastního řešení algoritmu detekce jízdního pruhu.

Hlavní část diplomové práce prezentuje vlastní řešení návrhu algoritmu rozpoznávání vodorovného dopravního značení v systému MATLAB, optimalizaci, dále realizaci a implementaci algoritmu do vestavěného systému.

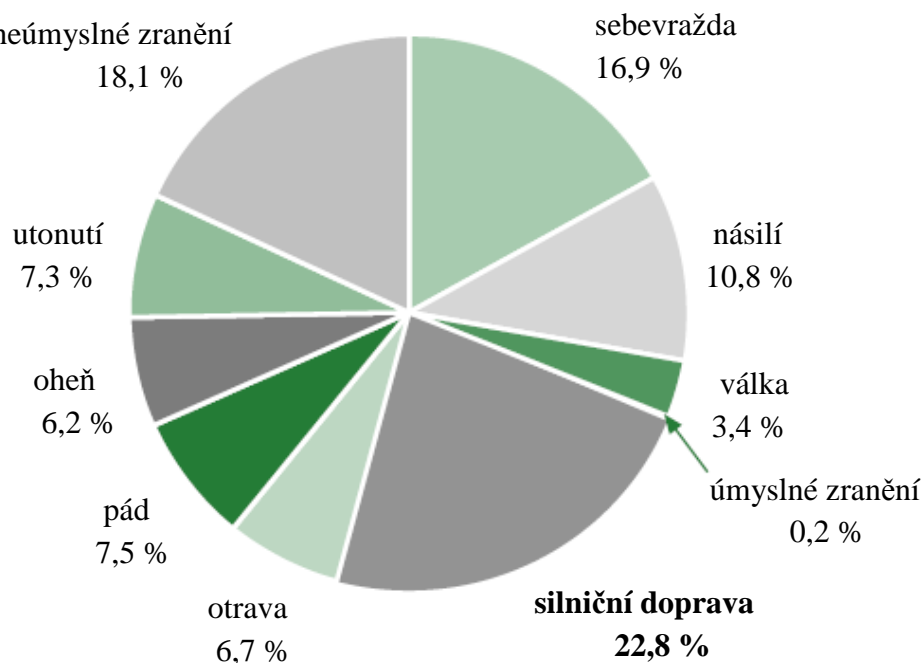
2 Rozbor problematiky

Před návrhem vlastního systému rozpoznávání vodorovného dopravního značení, bylo nutné provést rozbor stávajících řešení. Tento rozbor se následně stal základem pro vlastní řešení. Proto tato kapitola informuje o počátku vývoje systémů LDW, dále o vznikající legislativě pro tyto systémy a o přehledu současných komerčních řešení.

2.1 Počátek vývoje systémů LDW

Počátek vývoje těchto systémů sahá do roku 1995, kdy začaly vznikat na výzkumných pracovištích a řadě světových univerzit první studie těchto systémů. Důvodem zahájení vývoje systému, který by byl schopen upozornit řidiče v okamžiku, kdy automobil bude opouštět vozovku, bylo zjištění, že jeden z hlavních příčin úmrtí v automobilové dopravě jsou hlavně havárie, které jsou způsobeny vyjetím automobilu z jízdního pruhu [23]. I v současné době jsou alarmující statistiky, prezentované Světovou Zdravotnickou Organizací (WHO), která provedla pozorování vývoje celosvětové úmrtnosti s jejími příčinami od roku 1990 k roku 2020.

Vývoj celosvětové úmrtnosti s jejími příčinami od roku 1990 k roku 2020



Obr. č. 1 – Vývoj celosvětové úmrtnosti dle WHO [23]

Z Obr. č. 1 je vidět, že nejvíce úmrtnosti je zapříčiněno silničním dopravou a i do budoucna je predikován její stálý nárůst.

Proto je v současné době stále celosvětovou snahou nalézt takové technologie, které by snížily počet úmrtí v automobilové dopravě. První krok ke snížení dopravních nehod provedla OSN (Organizace spojených národů) v roce 2010 v New Yorku vyhlášením dekády 2011-2020 za „Dekády akcí pro vyšší bezpečnost silničního provozu ve světě“¹, jejímž cílem je zvýšit bezpečnost v silničním provozu. Následně se připojily i orgány Evropské unie, kdy 28. 3. 2011 Evropská komise zveřejnila Bílou knihu. Cílem je, aby se snížil počet dopravních nehod do roku 2020 na polovinu a do roku 2050 bylo počet úmrtí v silniční dopravě téměř rovno nule. Tuto strategii podporuje i Česká republika [9].

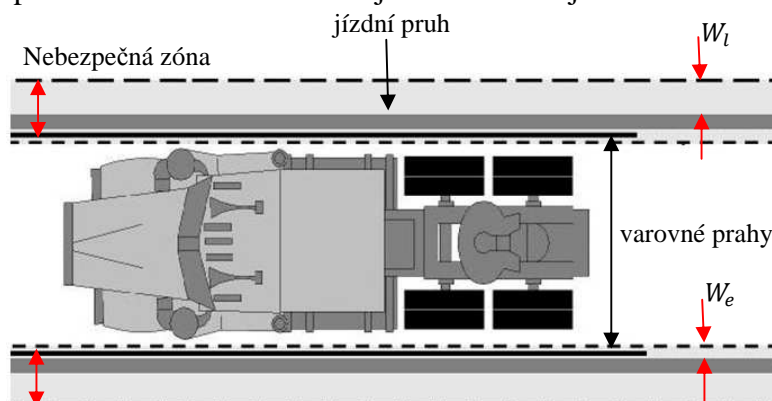
Další důležité zjištění obsahuje seznam hlavních příčin dopravních nehod, vydaný americkým úřadem pro bezpečnost silničního provozu NHTSA, který uvádí, že 41% nehod z celkového počtu dopravních nehod jsou zapříčiněny vyjetím vozidla z jízdního pruhu. NHTSA podporuje vývoj a využívání technologií varování před vyjetím vozidla z jízdního pruhu (LDW) a uvádí, že tento bezpečnostní systém by měl zabránit až 483 000 nehod ročně. Dále IIHS (Insurance Institute for Highway Safety) uvádí, že 10 000 lidí umírá ročně z důvodu vyjetí vozidla z jízdního pruhu [2], [12], [19]. V České republice je mezi 10 nejčtenějšími příčinami silničních dopravních nehod vždy na prvním či druhém místě uváděna příčina, že se řidič plně nevěnoval řízení vozidla, kdy tato příčina je uváděna i na prvních místech nejtragičtějších příčin silničních dopravních nehod. Za rok 2010 je uváděno 12 332 nehod, které byly způsobeny nepozorností řidiče při řízení vozidla a touto příčinou bylo usmrceno 88 osob [28].

2.2 Legislativa

V současné době neexistují v Evropě ani v USA předpisy, které by se týkaly systému LDW. Tvorbou norem pro systém LDW se zabývá Mezinárodní organizace pro normalizaci (ISO). Pro systémy LDW v současné době vznikla norma ISO 17361:2007 a doporučení FMCSA-MCRR-05-005 vydané U. S. ministerstvem dopravy (FMCSA's - Federal Motor Carrier Safety Administration's). Z těchto doporučení dále uvedu jen důležité body, ze kterých bude dále v diplomové práci vycházeno, při tvorbě systému rozpoznávání vodorovného dopravního značení. Požadavky na LDW systém lze rozdělit zásadě do dvou oblastí. První oblast se týká stanovení prahu vzdálenosti vozidla od okrajů

¹ http://www.ibesip.cz/files/=4221/NSBSP%2b2011-2020_form%c3%a1tov%c3%a1n%c3%ad_II.pdf

vozovky. V okamžiku kdy vozidlo překročí tuto vzdálenost by měl systém zareagovat a vydat varovné upozornění řidiči. Na následujícím Obr. č. 2 je tato situace znázorněná.



Obr. č. 2 – Varovné prahy dle doporučení FMCSA-MCRR-05-005 pro LDW [8]

V doporučení FMCSA-MCRR-05-005 je uvedený parametr rychlost vybočení z jízdního pruhu $V_d < 0,8$ m/s, který když je menší jak hodnota 0,8 m/s, tak by měl systém vydat varovné upozornění. Bohužel v normě ještě nejsou uvedeny požadavky na hodnoty W_e a W_l varovných vzdáleností od okraje vozovky. Tyto hodnoty jsou ale uvedeny v normě ISO 17361:2007, která stanovuje pro osobní automobily hodnotu $W_l = 0,3$. Velikost hodnoty W_e uvádějí, že je závislá na hodnotě V_d . Detailní popis lze nalézt v [15].

Druhá oblast obsahuje důležitější informace, které jsou stěžejní pro tuto práci a to stanovení hodnot rychlosti a zakřivení vozovky, za kterých má být systém LDW provozu schopen. V následujících bodech jsou uvedeny jednotlivé hodnoty pro uvedené normy.

- ISO 17361:2007
 - Třída I – systém LDW musí být provozu schopen:
 - při rychlosti vozidla ≥ 72 km/h,
 - při poloměru vozovky ≥ 500 m.
 - Třída II – systém LDW musí být provozu schopen:
 - při rychlosti vozidla ≥ 61 km/h,
 - při poloměru vozovky ≥ 250 m.
- FMCSA-MCRR-05-005
 - Třída I – systém LDW musí být provozu schopen:
 - při rychlosti vozidla ≥ 60 km/h,
 - při poloměru vozovky ≥ 250 m.

Dále jsou v normě FMCSA-MCRR-05-005 uvedeny typy značení jízdních pruhů, které se vyskytují v USA. Systém by měl fungovat při bílém či žlutém značení jízdního

pruhu, dále při plném či přerušovaném značení. Norma ISO 17361:2007 uvádí i požadavky na viditelnost, systém by měl pracovat jak za denního světla, soumraku, tak i v noci.

V poslední řadě by měl systém LDW projít řadou testů, které jsou prováděny v úsecích, kde je vodorovné dopravní značení v dobrém stavu dle národní definice viditelnosti značení jízdního pruhu [8], [15].

2.3 Přehled současných řešení

V současné době byla vyvinuta řada systémů varujících před vyjetím z jízdního pruhu. V této části bych uvedl systémy dostupné na současném automobilovém trhu, které jsou určené jak pro osobní automobily, tak i nákladní automobily. Následující tabulka zobrazuje přehled současných dostupných řešení.

OEM	Systém	Modely	Výrobce systému	Technologie	Akt. při rychlosti	Typ upozornění
Audi/VW	Asistenčními systémy řidiče	Q7, A8	Hella	Video kamera	65 km/h	Vybrace volantu
BMW	LDW	5, 6 série	Siemens VDO & Mobileye	Video kamera	70 km/h	Vizuální, vybrace volantu
GM	LDW	Casillac STS, Casilac DTS, Buick	Mobileye	Video kamera	56 km/h	Visualní, zvukové
Volvo	LDW	V70, XC70, S80	Mobileye	Video kamera	64 km/h	Zvukové
Citroen	LDW	C4, C5	Iteris	6 párů infra-červených senzorů	80 km/h	Vybracemi sedadla
MAN	Lane Guidance System	Nákladní	Iteris	Video kamera	60 km/h	Zvukové
Mercedes	SPA	Nákladní	Iteris	Video kamera	60 km/h	Zvukové
Lexus	LDW	LS 460	Denso	Video kamera		Visualní, zvukové
After Marker	SafeTRAC	Nákladní	Assistware	Video kamera		Visualni
Mercedes	SPA	Osobní	Iteris	Video kamera	80 km/h	Vybracemi sedadla
Nissan	LDW	Infiniti M45, Infiniti FX	Iteris & Valeo	Video kamera	72 km/h	Vizuální, zvukové

Tab. č. 1 – Přehled současných systémů[4]

V dnešní době lze tabulku rozšířit o nový automobil Opel Mokka, který je založený na kamerovém systému Opel Eye, Kia Kadenza, dále o nově představený na Detroitském autosalólu 2012 vůz Honda Accord Coupé již také obsahuje LDW systém.

2.4 Používané technologie snímačů

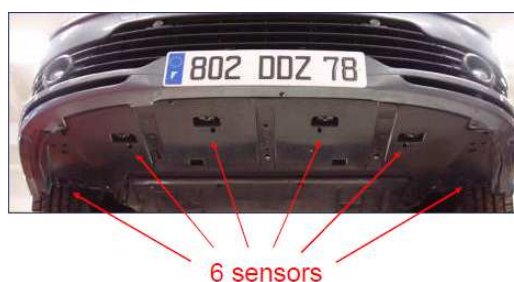
Z tabulky č. 1 je vidět, že pro systémy LDW se zásadně využívají dvě technologie snímačů. LDW systémy nejčastěji, dle tabulky č. 1 využívají snímacího prvku video kamery. Ta je většinou umístěna v místě zpětného zrcátka a jejím hlavním cílem je snímat scénu před jedoucím vozidlem. V LDW systémech jsou využívány především digitální video kamery, které využívají dvou technologií snímacích čipů a to technologie CCD (Charge – Coupled Device) a CMOS (Complementary Metal Oxide Semiconductor).

Nejčastěji jsou digitální kamery umístěny u zpětného zrcátka, situaci znázorňuje obrázek č. 3.



Obr. č. 3 – Umístění kamery v LDW [33]

V některých systémech LDW je využíváno pro detekci vodorovného dopravního značení infračervených senzorů. Senzory jsou zabudované v předním nárazníku, zobrazeno na obr. č. 4.



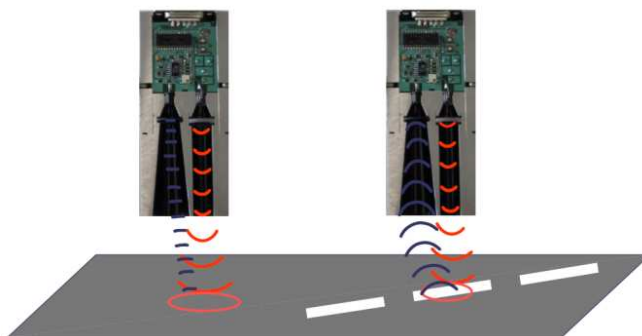
Obr. č. 4 - Umístění infračervených senzorů [29]

Infračervené senzory se skládají z diody (vysílače) emitující infračervené záření a z fotodiody (přijímače) detektoru infračerveného záření. Infračervené záření zahrnuje vlnové délky od 760nm do 1mm.

Tento rozsah v elektromagnetickém spektru lze rozdělit do tří částí:

- Blízkou infračervenou oblast – zaujímá vlnové délky $0,75\text{--}3\ \mu\text{m}$.
- Střední infračervenou oblast – tvořenou vlnovými délkami $3\text{--}6\ \mu\text{m}$.
- Dalekou infračervenou oblast – v intervalu vlnových délek $6\text{--}100\ \mu\text{m}$.

Příklad aplikace v LDW systémech je znázorněn na obr. č. 5.



Obr. č. 5 – Využití infračervených senzorů v LDW [29]

Tento systém se snaží rozpoznat rozdíl mezi odrazivostí vodorovného dopravního značení a odrazivostí vozovky.

Další technika, která má za cíl zabránit nekontrolovatelnému opuštění vozidla z jízdního pruhu je založená na zabudovaných magnetických prvcích ve vozovce. Tato technika by umožňovala vozidlu pomocí detektoru přímou detekci vozovky v reálném čase. Jednalo by se o nejrobustnější řešení, bohužel zásadní nevýhodou je, že tyto prvky by musely být zabudovány do veškerých silnic, tato skutečnost zatím neumožňuje realizaci tohoto systému z finančních důvodů [5]. Následující obr. č. 6 zobrazuje realizaci tohoto systému při příležitosti Expo 2005 v Aichi Japonsku, kdy byly představeny tři autonomní autobusy.



Obr. č. 6 – Využití magnetických prvků [31]

Červenými čtverci je na obr. č. 6 zvýrazněno umístění magnetických prvků ve vozovce.

Poslední zde uváděnou technikou je kombinace velmi přesných digitálních map a navigačního systému GPS (Global Positioning System) při požadované přesnosti 0,5m nebo lepší. Technika je testována v USA v Minneapolis, kde využívají pro dosažení maximální přesnosti diferenciální GPS. Tato technika naráží při realizaci na problém, který je způsoben požadavkem na vytvoření přesných map pro funkci tohoto systému. Realizace toho systému by byla jako předchozího velmi finančně náročná, a proto tento systém v současné době nenalézá uplatnění [5].

Z provedené analýzy systémů LDW se bude dále diplomová práce soustředit především na techniku řešení využívající digitální kamery, která je umístěná u zpětného zrcátka a dále na algoritmy zpracování obrazu s cílem detekovat vodorovné dopravní značení v zaznamenaném obraze.

2.5 Stanovení cílů práce

Cílem práce je vytvořit systém rozpoznávání vodorovného dopravního značení na vozovce a předávání této informace řidiči prostřednictvím vizualizačního systému.

Zásady pro vypracování diplomové práce:

- Analýza problematiky a rešerše stávajících způsobů řešení.
- Pořízení podkladových obrazových dat.
- Klasifikace, roztrídění a před příprava dat.
- Návrh vlastního algoritmu (např. programové prostředí MATLAB).
- Optimalizace parametrů algoritmu s ohledem na zpracování v reálném čase.
- Transformace navržených algoritmů a jejich implementace do zvoleného zařízení.

Dále bylo stanoveno, že systém rozpoznávání vodorovného dopravního značení bude realizován na stávajícím vývojovém kitu a bude funkce schopný na úsecích s kvalitním dopravním značením a za dobrých světelných podmínek.

3 Problematika algoritmů zpracování obrazu

V této kapitole budou uvedeny metody číslicového zpracování obrazu, které jsou využívány v následně uvedených algoritmech detekce vodorovného dopravního značení. První část této kapitoly se věnuje základním vlastnostem procesu digitalizace obrazové informace. Následné části se již věnují metodám číslicového zpracování obrazu a poslední část této kapitoly prezentuje přehled stávajících algoritmů využitých při detekci vodorovného dopravního značení.

3.1 Proces digitalizace

Matematický model statického obrazu je reprezentován spojitou skalární obrazovou funkcí $f(x, y)$, kde proměnné x, y odpovídají buňkám na snímacím čipu. Aby bylo možné využít algoritmů číslicového zpracování signálu, musí být nejprve elektrické napětí, které je úměrné dopadajícímu světlu (fotonům) na snímací buňku čipu CCD, nebo CMOS převedeno do diskrétní podoby. Proces digitalizace se skládá z kroků filtrace antialiasingovým filtrem typu DP, který má za cíl omezit spektrum spojitého signálu, poté následuje vzorkování, při kterém musí být dodrženy podmínky Shannon-Kotelníkova či Nyquistova vzorkovacího teorému. Výsledkem vzorkování je signál diskrétní v čase, ale stále spojitý v hodnotách. Proto následuje proces kvantování, které provede převod spojitých hodnot napětí (jasu) do konečné množiny $k = 2^b$ diskrétních hodnot. Nejčastěji je využíváno kvantování 8 bitů na obrazový bod, které provádí i kamera použitá v této práci. Tyto snímací buňky mohou být na čipu rozmístěny do několika plošných tvarů. Většinou se využívá pravidelná struktura, kdy tuto rovinu mohou zcela pokrýt tři pravidelné mnohaúhelníky: rovnostranné trojúhelníky, čtverce a pravidelné šestiúhelníky. V praxi je nejčastěji využívána čtvercová struktura, jak je uváděno v [14].

Procesem digitalizace je obrazová funkce $f(x, y)$ reprezentována maticí

$$\mathbf{F} = \begin{pmatrix} (0, 0) & \cdots & (M-1, 0) \\ \vdots & \ddots & \vdots \\ (0, N-1) & \cdots & (M-1, N-1) \end{pmatrix},$$

kde jednotlivé prvky této matice odpovídají jednotlivým buňkám na čipu senzoru a jejich hodnota je úměrná dopadajícím fotonům [14], [17], [29]. Takovouto reprezentaci obrazových dat používá i systém MATLAB, v kterém byl proveden návrh vlastního algoritmu.

3.2 Metody předzpracování obrazu

V této části budou uvedeny metody předzpracování obrazu, které mají za cíl co nejlépe připravit obraz pro následné segmentační metody.

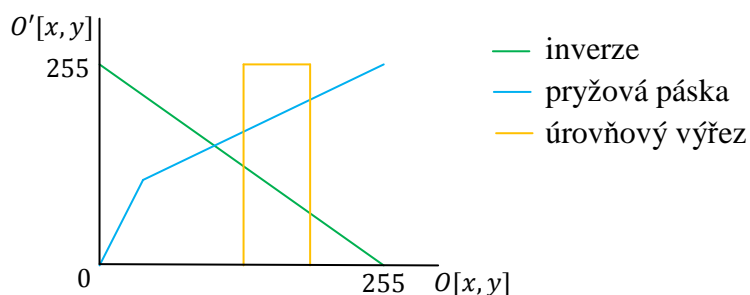
3.2.1 Bodové operace

Operace s jednotlivými obrazovými body jsou nejjednodušší operace, které lze s obrazem provádět. Transformace T hodnot bodů jasu lze zapsat rovnicí č. 2

$$O'[x, y] = T(O[x, y]),$$

Rovnice č. 2

kde $O[x, y]$ je vstupní hodnota jasu, $O'[x, y]$ je hodnota jasu po transformaci. Pro tyto transformace je charakteristické, že jsou nezávislé na poloze v obraze. Na obr. č. 7 jsou zobrazeny nejčastěji využívané přechodové charakteristiky.



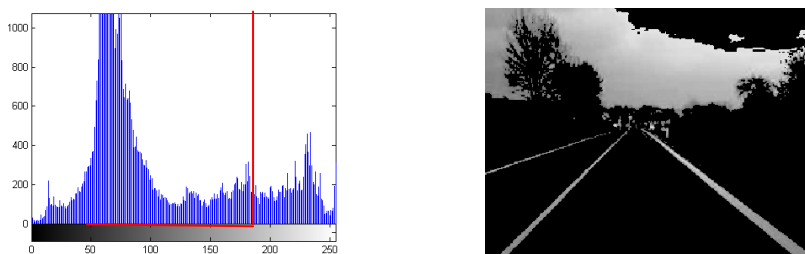
Obr. č. 7 – Přechodové charakteristiky

Na obr. č. 8 je zobrazen histogram monochromatického obrazu, který představuje četnost výskytu jasové úrovně v obraze, dáno rovnicí

$$N = N(u),$$

Rovnice č. 3

kde u je příslušná úroveň, N je četnost výskytu této úrovně.



Obr. č. 8 – Histogram monochromatického obrazu + úrovňový výřez a binární obraz

Histogram podává informaci o využití dynamického rozsahu. Dále je využíván pro binarizaci obrazu. Pro zvýšení kontrastu se často využívá metoda vyrovnání histogramu.

3.2.2 Detekce hran

Pro percepci lidského vjemu jsou velmi důležitá místa v obraze, ve kterých dochází k náhlým změnám hodnot jasu. Tohoto výsledku dospěli vědci při neurofyziologickém výzkumu, ve kterých prováděli řadu měření na sítnici oka. Tato místa s náhlou změnou jasu odpovídají hranám objektů a jsou důležitým základem pro metody následné segmentace, které se snaží rozpoznat objekty od pozadí. Cílem hranových detektorů je nalézt tyto nespojitosti v obraze [14]. V další části práce bude provedeno testování uvedených hranových detektorů pro detekci vodorovného dopravního značení.

Hranové detektory spojitě obrazové funkce $f(x,y)$ jsou založeny na výpočtu parciálních derivací v každém bodě obrazové funkce. Rovnice č. 4 udává velikost gradientu neboli velikost největší změny růstu jasové funkce a rovnice č. 5 udává směr gradientu.

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Rovnice č. 4

$$\psi = \arg\left(\frac{\partial f}{\partial x}; \frac{\partial f}{\partial y}\right) \quad [\text{rad}]$$

Rovnice č. 5

Další velmi užívanou matematickou operací pro nalezení hran je použit Laplaceův operátor ∇^2 , jenž je založen na druhých parciálních derivacích dle rovnice č. 6, která udává pouze informaci o velikosti změny.

$$\nabla^2(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Rovnice č. 6

Pro číslicové zpracování je potřeba uvedené spojitě vztahy aproximovat pomocí diferencí $\Delta_x f[x,y]$, $\Delta_y f[x,y]$

$$\Delta_x f[x,y] = f[x,y] - f[x-n,y],$$

$$\Delta_y f[x,y] = f[x,y] - f[x,y-n],$$

kde n je z množiny celých čísel. Pak lze gradient definovat rovnicí č. 7.

$$\nabla f[x,y] = \sqrt{(\Delta_x f[x,y])^2 + (\Delta_y f[x,y])^2}$$

Rovnice č. 7

Z hlediska implementace a požadavků na vyšší rychlost výpočtu se v uvedené rovnici nahrazují operace mocniny, odmocnin operací absolutní hodnoty. Výsledná rovnice pro výpočet gradientu je dána rovnicí č. 8.

$$\nabla f[x, y] = |f[x, y] - f[x - n, y]| + |f[x, y] - f[x, y - n]|$$

Rovnice č. 8

Z aproximačního vztahu gradientu je zřejmé, že se již nejedná o bodový operátor, ale operátor, který k výpočtu nové hodnoty v místě tzv. reprezentativního pixelu $f[x, y]$ uvažuje i okolní body. Velikost okolí, která je uvažována, je dána proměnnou n . Rovnici č. 8 lze vyjádřit i použitím tzv. masky. Masky pro $n = 1$ je znázorněná na obr. č. 9.

1	0
0	-1

0	1
-1	0

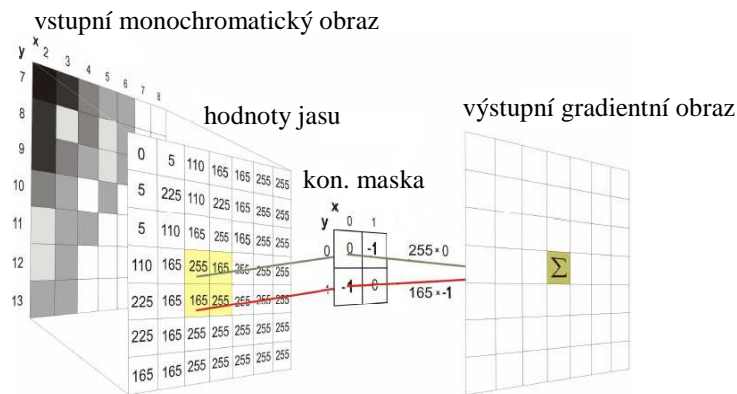
Obr. č. 9 – Masky pro okolí velikosti 2x2 obrazové body

Výpočet je pak realizován tak, že hodnota gradientu $\nabla f[x, y]$ je dána lineární kombinací hodnot kolem reprezentativního pixelu $f[x, y]$ s jednotlivými hodnotami masek a následným sečtením. Lineární kombinace hodnot okolí reprezentativního pixelu s hodnotami masky lze obecně zapsat rovnicí č. 9

$$\nabla f[x, y] = \sum_{i=0}^n \sum_{j=0}^n f[x, y] \cdot h[x - i, y - j]$$

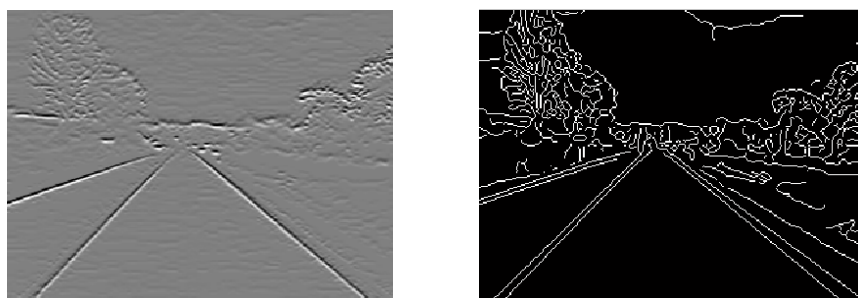
Rovnice č. 9

Rovnice vyjadřuje diskrétní konvoluci, kde h představuje tzv. konvoluční jádro, označované jako konvoluční maska. Konvoluční maska znázorněná na obr. č. 9 představuje gradientní operátor pro okolí 2x2. Aplikaci konvoluční masky na celý obraz zobrazuje obr. č. 10.



Obr. č. 10 – Aplikace konvoluční masky na obraz [37]

Výsledkem aplikace konvoluční masky na celý obraz je gradientní obraz, který má stejnou velikost i stejný rozsah jasových úrovní, zobrazen na obr. č. 11. Tento gradientní obraz je pak prahován, hodnota prahu je stanovena pomocí algoritmu Otsu, který vypočítá ideální práh, algoritmus je podrobně popsán v [22]. Výsledkem prahování je binární obraz obsahující místa s největší změnou jasu, tedy obsahuje jen hrany v obraze obr. č. 11.



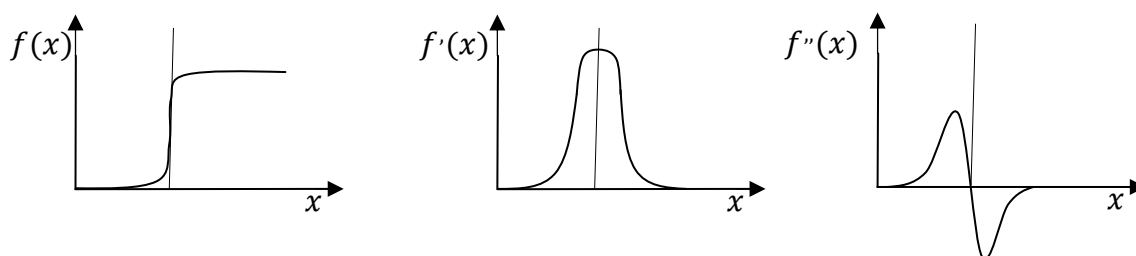
Obr. č. 11 – Gradientní obraz a binární obraz

Masky zobrazené na obr. č. 9 představují Robertsův operátor. Pro aproximaci první derivace se často využívají i další operátory, které velikost gradientu odhadují z okolí o velikosti 3×3 obrazových bodů, tedy je možné pomocí nich zachytit směr hrany v 8 směrech. Větší velikost okolí má výhodu, že tyto operátory jsou o něco více imunní vůči šumu, nežli Robertsův operátor. Mezi tyto operátory patří operátor Prewittové, Sobelův, Robinsonův, Kirschův. Rozdíl mezi těmito operátory je v jednotlivých koeficientech v maskách, jednotlivé masky jsou zobrazeny na obr. č. 12.

Prewittov			Sobelův			Robinsonův			Kirschův		
1	1	1	1	2	1	-1	1	1	-5	3	3
0	0	0	0	0	0	-1	-2	1	-5	0	3
-1	-1	-1	-1	-2	-1	-1	1	1	-1	3	3

Obr. č. 12 - Masky aproximující první derivaci

Základní nevýhodou výše uvedených operátorů aproximujících první derivaci je jejich velká citlivost na přítomnost šumu v obraze [13]. Proto se často využívá průchodu druhé derivace nulou (ang. zero crossing), který odpovídá místu, ve kterém se nachází hrana. Obrázek č. 13 zobrazuje řádkový profil jasu.



Obr. č. 13 – Průběh v řádku: hodnot jasu, první derivace, druhé derivace

Operátor LOG (ang. Laplacian of Gaussian) využívá průchodu druhé derivace nulou, ale před její aplikací provádí konvoluci obrazu s maskou, jejíž koeficienty odpovídají 2D gausovskému rozložení, dle rovnice č. 10

$$G(x, y) = e^{\frac{-x^2+y^2}{2\sigma^2}},$$

Rovnice č. 10

kde x, y jsou souřadnice v obraze, σ je středně kvadratická odchylka. Tato maska představuje vyhlazující filtr, který provede rozmazání obrazové funkce $f(x, y)$. Matematický zápis operátoru LOG, který nejprve provádí konvoluci obrazové funkce s Gaussovským filtrem a následně aplikuje Laplaceův ∇^2 je zapsán rovnicí č. 11.

$$\nabla^2(G(x, y) * f(x, y))$$

Rovnice č. 10

Využitím vlastností linearit operací derivace a konvoluce lze úpravami blíže popsány v [14] získat rovnici č. 11, která umožňuje vypočítat konkrétní hodnoty v konvoluční masce.

$$h[x, y] = c \cdot \left(\frac{x^2 + y^2 - \sigma^2}{\sigma^4} \right) \cdot e^{\frac{-x^2+y^2}{2\sigma^2}}$$

Rovnice č. 10

Obrázek č. 14 zobrazuje konvoluční masku o velikosti 5x5, která aproximuje operátor LOG.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Obr. č. 14 – Masky aproximující první derivaci

LOG operátor je odolnější oproti předchozím operátorům vůči šumu. Mezi jeho nevýhody patří vyhlazování ostrých tvarů, dále spojování hran do uzavřených křivek.

Poslední operátor, který zde bude uveden, se označuje jako Cannyho hranový detektor. Základem detektoru je průchod druhé derivace nulou. Detektor je optimálně navržený na základě tří kritérií, které stanovují následující požadavky: při detekci by nemělo docházet k vícenásobné detekci hrany či přehlédnutí hran, dále rozdíl mezi hranou ve vstupním obraze a detekovanou pozicí má být minimální, poslední kritérium je stejné jako první, ale platí pro zašuměné a nehladké hrany. Podrobný popis Cannyho detektoru

lze nalézt v práci [7], jejímž autorem je John Canny, po kterém je pojmenován detektor. Výsledkem detektoru je informace o směru i velikosti hrany.

3.2.3 Matematická morfologie

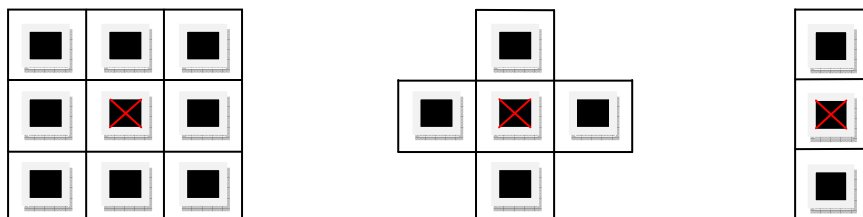
Matematická morfologie využívána v oblasti číslicového zpracování obrazu především při předzpracování obrazu, k zvýraznění požadovaných rysů v obraze a v dalších úlohách. Matematická morfologie je založená na teorii množin, objekt v obraze reprezentuje pomocí množiny v Euklidovském prostoru. V této části bude uvažován jen diskretizovaný prostor Z^2 , který reprezentuje binární obraz. Důvodem je, že v této práci bude využita matematická morfologie především na binární obraz, který je výstupem hranového detektoru. Snahou operací bude zvýraznit detekované hrady jízdních pruhů.

Binární obraz je vyjádřen diskretní bodovou množinou X . Hodnoty množiny X rovné 1 představují v obraze body objektu. Doplněk množiny X^c s hodnotami rovné 0 představuje pozadí v obraze. Rovnice č. 11 reprezentuje zápis binárního obrazu pomocí bodové množiny.

$$X = \{(1,1), (2,0), (2,1), (2,2), (3,1)\}$$

Rovnice č. 11

Transformace T jsou prováděny pomocí relací mezi bodovou množinou a tzv. strukturním elementem, který je reprezentantem menší bodové množiny S . Výsledek relace mezi binárním obrazem a strukturním elementem je hodnota rovna 1 či 0. Obrázek č. 15 zobrazuje typické představitele strukturních elementů, kde červený křížek představuje počátek souřadnic.



Obr. č. 15 – Strukturní elementy

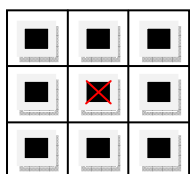
Dále budou uvedeny základní transformace matematické morfologie, které budou využity v této práci. Pro značení transformací bude využito často používaného Minkowského formalismu [14], [30].

Dilatace provádí skládání množiny X a množiny S pomocí vektorového součtu, definována rovnicí č. 12

$$X \oplus S = \{p \in Z^2 : p = x + s, x \in X, s \in S\}.$$

Rovnice č. 12

Dilatace je využívána při zvětšování objektu v binárním obraze. Aplikace dilatace na reálný obraz je zobrazena na obr. č. 16.



Obr. č. 17 – Použitý strukturní element



Obr. č. 16 – Binární obraz po aplikaci dilatace

Eroze je označována jako duální operace k dilataci, provádí složení dvou množin dle rovnice č. 13

$$X \ominus S = \{p \in Z^2 : p + s \in X \text{ pro každé } s \in S\}$$

Rovnice č. 13

Eroze nalézá uplatnění při zjednodušování objektů, např. při získávání obrysů.

Otevření a uzavření využívají předešlých transformací. Otevření je definováno jako eroze následovaná dilatací matematicky vyjádřeno rovnicí č. 14 a naopak uzavření je definováno jako dilatace následovaná erozí matematicky zapsáno rovnicí č. 15.

$$X \circ S = (X \ominus S) \oplus S$$

Rovnice č. 14

$$X \bullet S = (X \oplus S) \ominus S$$

Rovnice č. 15

Využití otevření a uzavření je při odstraňování detailů z obrazu. Tyto transformace mají důležité vlastnosti oproti předchozím: zachovávají monotónnost a tzv. idempotentnost, která říká, že opakováním těchto operací se nemění tvar objektu v obraze. Matematicky je vlastnost idempotentnost vyjádřena rovnicemi č. 16, č. 17.

$$X \circ S = (X \circ S) \circ S$$

Rovnice č. 16

$$X \bullet S = (X \bullet S) \bullet S$$

Rovnice č. 17

3.3 Segmentace obrazu

V předešlé části bylo uvedeno zjištění neurofyziologů, kteří uvádějí, že při vnímání reálného světa jsou pro člověka v obraze nejdůležitější místa, ve kterých dochází ke změnám jasu. Tato skutečnost platí i pro segmentační techniky, protože objekt může být v obraze detekován jen tehdy, liší-li se svou jasovou úrovní od pozadí. Segmentace obrazu představuje proces, jehož cílem je rozklad obrazu do oblastí, které odpovídají objektům ve snímané scéně. V této části budou uvedeny segmentační metody: metoda vycházející z detekce hran, metoda orientovaná na regiony a metoda barevné segmentace.

3.3.1 Metoda vycházející z detekce hran

Pro dále uvedené segmentační metody bude základem výstup z výše již popsanych metod předzpracování obrazu, které prováděly detekci hran v obraze. Část hranové detekce byla podrobněji popsána, protože následné segmentační metody jsou tím účinnější, čím jsou hrany přesněji detekovány.

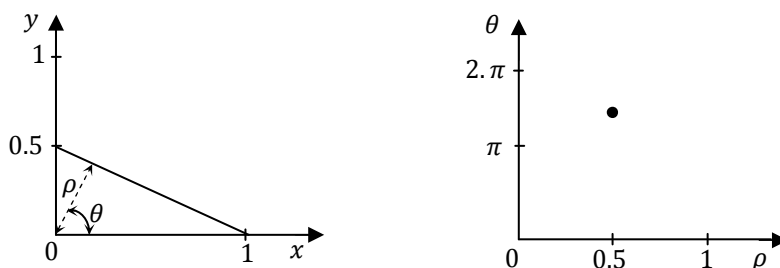
Houghova transformace dále již jen HT představuje metodu, která je využívána pro detekci analytických křivek: přímek, kružnic, elipsy či paraboly, tedy křivek, jež je možné analyticky vyjádřit. Transformace provádí převod křivek z kartézských souřadnic do jednoho bodu v parametrickém prostoru. Dále bude uveden popis HT s cílem detekovat přímku v obraze, z důvodu využití apriorní znalosti o objektu v obraze před zahájením procesu segmentace ze zadání této diplomové práce.

Normálová rovnice přímky je matematicky zapsána rovnicí č. 18 a zobrazená na obr. č. 18

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta),$$

Rovnice č. 18

kde x, y jsou souřadnice, ρ je normálová vzdálenost přímky od počátku, θ je úhel mezi normálou a osou x .

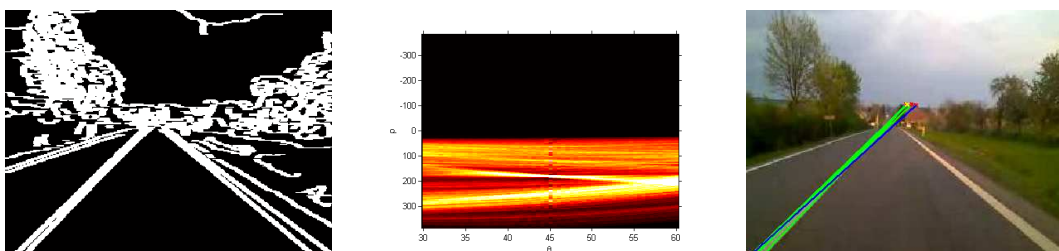


Obr. č. 18 – Zobrazení přímky v kartézských souřadnicích a v parametrickém prostoru

Parametry normálové rovnice přímky ρ a θ vytvářejí parametrický prostor, který je v odborné literatuře označován jako akumulátor [17]. Parametr θ může nabývat hodnot v rozsahu $(0 \leq \theta \leq 2\pi)$, parametr ρ nabývá jen kladných hodnot a jeho velikost je omezená rozměrem vstupního obrazu.

Algoritmus nalezení přímky v obraze pomocí HT lze popsat v následujících krocích:

- 1) Vstupem do HT je binární obraz, v kterém body o hodnotě rovné 1 představují hrany objektů.
- 2) Dále je vytvořen akumulátor (např. pomocí matice) o rozměru parametrů θ a ρ . Např. pro parametr $\theta \in (0 \leq \theta \leq 2\pi)$, kde každému úhlu od 0 do 360° odpovídá jeden bod v matici a parametru ρ odpovídá velikost vstupního obrazu (255 x 255), je rozměr akumulátoru rovna matici o velikosti (360 x 255) bodů.
- 3) Následně jsou do rovnice č. 18 dosazovány souřadnice bodů odpovídajících hodnotě rovné 1 v binárním obrazu. Rovnice je pak řešená pro všechny hodnoty parametrů θ a ρ . Často je při řešení parametr θ inkrementován po jednotlivých úhlech a parametr ρ je dopočítán.
- 4) Při výpočtu jsou v akumulátoru ukládány příspěvky od jednotlivých bodů v binárním obrazu. Po ukončení algoritmu je na obr. č. 19 zobrazen obsah akumulátoru, který obsahuje všechny příspěvky.



Obr. č. 19 – Proces detekce jízdního pruhu využitím HT

- 5) Pro nalezení přímky v obraze je pak v akumulátoru vyhledán bod, který odpovídá maximální hodnotě. Tento bod je určen souřadnicemi (ρ, θ) , pomocí kterých je jednoznačně definována přímka, v binárním obraze. Parametrický prostor může být využit i pro nalezení více přímek o požadované délce či o požadovaném úhlu. Při takovémto vyhledání je nutné uvažovat interval hodnot akumulátoru, které představují požadovanou délku nebo interval hodnot parametru θ , který představuje úhly přímek. Situace zobrazena na obr. č. 19.

3.3.2 Metoda orientovaná na regiony

Cílem metod, které jsou orientované na regiony je rozdělit obraz do souvislých homogenních oblastí na základě stanoveného kritéria homogenity, které může být představováno jasem, barvou, texturou či jinými vlastnostmi obrazu. Výsledky metod orientovaných na regiony se nemusí rovnat výsledkům dosažených pomocí segmentace založené na detekci hran [13]. Dále z těchto metod bude popsána metoda Watershed, kterou je možné využít při detekci vodorovného dopravního značení a byla uvažována při návrhu systému.

Watershed metoda neboli metoda označovaná jako zaplavování oblastí je podrobně popsána v [4] a uvedená v [16]. Tato metoda chápe obraz jako terén. Výšku terénu představují hodnoty jasu, nejnižší výšku terénu mají oblasti s černou barvou, nejvyšší výšku mají oblasti s bílou barvou. Obrázek č. 20 zobrazuje využití metody watershed při detekci jízdního pruhu.



Obr. č. 21 – Binární obraz



Obr. č. 20 – Detekce jízdního pruhu

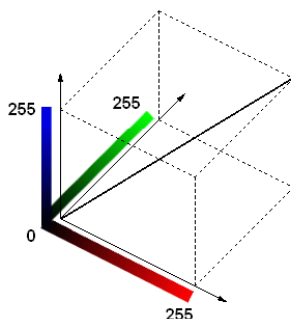
Metoda je založená na procesu zaplavování terénu vodou. Algoritmus lze popsat v následujících bodech:

- 1) Vstupem je monochromatický obraz nebo gradientní obraz jako výstup z hranového detektoru před prahováním, zobrazuje obr. č. 21.
- 2) Algoritmus nejprve nalezne místo s nejmenší výškou terénu (povodí), neboli minimum v obraze (místo s černou barvou na obr. č. 21).
- 3) V další kroku algoritmus začíná zaplavovat místa s minimální výškou, hladina se tedy rozlévá do okolí. V okamžiku, kdy by mělo dojít ke slítí vody ze dvou povodí, dochází k vytvoření hráze.
- 4) Takto je postupně zaplavován celý obraz, až do okamžiku, kdy hladina dosáhne maximální hodnoty.
- 5) Výsledkem je obraz, který je rozdělen do několika oblastí, které odpovídají jednotlivým povodím, které jsou odděleny hrázemi. Jednotlivým bodům rozdělených oblastí jsou přiřazeny stejné indexy, kterými se mezi sebou odlišují. Na základě těchto indexu lze provést následné prahování, kterým je možné při aplikaci této metody při rozpoznávání vodorovného dopravního značení detekovat jízdní pruh.

3.3.3 Metoda barevné segmentace

V této části bude uvedena metoda, která pro nalezení objektu v obraze využívá barevné reprezentace obrazu. Nejprve budou popsány barevné modely, které budou využity při segmentaci.

RGB (Red, Green, Blue) barevný model, který byl standardizován v roce 1931 komisí CIE (Commission International de l'Eclairage), je často využíván při snímání a zobrazování obrazu. Barevný obraz dle tohoto modelu je složen aditivně ze tří složek. Jednotlivými barevnými složkami jsou červená o vlnové délce 700nm , zelená o vlnové délce $546,1\text{nm}$ a modrá o vlnové délce $435,8\text{nm}$. RGB model zobrazuje obr. č. 22 pomocí krychle [10].



Obr. č. 22 – Znázornění RGB modelu pomocí krychle [38]

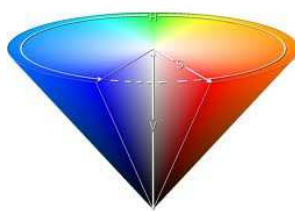
Jak již bylo výše řečeno, tento model je více využíván pro snímání či reprezentaci, protože byl navržen tak, aby co nejvíce vyhovoval spektrální citlivosti lidského oka. Pro účely segmentace v číslicovém zpracování obrazu je zásadním problémem tohoto modelu korelovanost dat v jednotlivých barevných složkách, protože každá barevná složka nese informace o barvě a dále přispívá určitou částí jasové složce. Hodnota jasové složky je vyjádřena rovnicí č. 19

$$Y = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B,$$

Rovnice č. 18

kde Y je jas, který udává intenzitu světla, R, G, B jsou jednotlivé barevné složky.

HSV představuje barevný model, který je složen ze tří složek. Jednotlivé složky tohoto modelu jsou označovány: H (Hue), složka nesoucí informaci o barevném tónu, dále S (Saturation), složka udávající sytost barvy a poslední V (Value), jasová složka. Výhodou HSV modelu oproti RGB je, nekorelovanou složek H, S, V. Této vlastnosti lze využít při barevné segmentaci. Přejít mezi modely RGB a HSV je možný pomocí algoritmu uvedeného v [24]. Na obr. č. 23 je zobrazena kónická reprezentace HSV modelu,



Obr. č. 23 – Kónická reprezentace HSV modelu [24]

kde barevný tón H vyjádřený v rozsahu úhlů $\langle 0, 360^\circ \rangle$ udává převládající spektrální barvu, sytost S vyjádřená v rozsahu $\langle 0, 1 \rangle$ určuje příměs ostatních barev, jasová hodnota V vyjádřená v rozsahu $\langle 0, 1 \rangle$ udává množství bezbarvého světla.

Barevnou segmentaci je pak možné provádět pomocí prahování v jednotlivých kanálech barevných modelů.

Postup lze popsat v následujících krocích:

- 1) Proveďte se výřez oblasti v barevné složce, která má být detekována.
- 2) Dále je odhadnuto rozdělení hodnot v daném výřezu.
- 3) Pro Gaussovské rozdělení je vypočítána střední hodnota a směrodatná odchylka.
- 4) Nastavení hodnot prahů P_{max} , P_{min} , které budou určovat interval prahování. Hodnoty P_{max} , P_{min} jsou určeny na základě výpočtu parametrů daného výřezu.
- 5) Proveďte se prahování celého obrazu v dané barevné složce dle nastavených kritérií.

Metoda byla uvažována při návrhu vlastního algoritmu detekce vodorovného dopravního značení. Následující obrázek zobrazuje výsledky prahování v modelu HSV, při použití této metody při detekci vozovky. Barevná segmentace je nejčastěji prováděna ve složkách H , S . Nepřítomnost složky V odstraní závislost na vlivu světelných podmínek.

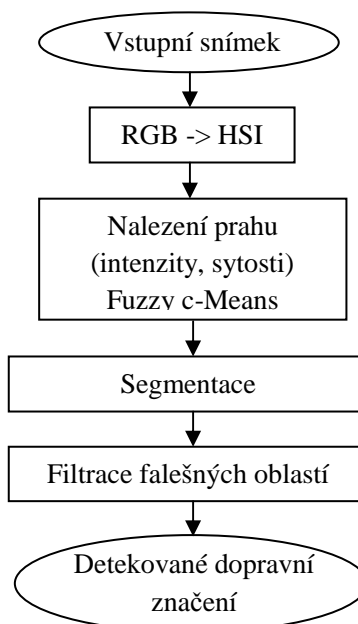


Obr. č. 24 – Detekce vozovky v modelu HSV

3.4 Přehled stávajících algoritmů pro rozpoznávání vod. dop. značení

V této části bude uveden přehled přístupů k detekci vodorovného dopravního značení. Tento přehled současných algoritmů se pak stane spolu s celou předešlou částí základem při návrhu vlastního řešení.

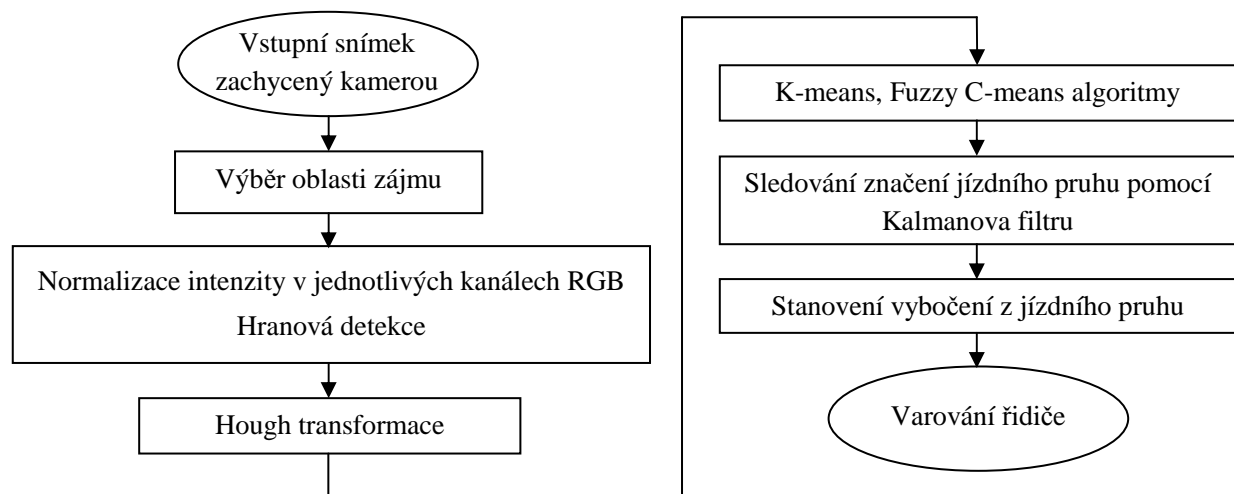
- Metoda z roku 1999 uvedená v [18] označovaná jako LANA je založená na DCT (Diskrétní kosinova transformace). Autoři se snaží ve frekvenční doméně oddělit diagonálně dominantní hrany, které odpovídají ve snímaném obraze jízdním pruhům od hran, které jsou orientovány náhodně. Pro nalezení jízdního pruhu je využito MAP (Maximum Aposteriori estimate), podrobný popis lze nalézt v [18].
- Metoda popsaná v [32] navrhuje detekci vodorovného dopravního značení využitím HSI (odstín, sytost, intenzita) barevného modelu. Metoda je založená na prahování. Hodnota prahu je nastavena tak, aby v obraze zůstaly jen jízdní pruhy. Algoritmus lze znázornit pomocí následujícího blokového schématu:



Obr. č. 25 – Blokové schéma algoritmu LDW

- Vstupní obraz z kamery je převeden do HSI barevného modelu
- Dále je definována oblast zájmu, pro další zpracování.
- Pro nalezení hodnot prahu intenzity a sytosti je využito FUZZY c-means algoritmu [32]

- Následují kroky segmentace pomocí metody prahování a filtrace pro konečné nalezení jízdních pruhů.
- Metoda uvedená v práci [8] je novým navrhovaným řešením v době řešení této práce. Autoři se snaží navrhnout systém LDW v souladu s doporučení z ISO 17361:2007 a FMCSA-MCRR-05-005. V práci je uveden navržený algoritmus detekce jízdních pruhů, který lze reprezentovat následujícím blokovým schématem.



Obr. č. 26 – Blokové schéma algoritmu LDW

Bloky posledního uvedeného algoritmu: výběr oblasti zájmu, detekce hran, Hough transformace jsou využívány v mnoha dalších algoritmech [1], [26], [27], [34], řešících detekci vodorovného dopravního značení.

- Metoda uvedená v [3] je navržena pro aplikace, které mají pracovat v reálném čase. Metoda je založena na morfologických operacích a segmentační metodě watershed. Postup algoritmu je následující: na vstupní obrazový snímek je aplikován morfologický filtr, který má za cíl odstranit stíny z vozovky a spojit přerušované dopravní značení. Takto předzpracovaný obraz je následně pomocí metody watershed segmentován. Výsledkem segmentace jsou detekované jízdní pruhy a oblast vozovky. V následující části algoritmus vypočítává geometrický model vozovky. Tento model je poté stále aktualizován.

4 Pořizování podkladových obrazových dat

Velmi důležitou částí práce bylo pořizování obrazových dat. V průběhu pořizování obrazové databáze byl vyřešen problém s umístěním kamery na vozidle, při kterém byl uvažován vliv geometrického zkreslení, jež by ovlivnil výpočet navrženého algoritmu. Při pořizování obrazových dat bylo využito dvou webkamer Genius Face Cam 1320 a Hercules webcam classic.

4.1 Specifikace snímacích zařízení

Pro záznam dat byly využity dvě stávající levné standardní webkamery.

Genius Face Cam 1320 představuje klasickou webkameru jejím jádrem je CMOS senzor s rozlišením až 1,3 megapixelů. V následujících bodech je uvedena její specifikace:

- Rozlišení fotografie až 1280 x 1024 (Software interpolation)
 - 320 x 240 pixel – 30fps
 - 640 x 480 pixel – 30 fps
 - 800 x 600 pixel – 9 fps
 - 1280 x 1024 pixel – 9 fps
- Manuální ostření
- Vícevrstvá čočka
- Formáty souborů: JPEG/WMV
- USB 2.0



Obr. č. 27 – Genius Face CAM 1320 [35]

Hercules webcam classic dosahuje maximálního rozlišení 1280 x 960 pixel, její základ tvoří CCD senzor.

- Frame rate až 30fps
- Manuální ostření
- USB 2.0



Obr. č. 28 – Hercules webcam classic [36]

4.2 Umístění kamery

Při pořizování obrazových dat byla rychlost vozidla v rozsahu 80 – 120 km/h. Kamery byly umístěny ve třech pozicích. Při prvních experimentech byla kamera umístěna v čelní masce vozidla. Poté byla připevněna na standardní místo u systému LDW. Poslední pozice, z které byla snímána scéna před vozidlem z příčnicku na střeše automobilu. Situaci znázorňují následné obrázky.



Obr. č. 30 – Umístění kamery



Obr. č. 29 – Umístění kamery na příčnku

Scéna před vozidlem byla snímána z obou kamer rychlostí 30 fps s rozlišením 640 x 480 pixelů. Při provedených měření bylo zaznamenáno 92 Gbit obrazových dat, což představuje 210 minut záznamu scény před automobilem. Byly zaznamenány silniční dopravní úseky s různou kvalitou vodorovného dopravního značení. Po pořizování obrazových dat byla tato data roztržena do několika skupin podle typu zaznamenaného úseku.

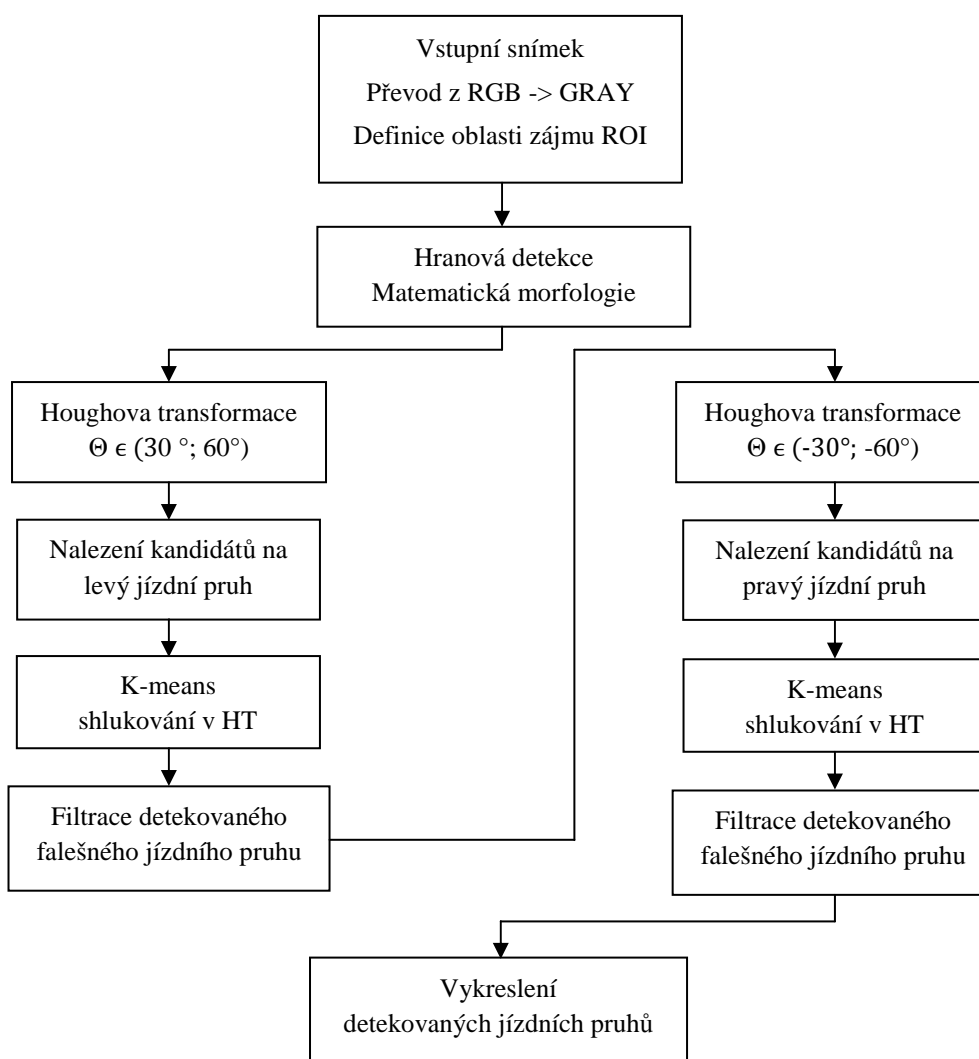
Klasifikace dat:

- Dálnice
- Silnice první třídy
- Silnice druhé třídy

Při návrhu algoritmu a jeho testování byly využity především úseky, které vyhovují doporučením ISO 17361:2007 a FMCSA-MCRR-05-005. Jedná se tedy především o úseky na dálnici s kvalitním dopravním značením. Z uvedených třech pozic je nejlepší umístění kamery na příčnku na střeše vozidla. Při této pozici kamery je dostatečný pohled na vozovku, dále při tomto umístění nevzniká rušení, které je způsobené odrazy od předního skla vozidla.

5 Popis vlastního řešení v prostředí MATLAB

V této kapitole bude uveden návrh vlastního algoritmu detekce vodorovného dopravního značení. Návrh algoritmu vychází z uvedených řešení v předešlé kapitole a je proveden dle doporučení zadání práce v programovém prostředí MATLAB. Při návrhu je velmi výhodné využít Image processing toolbox, který je určen pro zpracování obrazu. Navržený algoritmus je znázorněn na obr. č. 31 pomocí blokového schématu.



Obr. č. 31 – Blokové schéma navrženého algoritmu

Vstupní obraz je nejprve převeden z RGB barevného modelu na monochromatický obraz. Dále je v obrazu ve stupni šedi definována oblast zájmu ROI (Region of Interest). Oblast zájmu je výhodné definovat především z důvodu rychlosti zpracování, protože následující metody nemusí být aplikovány na celý obraz. Obr. č. 32 zobrazuje definovanou oblast zájmu.



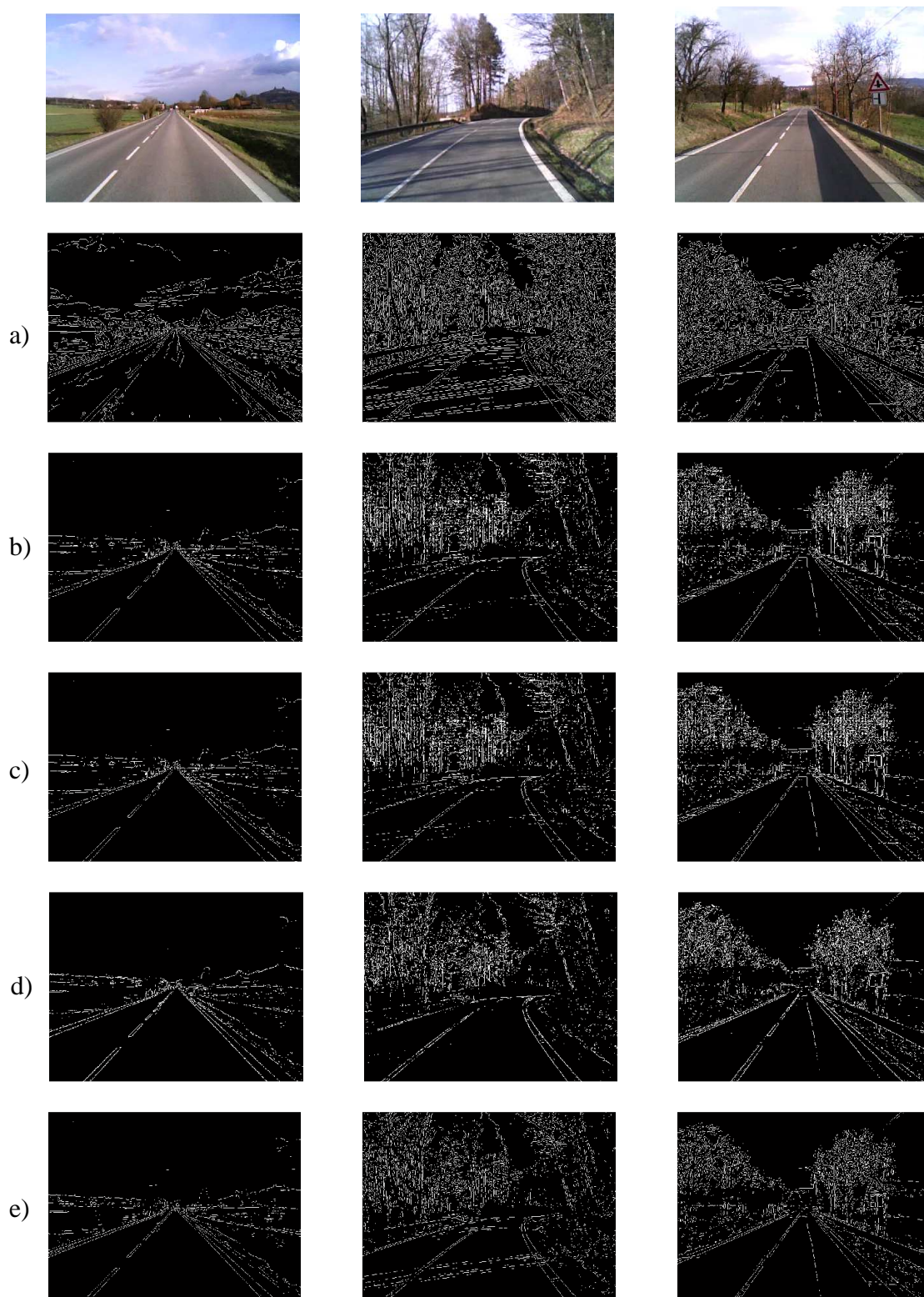
Obr. č. 32 - ROI

Dalším krokem algoritmu je hranová detekce. Při návrhu byly testovány hranové detektory uvedené v části 3.2.2., nastavení hodnot prahů těchto detektorů je provedeno pomocí Otsuva algoritmu.

Pro testování byly vybrány snímky, s kterými je možné se setkat při reálném zpracování obrazu. Snímky jsou zobrazeny na obr. č. 33. První testovací snímek představuje záběr při dobrých světelných podmínkách a při dobrém dopravní značení. Druhý snímek naznačuje problém, který nastává v okamžiku přítomnosti stínů, které jsou způsobeny stromy v okolí vozovky. Třetí snímek zobrazuje situaci, která může způsobit falešně detekovaný pravý okraj vozovky. Stín vytvořený silničními svodidly působí rušivě při aplikaci HT.

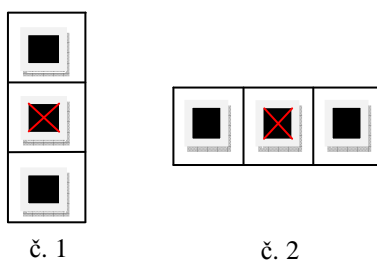
Z testovacích snímků na obr. č. 33 je vidět, že Cannyho detektor se snaží spojit jednotlivé hrany. Dále je vidět, že při takto nastaveném prahu dochází v binárním obraze u Cannyho detektoru k nad segmentaci detekovaných hran. Z těchto důvodů nebude tento detektor použit, protože nad segmentace by způsobovala problém v následné Hough transformaci při detekci vodorovného dopravního značení. Ze skupiny detektorů, které využívají první derivace, nejlepších výsledků dosáhl Robertsův hranový detektor. Z obr. č. 33 je vidět, že i za přítomnosti stínů, které jsou způsobeny stromy, dosáhl dobrých výsledků. Situaci, kterou znázorňuje třetí snímek, nedokázal vyřešit žádný hranový detektor. Tento problém musí být řešen následnou filtrací detekovaných jízdních pruhů, nebo jinými metodami při předzpracování obrazu před segmentací.

Testovací snímky:



Obr. č. 33 – Testování hranových detektorů a) Canny, b) Sobel, c) Prewitt, d) Roberts, e) LOG

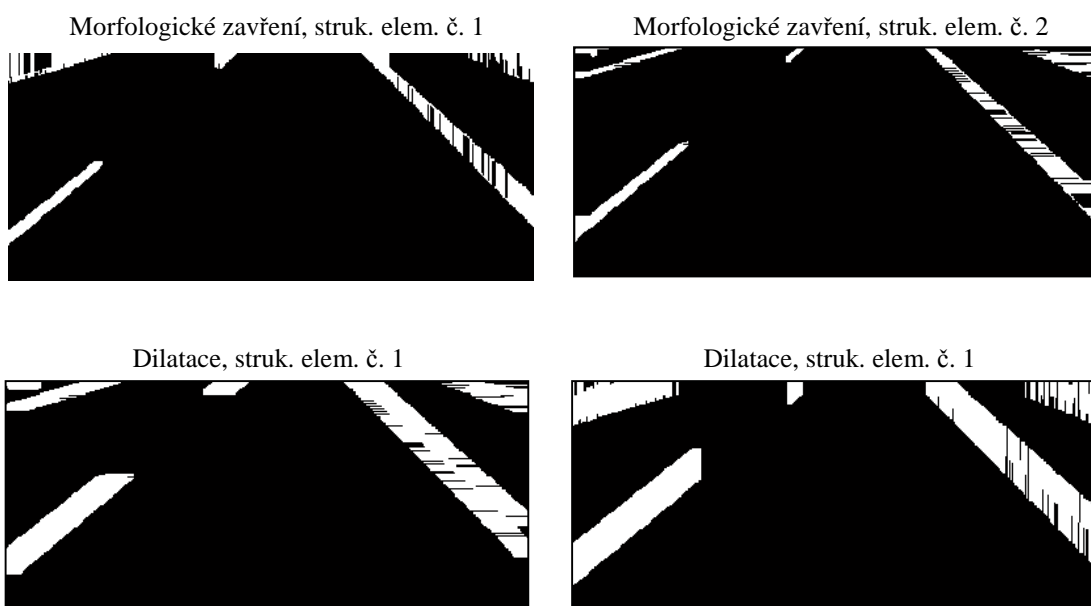
Následující krok po aplikaci hranového detektoru, je matematická morfologie. Cílem matematické morfologie je zvýraznit detekované hrany pro následnou Hough transformaci. Z morfologických transformací je možné pro potřeby zvětšení či zvýraznění objektů v obraze využít transformace dilatace a morfologické uzavření. Dále je možné definovat různé typy strukturních elementů. Následující obr. č. 36 zobrazuje výsledky transformací dilatace a morfologického uzavření při experimentech. Obr. č. 34 informuje o použitých strukturních elementech. Její rozměr byl zvolen na základě velikosti šířky vodorovného dopravního značení. Experimentem ze zaznamenané databáze byla stanovena šířka 20 obrazových bodů.



Obr. č. 34 – Strukturní element



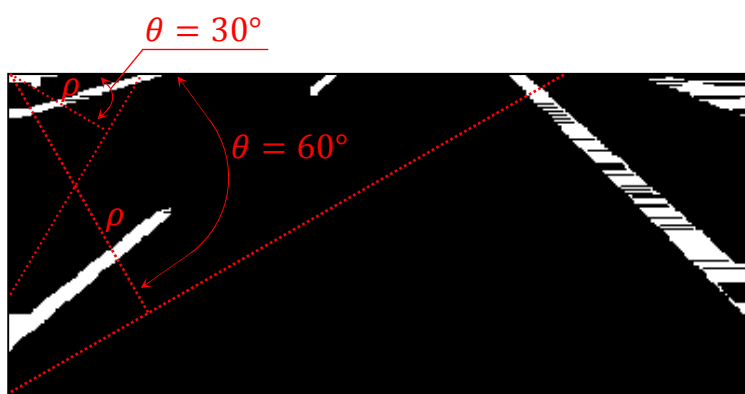
Obr. č. 35 – Binární obraz (detekované hrany)



Obr. č. 36 – Výsledky morfologických transformací

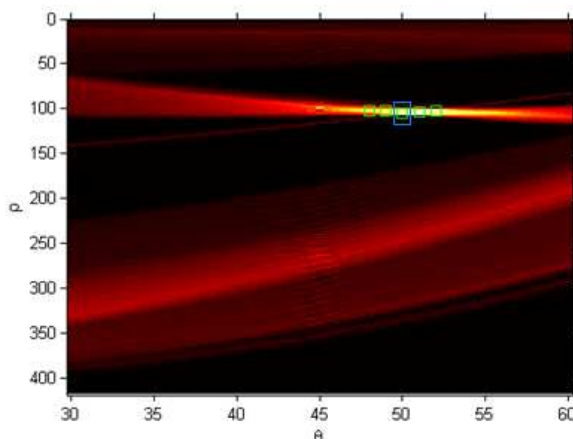
Cílem aplikace matematické morfologie při detekci vodorovného dopravního značení je vyplnit detekované hrany jízdních pruhů. Touto operací bude dosaženo ulehčení úlohy pro následnou segmentaci pomocí Hough transformace (HT). Z obr. č. 36 jsou patrné vlastnosti transformací dilace a morfologického zavření. Využití dilatace není výhodné pro vyplnění (zacelení) z důvodu změny velikosti objektu. Tato skutečnost může ovlivnit odhad umístění jízdního pruhu pomocí HT. Využitím vlastností morfologického zavření, je možné dosáhnout vyplnění detekovaného jízdního pruhu bez změny velikosti objektu. Nejlepším řešením bylo zvolení morfologického zavření využívající strukturní element č. 2. Při této aplikaci transformace jsou jízdní pruhy dostatečně zacelené a je zachována skutečná velikost. Z obrázku je vidět i zachování levého vodorovného dopravního značení oproti aplikaci se strukturním elementem č. 1.

Matematická morfologie byla poslední částí předzpracování obrazu. V následující části algoritmus provede segmentaci pomocí Hough transformace. Z blokového schématu je vidět, že Hough transformace bude aplikována samostatně při detekci levého a pravého jízdního pruhu. Při detekci levého jízdního pruhu se předpokládá na základě uvedených předpokladů, že levý jízdní pruh se může nacházet v rozmezí úhlu $\theta \in (30^\circ; 60^\circ)$. Pro řešení v systému MATLAB je situace znázorněná na obr. č. 37. Parametry θ, ρ mají stejný význam jako na obr. č. 18, jen v systému MATLAB je počátek souřadnic umístěn v levém horním rohu. Červená výseč vymezuje prostor, v kterém se předpokládá levý jízdní pruh. Stejná situace je uvažována při detekci pravého jízdního pruhu, kde rozmezí úhlu je rovno $\theta \in (-60^\circ; -30^\circ)$.



Obr. č. 37 – Význam parametrů HT

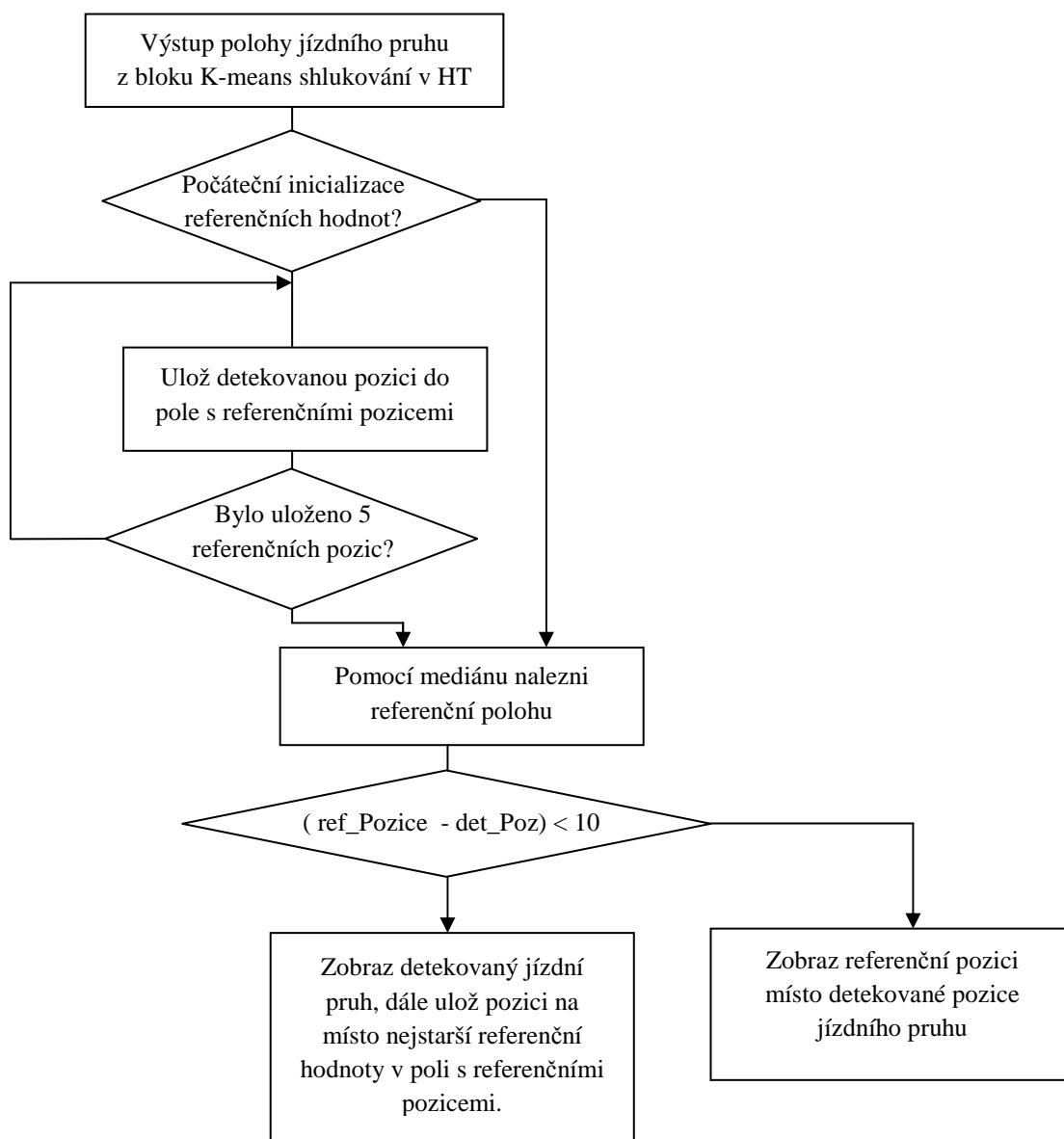
Algoritmus prochází jednotlivé body v binárním obraze a provádí výpočet pro požadované úhly dle rovnice č. 18. Výsledkem je vytvořený parametrický prostor, který je znázorněn na následujícím obrázku č. 38.



Obr. č. 38 – Parametrický prostor

Světlá barva představuje nejvyšší hodnoty. V dalším kroku je vyhledáno 5 nejvyšších hodnot v parametrickém prostoru, v obr. č. 38 jsou zobrazeny zelenými čtverci. Pět nejvyšších hodnot představuje kandidáty na detekovaný levý jízdní pruh. Těchto pět kandidátů vytváří shluk bodů, každý bod představuje v prostoru obrazu jednu přímku, odpovídající detekovanému jízdnímu pruhu. Pomocí k-means algoritmu je nalezen střed tohoto shluku, který odpovídá detekovanému levému jízdnímu pruhu v prostoru obrazu. Střed shluku je zobrazen v obr. č. 38 modrou barvou. Stejná situace se provádí při detekci druhého jízdního pruhu.

Posledním krokem algoritmu je zajištění filtrace falešných detekovaných jízdních pruhů, které mohou být způsobeny různým rušením. Algoritmus, který má za cíl provádět filtraci, zobrazuje blokové schéma na obr. č. 39. Tento algoritmus je aplikován zvlášť na levý i pravý jízdní pruh. Vstupem je detekovaná pozice jízdního pruhu. Z počátku je provedeno uložení pozice detekovaného jízdního pruhu z prvních pěti snímků. Těchto pět pozic slouží pro nastavení referenční hodnoty. Referenční hodnota je dále porovnávána s novou detekovanou pozicí. Jestliže odchylka od referenční pozice není vyšší jak 15 obrazových bodů, je prohlášen detekovaný jízdní pruh za správný a je zobrazen. Dále je tato poloha uložena do referenčního pole místo nejstarší referenční pozice. Takto je prováděna aktualizace referenční hodnoty v průběhu algoritmu. V případě, že je odchylka vyšší, byla provedena špatná detekce a za správnou polohu jízdního pruhu se předpokládá referenční poloha.



Obr. č. 39 – Blokové schéma navrženého algoritmu



Obr. č. 40 – Výstup z navrženého algoritmu v systému MATLAB

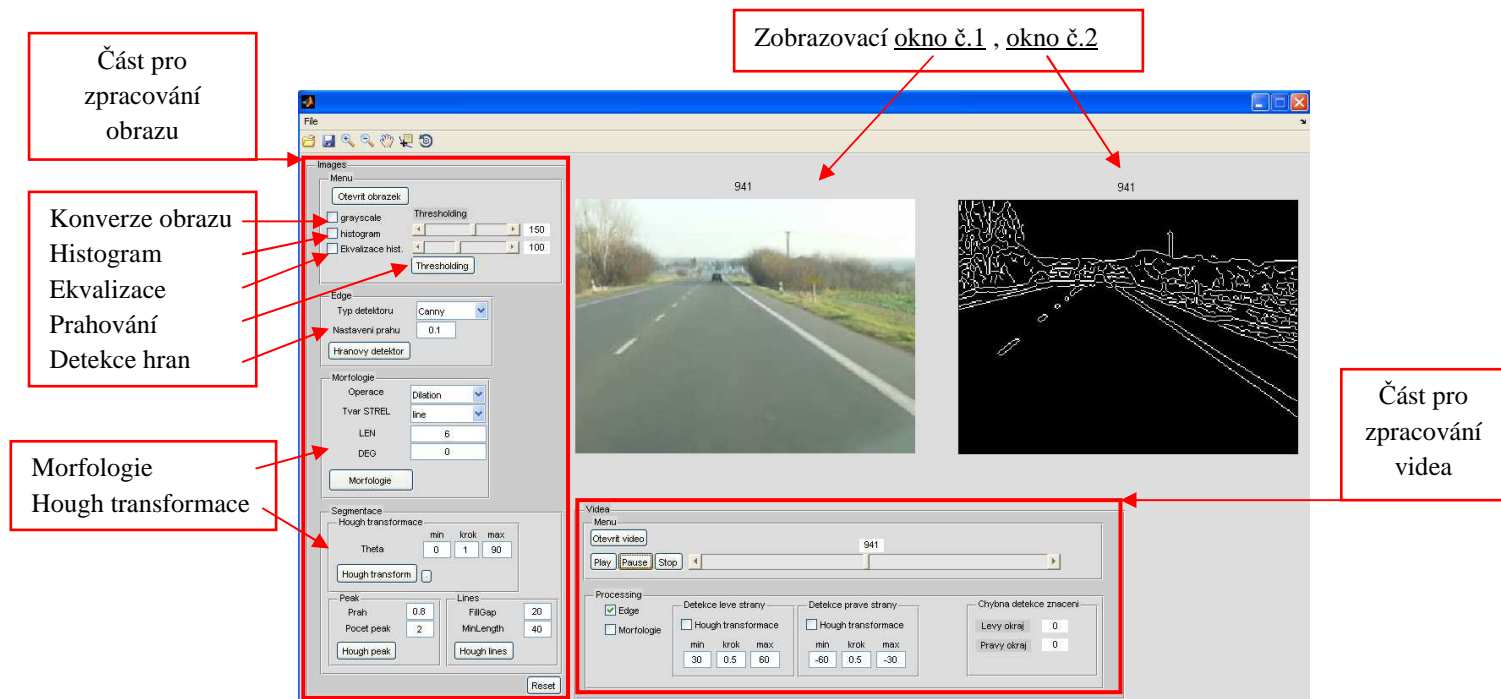
5.1 Interaktivní grafické prostředí

Při návrhu vlastního řešení, bylo vytvořeno interaktivní grafické prostředí, v kterém jsou realizovány jednotlivé algoritmy číslicového zpracování obrazu prostřednictvím ovládacích prvků. Ovládací prvky umožňují modifikovat parametry jednotlivých algoritmu a tím provádět experimenty při návrhu vlastního řešení bez zasahování do zdrojového kódu. Interaktivní prostředí umožňuje urychlit a optimalizovat proces návrh vlastního řešení.

Základní funkce prostředí:

- 1) Načtení dat pro zpracování (obraz, video)
- 2) Metody předzpracování obrazu
 - a. Konverze barevného obrazu na monochromatický obraz
 - b. Konverze obrazu z RGB modelu do HSV modelu
 - c. Zobrazení histogramu
 - d. Ekvalizace histogramu
 - e. Prahování histogramu
 - f. Detekce hran
- 3) Metody zpracování obrazu
 - a. Morfologické operace
 - b. Hough transformace
 - c. Shlukování k-means algoritmus

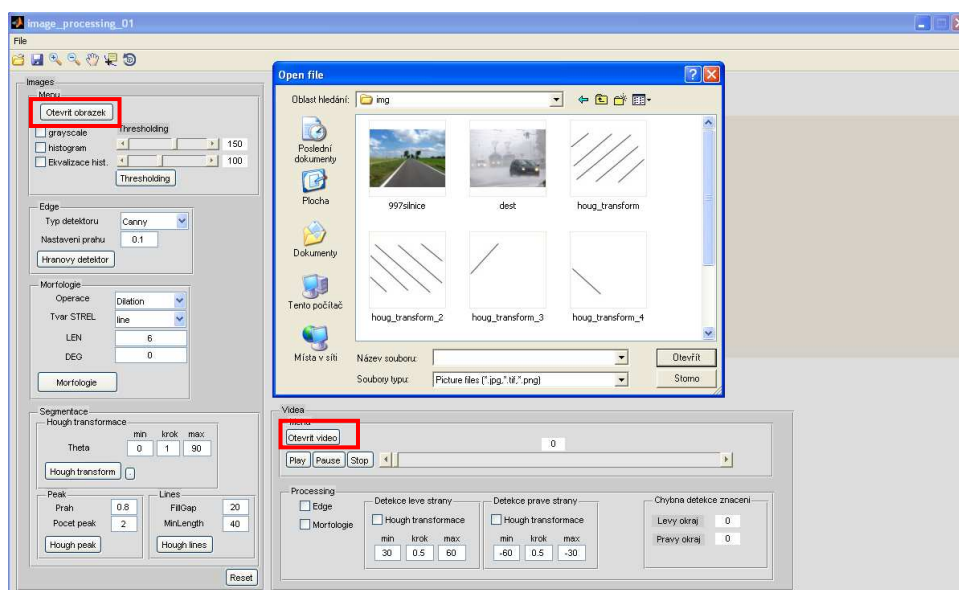
5.1.1 Popis pracovního dialogového okna



Obr. č. 41 – Vytvořené grafické prostředí

5.1.2 Načtení a zobrazení vstupních dat

Aplikace umožňuje načíst obrazová data formátu .jpg, .tif, .png a video soubor formátu .avi. Uživatel může načíst data aplikací tlačítek Otevřít obrázek, Otevřít video. Poté je otevřeno dialogové okno, ve kterém lze vybrat vstupní soubor dat.



Obr. č. 42 – Načtení vstupních dat

Načtená data jsou následně zobrazena v zobrazovacím okně č.1 a jsou připravena pro následnou analýzu.

5.1.3 Práce s obrazovými daty

V následující části budou ukázány postupy práce s aplikací při předzpracování vstupních obrazových dat následnou hranovou detekcí, morfologickými operacemi, houghfovou transformací.

V hlavním dialogovém okně v části zpracování obrazu jsou pro uživatele připraveny základní ovládací prvky pro aplikaci algoritmů na obrazová data.

- Konverze barevného obrazu na monochromatický obraz

Aplikací checkboxu grayscale se provede převod načteného barevného obrazu na monochromatický obraz.



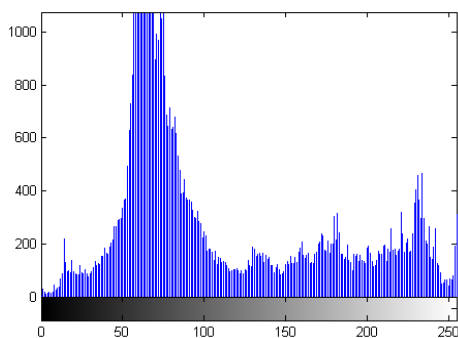
Obr. č. 44 – Vstupní obraz



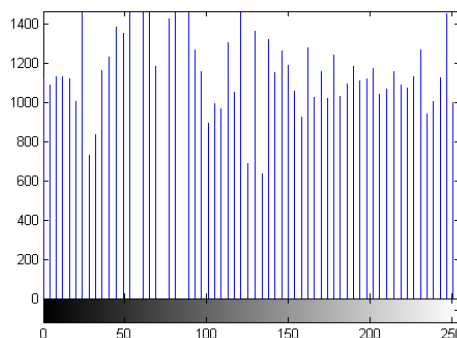
Obr. č. 43 – Monochromatický vstupní

- Zobrazení histogramu

Aplikací checkboxu histogram se v zobrazovacím okně č. 2 zobrazí histogram monochromatického obrazu.



Obr. č. 46 – Histogram šedé stupnice



Obr. č. 45 – Histogram po ekvalizaci obraz

- Ekvalizace histogramu

Aplikací checkboxu ekvalizace hist. je provedena ekvalizace histogramu a můžeme pozorovat, jak se zvýší kontrast obrazu. Ekvalizovaný histogram je zobrazen na obr. č. 47.



Obr. č. 47 – Monochromatický obraz po ekvalizaci histogramu

- Prahování histogramu

Použitím posuvníku je možné nastavit spodní a horní hranici prahu pro prahování histogramu. Změnou polohy posuvníků může uživatel vidět, jak se mění monochromatický obraz a jeho histogram. Aplikací tlačítka Thresholding je následně provedeno prahování histogramu podle nastavených prahů.



Obr. č. 48 – Vliv prahování na monochromatický obraz

- Detekce hran

Menu Edge nabízí uživateli zvolit si typ hranového detektoru (Canny, Sobel, Prewitt, Roberts, Log, Zeroctoss) a nastavit velikost prahu detektoru. Aktivací tlačítka Hranový detektor je aplikován hranový detektor na monochromatický obraz.



Obr. č. 49 – Aplikace hranového detektoru na monochromatický obraz

- Morfologické operace

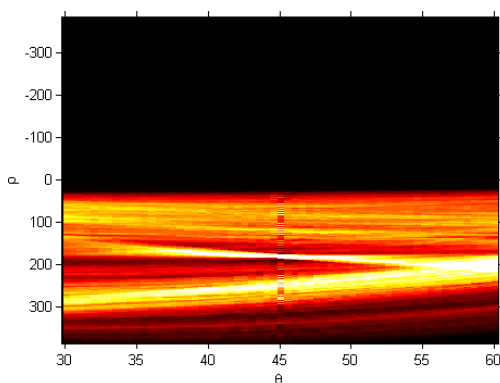
Menu Morfologie umožňuje uživateli jednoduše aplikovat na obraz morfologické operace. Uživatel má možnost si zvolit typ operace (Dilatace, Eroze, Otevření, Zavření, atd.), tvar strukturálního elementu a jeho velikost. Aktivací tlačítka Morfologie je aplikována na obraz morfologická operace.



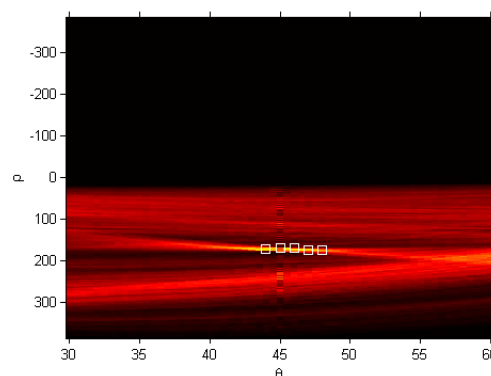
Obr. č. 50- Aplikace morfologické operace dilatace, typ struk. element line na obraz

- Hough transformace

Menu segmentace obsahuje části Hough transformace, Peak, Lines. Pomocí těchto částí je možné detekovat přímku v obraze. V části Hough transformace je možné nastavení parametru úhlu (Theta) hough transformace a následně aktivací tlačítka aplikovat hough transformaci na obraz. Aplikací transformace na obraz se zobrazí v zobrazovacím okně č. 2 hough prostor.



Obr. č. 52 – Parametrický prostor



Obr. č. 51 – Zobrazení detekovaných

V části Peak je možné provést prahování hough prostoru. Aktivací tlačítka Hough peak se provede prahování hough prostoru podle nastavených parametrů Prah a Počet peak. V hough prostoru zobrazeném v zobrazovacím okně č. 2 se zobrazí nalezené hodnoty.

Protože nalezené hodnoty v hough prostoru představují parametry parametrického popisu přímky, je v části Lines implementován algoritmus pro vykreslení těchto přímek ve vstupním obraze. Uživatel má možnost ovlivnit parametry FillGap², MinLength³ a aktivací tlačítka Hough lines se zobrazí detekované přímky na vstupním obraze.



Obr. č. 53 - Detekované přímky pomocí hough transformace pod úhlem 30° - 60°

² Tato hodnota představuje vzdálenost mezi nalezenými přímkami pod stejným úhlem. Přímký budou spojeny, jestliže vzdálenost mezi přímkami bude menší než zadaná hodnota.

³ Tato hodnota představuje délku přímky. Přímký, které budou menší než daná hodnota, nebudou zobrazeny.

6 Realizace systému rozpoznávání jízdního pruhu

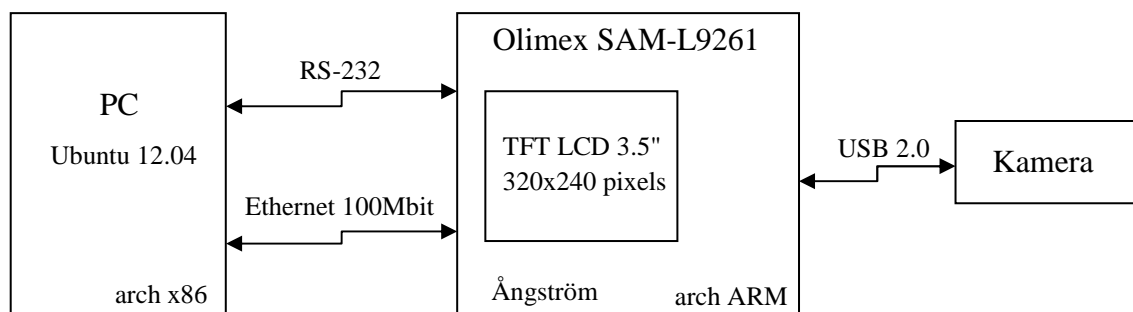
V této kapitole bude popsán návrh konkrétního řešení systému rozpoznávání vodorovného dopravního značení. Z počátku bude uveden použitý hardwarový modul a operační systém. Dále budou následovat jednotlivé části, které obsahují důležité kroky potřebné při návrhu aplikací na operačním systému Linux pro cílovou architekturu ARM. Cílem těchto kapitol je připravit referenční materiál, který by případným dalším zájemcům o vývoj aplikací do vestavěných systémů umožnil rychlé seznámení s touto problematikou.

6.1 Hardwarový modul

Pro realizaci systému rozpoznávání vodorovného dopravního značení, bylo využito stávající vývojové desky Olimex SAM9-L9261. Tato vývojová deska je postavená na 32 bitovém mikroprocesoru AT91SAM9261 s jádrem ARM od výrobce Atmel.

Mezi výhody této vývojové desky při realizaci systému rozpoznávání vodorovného dopravního značení patří zejména přítomnost 3.5 palcového dotykového barevného TFT LCD displeje s rozlišením 320x240 pixelů, který bude využit pro zobrazení scény před vozidlem a bude informovat o detekovaném vodorovném dopravním značení. Dále vývojová deska obsahuje USB host port, který umožňuje připojit různá externí zařízení. V této práci bude především USB host port využit pro připojení kamery. Mezi další využití rozhraní patří RS-232 a Ethernet, které budou využity pro přístup k vestavěnému zařízení a přenosu aplikace do vestavěného zařízení. Další pro tuto práci důležitou výhodou je, přítomnost jednotky pro správu pamětí MMU (Memory Management Unit) v mikroprocesoru AT91SAM9261, která umožňuje běh operačních systémů Linux či WindowsCE.

Při realizaci systému bude využit od výrobce připravený operační systém, který je založený na Linuxové distribuci Ångström. V předešlé kapitole č. 4 bylo uvedeno, že snímací zařízení představuje webkamera Genius Face Cam 1320. Následující obr. č. 54 zobrazuje blokové schéma hardwarového modulu realizující systém rozpoznávání vodorovného dopravního značení a obr. č. 55 zobrazuje použitou vývojovou desku Olimex SAM9-L9261.



Obr. č. 54 – Blokové schéma hardwarového modulu



Obr. č. 55 – Vývojová deska Olimex SAM-L9261

6.2 Operační systém GNU/Linux

V předešlé části již bylo předesláno, že pro realizaci systému rozpoznávání vodorovného dopravního značení bude využito operačního systému Linux, který je již připraven na vývojové desce. Hlavní výhodou použití operačního systému Linux při realizaci systému rozpoznávání vodorovného značení je v možnosti využití již vytvořených modulů (ovladačů jádra), které zajišťují přístup k perifériím či open-source knihoven (OpenCV, GTK+), které zjednodušují vytváření vlastních aplikací. Tyto výhody jsou vykoupeny cenou snížením výpočetního výkonu, který je potřebný pro zajištění běhu operačního systému. I přes nevýhodu částečného snížení výkonu bude při realizaci operační systém využit, a proto v následné části budou uvedeny základní informace o operačním systémem GNU/Linux.

Linux je jádro operačního systému, které vytvořil v roce 1991 Linux Torvalds při svém studiu na univerzitě v Helsinkách. Pod zkratkou GNU je pak skryto mnoho dalšího vytvořeného softwaru, který společně s Linuxovým jádrem vytváří plnohodnotný operační systém neboli tzv. Linuxovou distribuci. V současné době existuje mnoho distribucí nejen pro osobní počítače s 32/64bitovými procesory např. Red Hat, Debian, Ubuntu, ale i pro jiné architektury procesorů, např. ARM (distribuce Ångström, µLinux), PowerPC, MIPS [20].

V diplomové práci bude pro vývoj aplikací pro cílovou architekturu ARM využita distribuce Ubuntu 12.04. Ubuntu je distribuce vhodná pro začátečníky, vychází z distribuce Debian GNU/Linux. Distribuci je možné stáhnout z <http://www.ubuntu.cz/stahnout> v podobě obrazu ISO. Po stažení ISO souboru je možné instalaci provést prostřednictvím CD/DVD či z flash disku. Jednotlivé kroky jsou velmi dobře popsány na <http://www.ubuntu.cz/stahnout>. Na obr. č. 56 je zobrazeno prostředí GNOME distribuce Ubuntu.



Obr. č. 56 – Prostředí GNOME distribuce Ubuntu 12.04

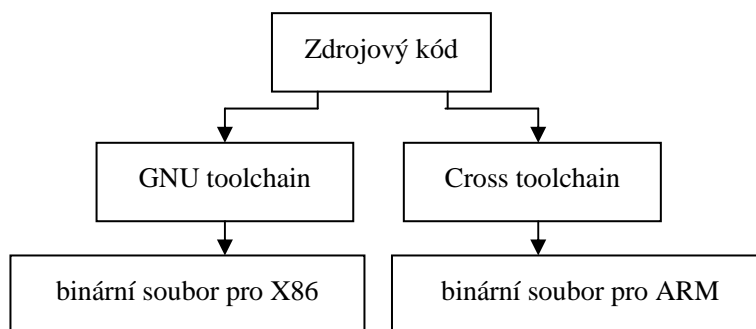
6.2.1 Vytváření aplikací pro platformu ARM

Při vytváření aplikací pro cílovou platformu ARM je vhodné využít výše uvedený operační systém GNU/Linux nebo operační systém Windows. V této práci bude dále uveden popis vytváření aplikací v distribuci Ubuntu 12.04. V předešlé části byl instalací distribuce Ubuntu udělán první krok při vytváření aplikací pro platformu ARM. Po instalaci distribuce Ubuntu je dalším krokem získat kompletní balíček nástrojů, označované jako toolchain.

Toolchain představuje nástroje, které jsou nezbytné pro kompilaci a ladění vytvářených aplikací. Zejména se jedná o kompilátor, linker, assembler a debugger. Při vytváření aplikací pro platformu x86 (PC) se tato kolekce nástrojů označuje jako GNU toolchain a pro uživatele (vývojáře) byla vytvořena dodavatelem Linuxové distribuce. V následující části bude uveden krátký popis jednotlivých komponentů GNU toolchain, protože v dalších krocích budou využity při kompilaci open-source knihovny OpenCV. Komponenty GNU Toolchain:

- GCC (GNU Compiler Collection) kolekce kompilačních nástrojů představuje již v současné době kompilátor, který podporuje velké množství programovacích jazyků (C, C++, Java). Podporuje celou řadu procesorů pro PC, dále i architektury ARM, PowerPC, SPARC. GCC obsahuje mnoho parametrů pro kontrolu nad procesem kompilace, podrobně uvedeno v [20], [21].
- GNU binutils představuje externí nástroje, které jsou důležité pro GCC. Jedná se o nástroje, které jsou využity při práci s binárním kódem při vývoji aplikací v Linuxu.
 - GNU assembler zajišťuje převod assembleru, který představuje zkompilovaný kód v jazyce C do objektového kódu.
 - GNU linker slouží k vytvoření spustitelné aplikace. Provádí zlinkování objektového kódu do standardního ELF (Executable and Linking Format) formátu, který je již spustitelný na cílové platformě.
- GNU objcopy a objdump nástroje jsou určeny pro manipulaci s binárním objektovým kódem
- GNU Make nástroj je často využíván pro kompilaci aplikací pro operační systém GNU/Linux. Je výhodné ho využít při kompilaci velkých projektů. Umožňuje nastavit sérii pravidel, která mají být vykonána při kompilaci. Podrobně je nástroj popsán v [20], [21], [11]
- GNU debugger je nástroj určený pro ladění vytvořených aplikací. Je využit v mnoha vývojářských nástrojích.

Při vytváření spustitelných aplikací pro platformu PC je výsledná aplikace vytvořená pomocí výše popsaných nástrojů také na platformě PC. Jedná se o standardní proces vývoje softwaru na platformě PC. Vytváření aplikací na platformě ARM není v řadě případů možný, protože některé vestavěné systémy nemají dostatečný výkon či úložný prostor. Řešení, které bylo před několika lety navrženo, je nazýváno křížová kompilace (ang. cross compiling). Situaci znázorňuje obr. č. 57.



Obr. č. 57 – Vytváření aplikací pro x86 a ARM

Křížová kompilace představuje vytváření aplikace na platformě PC, která ale bude běžet na jiné cílové platformě [20].

Při realizaci systému, jehož cílem je rozpoznávání vodorovného dopravního se jedná o situaci, kdy je aplikace vytvářena na platformě PC v distribuci Ubuntu 12.04 a výsledná aplikace bude běžet na platformě ARM v distribuci Ångström. Pro tento proces vývoje softwaru je nutné získat potřebné vývojové nástroje pro cílovou platformu, označované jako cross toolchain. Tyto vývojové nástroje jsou většinou obsaženy na CD od dodavatele vývojové desky.

Cross Toolchain pro Olimex SAM-L9261 je uložen na CD, které je součástí vývojové desky. Postup jak připravit cross toolchain pro vývoj aplikací na platformě PC je podrobně popsán v příloze A. Zde bude uvedeno jen porovnání standardního příkazu pro vytvoření spustitelné aplikace hello world pro platformu PC s příkazem pro vytvoření spustitelné aplikace pro platformu ARM.

```
$ gcc -o helloWorld helloWorld.c
```

```
$ arm-none-linux-gnueabi-gcc -o helloWorld helloWorld.c
```

Z uvedených příkazů je vidět, že rozdíl je jen v názvu kompilátoru, který se má zavolat. Zavolání křížového kompilátoru arm-none-linux-gnueabi-gcc vznikne spustitelný soubor, který je možné přenést do cílového vestavěného systému.

6.3 Použité open-source knihovny

Při realizaci systému rozpoznávání vodorovného dopravního značení byly využity open-source knihovny OpenCV a GTK+.

OpenCV (Open Source Computer Vison) knihovna byla využita při realizaci algoritmů číslíkového zpracování obrazu a pro vyčítání vstupních snímků z kamery. OpenCV knihovna představuje multiplatformní open-source knihovnu, která je zaměřená na oblast počítačového vidění. Knihovna je vytvořena firmou Intel a je šiřitelná pod licencí BSL (Boost Software License). První verze byla uvolněna v roce 1999. Knihovna je napsána v jazyce C/C++, obsahuje více jak 2500 optimalizovaných algoritmů určených pro aplikace, kde je požadováno zpracování obrazu v reálném čase [OO], [PP].

Celou knihovnu lze rozdělit do pěti následujících částí:

- **CV** obsahuje algoritmy pro zpracování obrazu a počítačové vidění.
- **MLL** obsahuje statistické nástroje a funkce.
- **HighGUI** obsahuje I/O funkce pro načítání a ukládání obrazových dat.
- **CVAux** obsahuje mnoho zajímavých funkcí, které jsou určeny např. pro detekci obličeje [OO], [PP].

V této části bude dále popsána kompilace OpenCV knihovny pro platformu ARM. V této práci byla využita knihovna OpenCV–1.1, která je implementovaná v jazyce C. Důvodem volby této verze jsou provedené testy na [RR], kde jsou uvedeny dosažené výsledky experimentů při testování výpočetní náročnosti implementované Hough transformace ve všech verzích OpenCV při detekci přímek v obraze. Z provedených experimentů uvádějí, že výpočet Hough transformace ve verzi 1.1 je o polovinu doby rychlejší nežli ve vyšších verzích.

Knihovnu je možné volně stáhnout z <http://opencv.willowgarage.com>. Postup jednotlivých kroků konfigurace, kompilace s přípravou pro vytváření aplikací pro platformu ARM je uveden v příloze B. V této části bude dále popsán konfigurační skript, který nastavuje parametry kompilace. Po stažení a rozbalení knihovny je nejprve nutné před procesem kompilace provést počáteční konfiguraci pomocí nástroje GNU Autoconf zavoláním skriptu `configure`.

Tento skript provádí konfiguraci procesu kompilace, definuje pomocí přepínačů pro jakou cílovou platformu bude kompilace prováděna, jaké mají být použity vývojové nástroje, kde má být vytvořen zkompileovaný soubor a dále umožňuje nastavit řadu doplňků.

Konfigurační příkaz pro nastavení křížové kompilace pro cílovou platformu ARM má následující podobu:

```
./configure --host=arm-Linux --without-gtk --without-carbon --  
without-quicktime --without-1394libs --without-ffmpeg --without-  
python --without-swig --enable-static --disable-shared --disable-apps  
CC=arm-none-Linux-gnueabi-gcc CXX=arm-none-Linux-gnueabi-g++  
CXXFLAGS="-fsigned-char -O2 -pipe" --prefix="/opt/OpenCV_1.1.0"  
LD=arm-none-Linux-gnueabi-ld CPP=arm-none-Linux-gnueabi-cpp  
CXXCPP=arm-none-Linux-gnueabi-cpp AR=arm-none-Linux-gnueabi-ar  
RANLIB=arm-none-Linux-gnueabi-ranlib NM=arm-none-Linux-gnueabi-nm  
STRIP=arm-none-Linux-gnueabi-strip AS=arm-none-Linux-gnueabi-as
```

Význam jednotlivých přepínačů:

`--host` definuje cílovou platformu.

`--without` definuje jaké doplňky mají být vypnuty:

Např. `-without` vypíná doplněk `gtk`, tím je stanoveno, že nebude možné používat funkce pro práci s grafickým uživatelským prostředím, které nabízí knihovna OpenCV. Dále `-without` vypíná volání funkcí, které umožňují načítat a ukládat video soubory.

`--enable-static` definuje vytvoření statických knihoven.

`--disable-shared` zakazuje vytvoření dynamických knihoven.

`CC`, `CXX`, `AS`, `LD`, `AR`, `RANLIB`, `NM`, `STRIP` specifikují jednotlivé nástroje `cross toolchain`, které mají být použity pro proces kompilace.

`--prefix` přepínač určuje místo, ve kterém bude vytvořena zkompileovaná knihovna OpenCV

Po konfiguraci je možné příkazy `make`, `make install` zkompileovat knihovnu OpenCV pro platformu ARM. Výsledkem procesu kompilace jsou hlavičkové soubory, které je možné připojit ve zdrojovém kódu a statické knihovny, které budou přilinkovány k vytvořené aplikaci pro platformu ARM. V příloze B je uvedena ukázka jak vytvořit jednoduchou aplikaci s OpenCV knihovnami pro platformu ARM.

Další použitou knihovnou při realizaci systému rozpoznávání vodorovného dopravního značení byla knihovna GTK+. Knihovna GTK+ byla použita pro zobrazení zachycené scény kamerou před vozidlem na LCD displeji, dále pro informování o detekovaném vodorovném dopravním značení. Knihovna GTK+ patří mezi nejrozšířenější grafické rozhraní pod systémem Linux a je šířena pod licencí LGPL (Lesser General Public Licence). Knihovna obsahuje mnoho vytvořených ovladačích prvků, v odborné literatuře označované jako tzv. Widgets. Ovladačí prvky jsou realizovány pomocí funkcí. Aplikace s grafickým uživatelským rozhraním je pak vytvářena jednoduchým voláním těchto funkcí.

Knihovna GTK+ je napsána v jazyce C a je postavená na řadě dalších knihoven:

- Glib - definuje datové struktury nízké úrovně.
- GObject - implementuje objektově orientovaný systém v C.
- Pango - podporuje vykreslování a rozložení textu.
- ATK - pomáhá při tvorbě aplikací s usnadněním přístupu.
- GDK - zajišťuje vykreslení grafiky na nízké úrovni.
- Xlib - poskytuje služby grafiky na nízké úrovni.

Díky knihovnám Glib, GObject je možné přenášet GTK+ knihovnu na různé architektury [21].

Při vytváření aplikací s grafickým uživatelským rozhraním bude využito dynamických knihoven, které jsou součástí obrazu souborového systému, který je uložen na CD dodávané s vývojovou deskou. Nebude tedy nutné provádět křížovou kompilaci knihovny GTK+. Bohužel obraz souborového systému neobsahuje hlavičkové soubory, které jsou potřeba připojit ve zdrojovém kódu a dále jsou potřena v procesu křížové kompilace při vytvoření spustitelné aplikace. Proto je nejprve nutné instalovat vývojové nástroje GTK+ na platformě PC a při procesu kompilace z nich využít hlavičkové soubory. Podrobný postup vytváření aplikací s grafickým uživatelským rozhraním je obsažen v příloze C.

6.4 Přístup ke kameře

V této části bude uveden jeden z možných postupů, jakým je možné přistoupit ke kameře a vyčítat z ní obrazová data. Operační systém Linux používá pro přístup k hardwarovým zařízením adresář /dev. Jednotlivá hardwarová zařízení jsou v tomto adresáři reprezentována soubory zařízení. Kamera představuje znakové zařízení, do kterého je možno zapisovat či číst sekvenčním způsobem. Dostupná zařízení na operačním systému Linux je možné zobrazit následujícím příkazem:

```
$ ls -l /dev/

crw-rw-rw-  1 root    root      1,    9 Jan  1 00:00 urandom
crw-rw----  1 root    root     189,   0 Jan  1 00:00 usb1
crw-rw-r--  1 root    root     189,   0 Jan  1 00:00 usbdev1.1
```

Znaková zařízení jsou ve výpisu reprezentována znakem c v prvním sloupci.

V této práci bude pro práci s obrazovým zařízením využito rozhraní V4L (Video4Linux). V4L zajišťuje přístup ke kameře a vyčítání obrazových dat prostřednictvím API funkcí. Rozhraní V4L představuje modul jádra. Modul V4L může být již součástí jádra v dané distribuci Linuxu. Aby bylo možné pomocí rozhraní V4L přistoupit k obrazovému zařízení je nutné, aby jádro Linuxu dále obsahovalo i modul, který reprezentuje ovladače pro dané obrazové zařízení. Poté, když je obrazové zařízení připojeno je rozhraním V4L identifikováno a v adresáři /dev je vytvořen soubor video zařízení, prostřednictvím něhož je možné s daným video zařízením komunikovat. Při zadání příkazu vypsání dostupných zařízení by měl být soubor video zařízení jeho součástí.

```
$ ls -l /dev/

crw-rw-rw-  1 root    root      1,    9 Jan  1 00:00 urandom
crw-rw----  1 root    root     189,   0 Jan  1 00:00 usb1
crw-rw-r--  1 root    root     189,   0 Jan  1 00:00 usbdev1.1
crw-rw----  1 root    video     81,   0 Jan  1 00:18 video0
```

Když je možné k video zařízení přistupovat, je možné využít knihovnu OpenCV ve které jsou implementovány funkce pro inicializaci kamery (cvCaptureFromCAM) a funkce pro vyčítání obrazových dat (cvQueryFrame). Tyto funkce jsou založeny na API funkcích V4L.

Protože na vývojové desce Olimex SAM-9261 modul V4L dodavatel distribuce Ångström nezahrnul do jádra a dále jádro neobsahuje modul s ovladači pro webkameru Genius Face Cam 1320, vypracoval jsem v příloze D postup, který informuje, jak provést křížovou kompilaci modulu V4L a modulu s ovladači kamery pro platformu ARM. Dále jak pomocí dynamického načítání modulů připojit zkompilované moduly do jádra. V příloze D je dále uvedená ukázková aplikace s vyčítáním obrazových dat z kamery pomocí funkcí OpenCV, provedení inverze obrazu a jeho zobrazení na LCD displeji pomocí funkcí knihovny GTK+.

7 Implementace

Kapitola navazuje na pátou kapitolu, ve které je uveden návrh vlastního řešení rozpoznávání vodorovného dopravního značení v systému MATLAB. Cílem této části je popsat jednotlivé kroky transformace, které byly nutné vykonat při implementaci navrženého algoritmu do cílové platformy ARM na vestavěném systému. Jednotlivé kroky implementace zobrazuje blokové schéma na obr. č. 58.

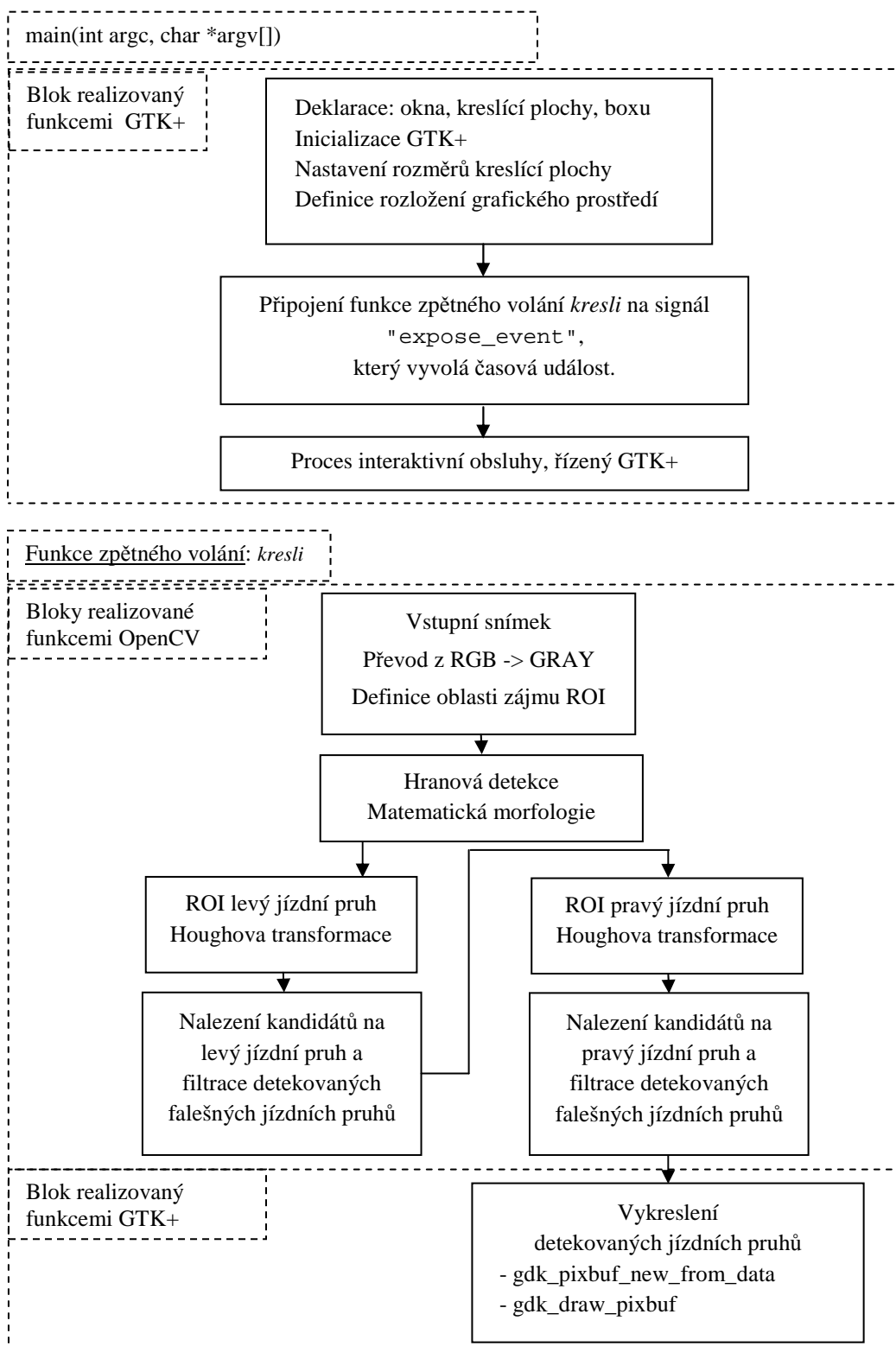
Implementaci lze rozdělit do dvou částí:

- Transformace navrženého algoritmu z programového jazyka MATLAB do jazyka C.
- Vytvoření grafického uživatelského rozhraní, které umožní zobrazení výstupu z navrženého algoritmu.

7.1 Transformace navrženého algoritmu do jazyka C

Implementace algoritmu do jazyka C je zobrazena na obr. č. 58 pomocí blokového schématu. Při návrhu algoritmu v systému MATLAB byly využívány funkce z Image processing toolboxu, který je určen pro zpracování obrazu. Pro implementaci navrženého algoritmu bude základem převod těchto funkcí do jazyka C. Rovnocennou transformaci funkcí z jazyka MATLAB do jazyka C zajišťuje knihovna OpenCV, která je určená pro zpracování obrazu a obsahuje mnoho optimalizovaných funkcí pro běh v reálném čase. V této následující části bude uveden popis transformace jednotlivých bloků navrženého algoritmu do jazyka C.

Prvním krokem, který představuje transformace algoritmu mezi programovacími jazyky, je reprezentace obrazových dat. V systému MATLAB jsou obrazová data reprezentována maticemi, oproti tomu knihovna OpenCV používá pro reprezentaci obrazových dat datovou strukturu `IplImage`, která obsahuje všechny důležité informace o obrazu, podrobně popsáno v [6].



Obr. č. 58 – Blokové schéma procesu implementace

První blok navrženého algoritmu lze realizovat pomocí následujících funkcí:

```
IplImage* snimek = cvCreateImage(cvSize(320,240),IPL_DEPTH_8U,3);
CvCapture* capture = cvCreateCameraCapture(0);
snimek = cvQueryFrame( capture );
cvCvtColor(snimek, snimek, CV_BGR2GRAY );
cvSetImageROI(snimek,cvRect(0,60, 160,60));
cvResetImageROI(snimek);
```

Funkce `cvCreateImage` umožňuje vytvořit prázdný obraz. Funkce `cvCreateCameraCapture` slouží pro inicializaci kamery, ze které je možné následně pomocí funkce `cvQueryFrame` vyčítat obrazová data. Převod z barevného RGB modelu na monochromatický obraz zajišťuje funkce `cvCvtColor` s parametrem `CV_BGR2GRAY`. Funkce `cvSetImageROI` nastavuje požadovanou oblast zájmu, se kterou budou dále pracovat další algoritmy zpracování obrazu. Funkce `cvResetImageROI` provádí rušení této oblasti zájmu.

Druhý blok představuje aplikaci hranového detektoru a matematické morfologie na obraz pomocí funkcí:

```
IplImage* snimek_edg =cvCreateImage(cvSize(160,60),IPL_DEPTH_8U,1);
IplConvKernel* element;
element = cvCreateStructuringElementEx(4,4,2,2,CV_SHAPE_RECT,0);
cvSobel(snimek,snimek_edg, 1, 0, 3 );
cvThreshold(snimek_edg, snimek_edg,100,255,CV_THRESH_BINARY);
snimek_edg = cvMorphologyEx(snimek_edg, snimek_edg,0,element,
                           CV_MOP_CLOSE,1)
```

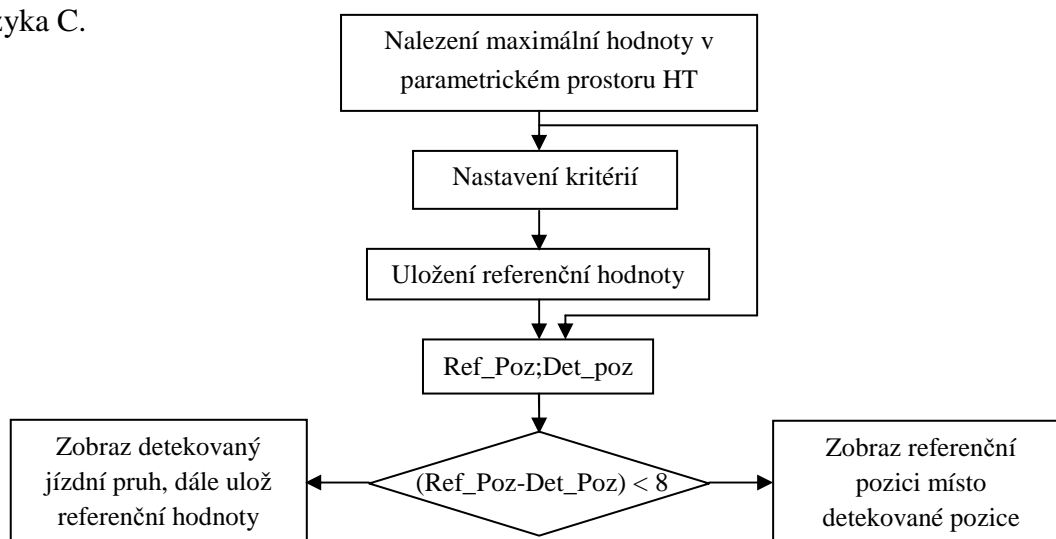
Pro implementaci detektoru hran bude použit Sobelův hranový detektor oproti Robertsovu detektoru, který je využit při realizaci v systému MATLAB. Důvodem je, že v knihovnách OpenCV jsou realizovány hranové detektoři: Sobel, Canny, Laplace, Harrisův. Hranový detektor je realizován funkcí `cvSobel`. Výsledkem hranové detekce je gradientní obraz, který je prahován funkcí `cvThreshold`, jejímž výstupem je binární obraz. Funkce `cvCreateStructuringElementEx` umožňuje definovat strukturní `element`, který je pomocí funkce `cvMorphologyEx` aplikován na binární obraz.

Další skupiny bloků jsou realizovány zvlášť pro detekci levého a pravého jízdního pruhu. Pro realizaci je využita funkce `cvSetImageROI`, která definuje oblasti zájmu pro levý a pravý jízdní pruh. Na každou oblast zájmu je následně aplikována Hough transformace. Hough transformace je v knihovně OpenCV realizována ve standardní a progresivní pravděpodobnostní variantě. Při implementaci byla zvolena varianta progresivní pravděpodobnostní HT. Důvodem byla výhoda v její realizaci. Dále je možné mezi vstupními parametry nastavit minimální délky detekované přímky a vzdálenost mezi přímkami, které mohou být spojeny. Těmito parametry je možné částečně potlačit falešné detekované přímky, které neodpovídají poloze jízdního pruhu. Dalším důvodem byla rychlost výpočtu na základě informací v [6].

```
CvMemStorage* storage = cvCreateMemStorage(256);
CvSeg* lines;
lines = cvHoughLines2(snimek_edg, storage, CV_HOUGH_PROBABILISTIC, 1
CV_PI/15, 30, 10, 10);
```

Pro vytvoření akumulátoru hodnot v HT musí být rezervováno paměťové místo pomocí funkce `cvCreateMemStorage`. Výsledky HT jsou uloženy do struktury `CvSeg`, ve které jsou reprezentovány pozice detekovaných přímek.

Bloky nalezení kandidátu na jízdní pruhy, shlukování a filtrace detekovaných falešných jízdních pruhů, byly realizovány modifikovaným způsobem na rozdíl od návrhu z důvodu běhu aplikace v reálném čase. Modifikovaný algoritmus realizující tyto bloky zobrazuje následné blokové schéma při jehož realizaci byly využity standardní funkce jazyka C.



Obr. č. 59 – Blokové schéma algoritmu

Výsledkem předešlých kroků jsou souřadnice s polohou jízdního pruhu v obrazu. Jednotlivé detekované jízdní pruhy je možné vykreslit do vstupního obrazu, který je reprezentován strukturou `IplImage` funkcí `cvLine(snimek, lines[0], lines[1], CV_RGB(0,0,255,0),1,0,0)`. Při realizaci tohoto algoritmu na platformě PC, je možné využít pro zobrazení obrazu se zachycenou scénou před vozidlem s detekovanými jízdními pruhy další funkce knihovny OpenCV, které umožňují vytvořit jednoduché grafické uživatelské rozhraní. Zobrazení obrazu je pak realizováno pomocí funkce `cvShowImage(WindowName,snimek)`. Při realizaci tohoto algoritmu na platformě ARM nelze využívat funkce knihovny OpenCV vytvářející grafické uživatelské rozhraní, proto bude v další podkapitole popsán postup vytvoření grafického uživatelského rozhraní pro platformu ARM.

7.2 Vytvoření grafického uživatelského rozhraní

Důvodem tvorby grafického uživatelského rozhraní je zobrazení pořízených vstupních obrazových dat a detekovaných jízdních pruhů. Pro tvorbu grafického uživatelského rozhraní bude využita knihovna GTK+. Při vytváření budou využity již vytvořené ovládací prvky (Widgets).

Prvním krokem při vytváření grafického uživatelského rozhraní je deklarace okna `GtkWidget * okno`, deklarace kreslicí plochy `GtkWidget * kresli_plocha`, ve které budou zobrazena obrazová data a dále deklarace ovládacího prvku `box GtkWidget *box`, který zajišťuje rozložení grafického uživatelského rozhraní. Dále je nutné provést inicializaci knihovny GTK+ prostřednictvím funkce `gtk_init`. Jednotlivé ovládací prvky jsou vytvořeny pomocí funkcí: `gtk_window_new`, `gtk_drawing_area_new`, `gtk_vbox_new`.

```
GtkWidget *okno;
GtkWidget * kresli_plocha;
GtkWidget *box;
gtk_init (&argc, &argv);
okno = gtk_window_new (GTK_WINDOW_TOPLEVEL);
kresli_plocha = gtk_drawing_area_new();
box = gtk_vbox_new (GTK_ORIENTATION_VERTICAL, 5);
```

V dalších krocích je provedeno nastavení rozměrů kreslicí plochy dle velikosti zobrazovaného obrazu pomocí funkce `gtk_widget_set_size_request`. Před zobrazením okna a kreslicí plochy je nutné přiřadit tyto prvky ovládacímu prvku box, který zajišťuje rozložení těchto prvků v grafickém uživatelském rozhraní. Přiřazení realizuje funkce `gtk_box_pack_start`. Okno a kreslicí plocha je zobrazena pomocí funkce `gtk_widget_show_all`.

```
gtk_widget_set_size_request(kresli_plocha, snimek ->width,  
                             snimek ->height);  
gtk_box_pack_start (GTK_BOX (box), kresli_plocha, TRUE, TRUE, 0);  
gtk_container_add (GTK_CONTAINER (okno), box);  
gtk_widget_show_all(window);
```

Poslední příkaz provede zobrazení okna a prázdné kreslicí plochy. Pro zobrazování obrazových dat, která zachytila kamera, je dále nutné vytvořit časovou událost, která bude v přesně definovaných časových okamžicích vytvářet signál, který vyvolá funkci zpětného volání, jež je na signál napojená. Ve funkci zpětného volání je již možné vyčíst obrazová data z kamery pomocí funkce OpenCV, provádět zpracování těchto obrazových dat a následně pomocí funkce GTK+ provést jejich zobrazení prostřednictvím kreslicí plochy.

```
g_signal_connect(G_OBJECT(kresli_plocha), "expose_event",  
                 G_CALLBACK(kresli), NULL);  
g_timeout_add(30, (GSourceFunc)time_handler, (gpointer)kresli_plocha);
```

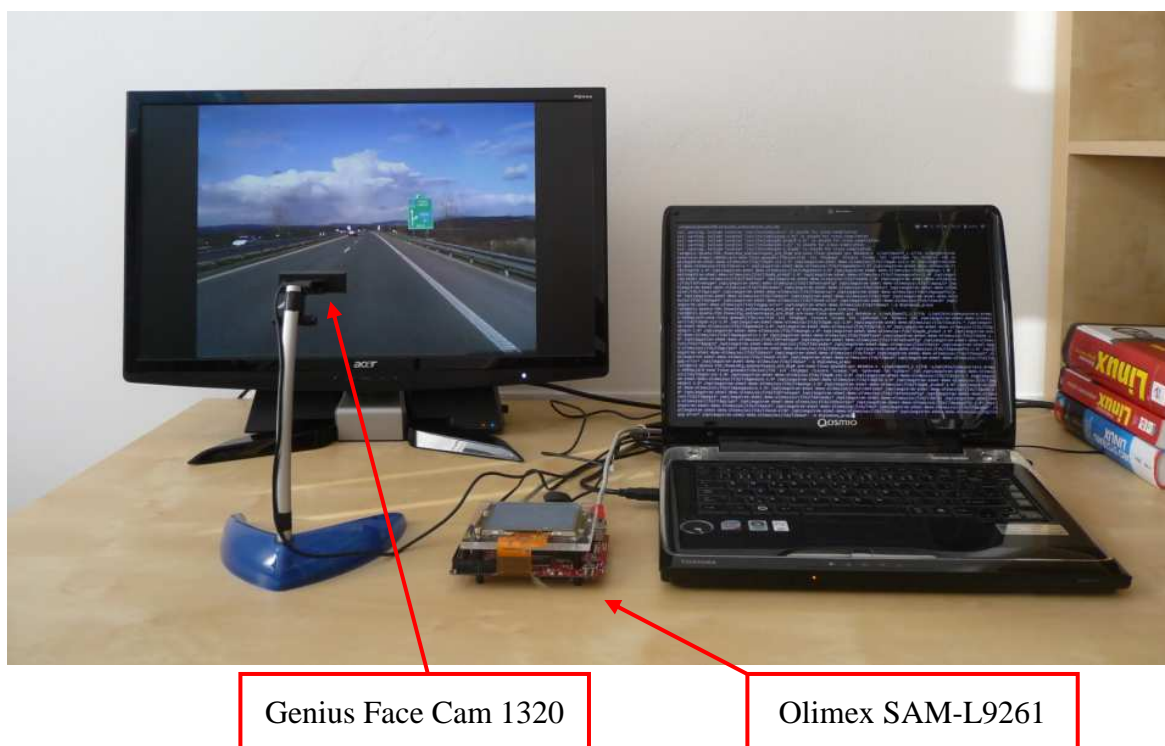
Časová událost je vytvořena funkcí `g_timeout_add`. Funkce `g_signal_connect` zajišťuje připojení funkce zpětného volání `kresli` na signál `"expose_event"`, který každých 30ms vyvolá časová událost.

Posledním krokem při vytváření grafického uživatelského prostředí, které má za cíl zobrazit obrazová data z kamery, je vytvoření funkce zpětného volání `kresli`. Funkce zpětného volání i s ukázkovou aplikací jsou vytvořeny v příloze D. Zobrazení na kreslicí plochu je realizováno funkcí `gdk_pixbuf_new_from_data`, která představuje zásobník pro obrazová data a funkcí `gdk_draw_pixbuf`, která provádí zobrazení.

8 Zhodnocení funkčnosti navrženého systému

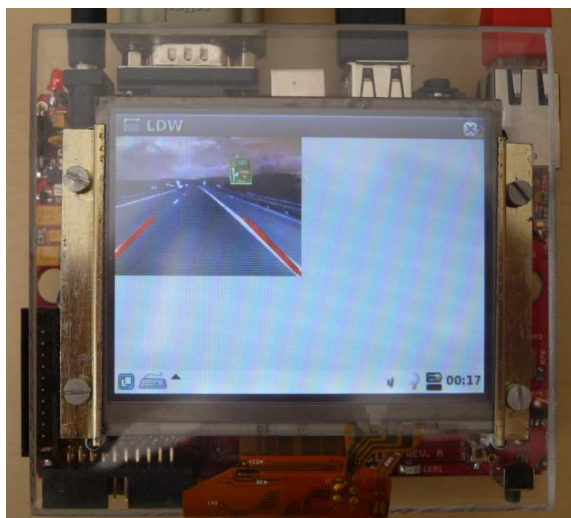
Navržený systém rozpoznávání vodorovného dopravního značení byl testován na základě zaznamenaných dat z provedených měření. Testovací úseky byly vybrány z databáze vytvořené v průběhu práce na základě doporučení z ISO 17361:2007 a FMCSA-MCRR-05-005. Dále bylo snahou dodržet následující doporučení, aby systém LDW třídy I. Byl funkční při rychlosti nad 72km/h při poloměru vozovky větší jak 500m. Dále by měl fungovat při plném či přerušovaném značení a za různých světelných podmínkách.

Následující obrázek znázorňuje uspořádání stanoviště, na kterém probíhalo testování a experimenty při návrhu řešení.



Obr. č. 60 – Testovací stanoviště

Dopravní scéna před vozidlem byla simulována pomocí LCD monitoru s úhlopříčkou 22 palců při rozlišení 1920:1080 obrazových bodů. Webkamera Genius Face Cam 1320 byla umístěna ve výšce 25cm a ve vzdálenosti 45cm od LCD monitoru. Zaznamenaný obraz dopravní scény byl následně zpracován vytvořenou aplikací ve vývojovém kitu.

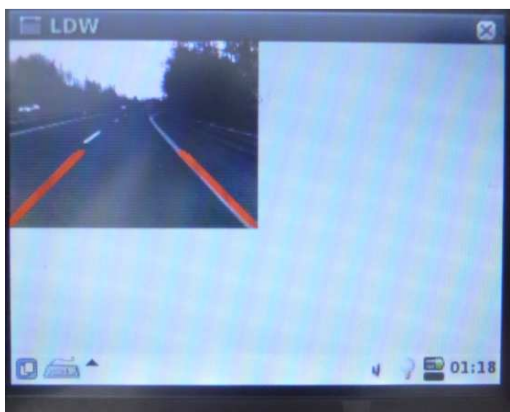


Obr. č. 61 – Navržený systém, detekované vodorovné dopravní značení

Vyhodnocení funkčnosti navrženého systému bude prezentováno pomocí série obrázků z testovacích úseků. Na levé straně obrázku č. 62 jsou zobrazeny ukázky z testovaných úseků na dálnici při kvalitním značení vozovky a při dobrých světelných podmínkách. V pravé části jsou zobrazeny snímky z úseku silnice první třídy.

Z provedených experimentů lze soudit, že navržený systém je schopný provozu na úsecích s kvalitním značením vozovky a dobrými světelnými podmínkami. Na úsecích se zastaralým či poškozeným dopravním značením systém vykazuje určité výpadky.

Provedenými optimalizacemi, které se týkaly především stanovení oblasti zájmu ROI při detekci levého a pravého okraje, nastavení parametrů Hough transformace, bylo dosaženo rychlosti výpočtu navrženého algoritmu pod 0.030 s. Dosažený čas ovlivnil i rozměr vstupního obrazu, který má rozměr 160 x 120 obrazových bodů.



Obr. č. 62 – Série testovacích obrázků

9 Závěr

Cílem této práce bylo prostudovat a vytvořit systém rozpoznávání vodorovného dopravního značení. Problematika rozpoznávání vodorovného dopravního značení je dnes velmi aktuální a nalézá uplatnění zejména u dopravních asistentů motorových vozidel.

V práci byla provedena analýza dostupných systémů a stávajících způsobů řešení. Tato analýza se posléze stala východiskem k navrhnutému vlastnímu systému na deterministických postupech.

Vývoj byl proveden v několika fázích. Nejprve byla pořízena obrazová data, která reprezentují dopravní scénu před jedoucím vozidlem. Následně byl navržen algoritmus v prostředí MATLAB. Dále bylo vytvořeno grafické uživatelské prostředí, které umožňuje modifikovat parametry navrženého algoritmu a simulovat různé situace ve snímané dopravní scéně. Vytvořené prostředí vedlo k urychlení vývoje celého systému.

Navržený algoritmus byl následně transformován do cílové platformy v jazyce C. Vlastní systém byl vytvořen za pomoci vývojového kitu s 32 bitovým procesorem s jádrem ARM. Po provedené optimalizaci parametrů navrženého algoritmu ve vytvořeném grafickém prostředí je systém schopen práce v reálném čase.

V práci se podařilo vytvořit komplexní systém, nicméně stále zůstává celá řada dílčích aspektů, které nebyly řešeny, např. problematika prediktivní filtrace, či řešení problémů způsobených nápisy na vozovce, přechody pro chodce, nebo problémy na mokré vozovce po dešti.

Dosažené výsledky lze shrnout v následujících jednotlivých bodech.

- V průběhu práce byla vyřešena problematika vývoje aplikací s knihovnami OpenCV a GTK+ pro platformu ARM.
- Byl proveden návrh, optimalizace a implementace celého systému.
- Navržený systém je schopen práce na úsecích s kvalitním dopravním značením.

Nedílnou součástí práce je i několik příloh, ve kterých jsou uvedeny postupy a příklady řešených problémů při implementaci. Na doprovodném CD jsou přiložena pořízená obrazová data, která umožňují případné další práce na systému.

Literatura

- [1.] ABDULHAKAM, AM.Assidiq, Othman O. K., Rafiqul I., Sheroz K., Real Time Lane Detection for Autonomous Vehicles. [online]. [cit. 2012-05-23]. Dostupné z: <http://irep.iium.edu.my/5896/1/04580573.pdf>
- [2.] ALDANA, K., Traffic Fatalities in 2010 Drop to Lowest Level in Recorded History. Traffic Fatalities in 2010 Drop to Lowest Level in Recorded History [online]. 2011, [cit. 2012-05-23]. Dostupné z: <http://blogs.cars.com/kickingtires/2009/07/nhtsa-considers-requiring-new-advanced-safety-features-in-cars.html>
- [3.] BEUCHER, S., BILODEAU, M., Road segmentation by watersheds algorithms. [online]. [cit. 2012-05-23]. Dostupné z: http://homepages.uel.ac.uk/u0224863/Project_Msc/Papers_References/road%20detect_watershed.pdf
- [4.] BEUCHER, S., MARCOTEGUI, B., P algorithm, a dramatic enhancement of the watherfall transformation. [online]. [cit. 2012-05-23]. Dostupné z: http://cmm.ensmp.fr/~beucher/publi/P-Algorithm_SB_BM.pdf
- [5.] BISHOP, R., Intelligent Vehicle Technology and Trends. Canton Street,Norwood: ARTECH HOUSE, 2005. ISBN 1-58053-911-4. Dostupné z: <http://203.158.253.140/media/e-Book/Engineer/Automotive/Intelligent%20Vehicle%20Technology%20and%20Trends.pdf>
- [6.] BRADSKI, G., KAEHLER, A., Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, 2008. ISBN 0596516134.
- [7.] CANNY, J. A Computational Approach To Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986. [cit. 2012-05-23].
- [8.] CUALAIN, D.O., C. HUGHES, M. GLAVIN a E. JONES. Automotive standards-grade lane departure. [online]. [cit. 2012-05-23]. Dostupné z: www.fmcsa.dot.gov/facts-research/research-technology/report/lane-departure-warning-systems.htm
- [9.] Česká republika. Národní strategie bezpečnosti silničního provozu 2011 - 2020. In: Ministerstvo dopravy. 2011. [online]. [cit. 2012-05-23]. Dostupné z: http://www.ibesip.cz/files/=4221/NSBSP%2b2011-2020_form%c3%a1tov%c3%a1n%c3%ad_II.pdf

- [10.] DOBEŠ, M., Zpracování obrazu a algoritmy v C#: automatické rozpoznávání, úprava snímků. Praha: BEN-Technická literatura, 2008. ISBN 978-80-7300-2.
- [11.] DOBROVOLNÝ, M., Studijní materiály k předmětu Zpracování obrazu. Dostupné z: <http://martindobrovolny.cz/index.phpid=vyuka>
- [12.] GAIKWAD, V., An improved lane departure method for Advanced Driver Assistance System. Computing, Communication and Applications (ICCCA), 2012 International Conference on. 22-24 Feb. 2012
- [13.] HLAVÁČ, V., ŠONKA, M. Počítačové vidění. Praha: Grada, 1992. ISBN 80-85424-67-3
- [14.] HLAVÁČ, V., SEDLÁČEK, M. Zpracování signálu a obrazu. Pracovní verze skripta v tisku pro studenty. Praha: ČVUT, 1999.
- [15.] ISO 17361:2007. Intelligent transport systems Lane departure warning systems: Performance requirements and test procedures.,2010. Dostupné z: http://www.iso.org/iso/catalogue_detail.htm?csnumber=41105
- [16.] KALOVÁ, I., Regionální segmentace a shlukování: Počítačové vidění. [online]. [cit. 2012-05-23]. Dostupné z: <http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/06%20-%20Regionalni%20segmentace%20a%20shlukovani.pdf>
- [17.] KLÍMA, M., BERNAS, M., HOZMAN, J., DVOŘÁK, P. Zpracování obrazové informace. První vydání. Praha: ČVUT, 1996. ISBN 80-01-01436-3.
- [18.] KREUCHER, C. a S. LAKSHMANAN. LANA: A Lane Extraction Algorithm that Uses Frequency. IEEE Transactions on Robotics and Automation [online]. [cit. 2012-05-23]. Dostupné z: <http://www-personal.umich.edu/~ckreuche/PAPERS/1999IEEEERA.pdf>
- [19.] MARKLEY, S., NHTSA Considers Requiring New Advanced Safety Features in Cars. NHTSA Considers Requiring New Advanced Safety Features in Cars [online]. [cit. 2012-05-23]. Dostupné z: <http://blogs.cars.com/kickingtires/2009/07/nhtsa-considers-requiring-new-advanced-safety-features-in-cars.html>
- [20.] MASTERS, J., BLUM, R., Linux profesionálně: programování aplikací. BRNO: Zoner Press, 2008. ISBN 978-80-86815-71-8.

- [21.] MATTHEW, N., STONES, R., Linux: Začínáme programovat. BRNO: Computer Press, a.s., 2008. ISBN 9788025119334.
- [22.] Otsu, N., A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics, 1979, pp. 62-66.
- [23.] PEDEN, M., SCURFIELD, R., SLEET, D., MOHAN, D., HYDER, A., JARAWAN, E., MATHERS, C., World report on road traffic injury prevention. [online]. [cit. 2012-05-23]. Dostupné z: http://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/en/index.html
- [24.] PELIKÁN, J., Barevné systémy. [online]. [cit. 2012-05-23]. Dostupné z: <http://cgg.mff.cuni.cz/~pepca/lectures/pdf/colors.pdf>
- [25.] RIAT, Jean-Christophe. Lane departure warning system developed by PSA Peugeot Citroën. 2005, č. 1. Dostupné z: <http://www.unece.org/fileadmin/DAM/trans/doc/2005/wp29/ITS-09-05e.pdf>
- [26.] RONG, J. H., GONG, J., KHAN, S., A Lane Detection Method for Lane Departure Warning System. International Conference on Optoelectronics and Image Processing, 2010.
- [27.] Rudra N. H., Shahanaz S., Subhadip B., Radha K., P.: A Simple and Efficient Lane Detection using Clustering and Weighted Regression. COMAD 2009, [cit. 2012-05-23]. Dostupné z: http://www.cse.iitb.ac.in/~comad/2009/proceedings/I2_3.pdf
- [28.] Ředitelství služby dopravní policie Policejního prezidia České republiky. Přehled o nehodovosti na pozemních komunikacích v České Republice za rok 2010 [online]. 2012 [cit. 2012-05-23]. Dostupné z: www.policie.cz/soubor/nehody-2010-text-final-tisk-pdf.aspx
- [29.] ŘÍČNÝ, V., VIDEOTECHNIKA. 4. upravené. Brno: Vysoké učení technické v Brně, 2006. ISBN 80-214-3225-X. Dostupné z: http://www.urel.feec.vutbr.cz/web_documents/literature/mvdk.pdf
- [30.] SERRA, J., Image Analysis and Mathematical Morphology. Academic Press, 1982. ISBN 012637242X.
- [31.] Toyota IMTS at Expo 2005 in Japan. Toyota IMTS at Expo 2005 in Japan. 2005, č. 1. Dostupné z: <http://www.johnsonvisual.com/imts-expo2005.html>

- [32.] Tsung-Ying Sun, S.-J. T., CHAN. V., Hsi color model based lane-marking detection. IEEE Intelligent Transportation Systems Conference, 2006.
- [33.] VAVERKA, L., Infiniti M30d S – Essence luxus. Infiniti M30d S – Essence luxus. Dostupné z: <http://www.auto.cz/test-infiniti-m30d-s-essence-luxusu-56591>
- [34.] Voisin, V. ,Avila, M., Emile, B., Bégot, S., Bardet, J. Road Markings Detection and Tracking Using Hough Transform and Kalman Filter, Proc. ACIVS, 2005, pp.76-83.
- [35.] Web kamera GENIUS. Dostupné z: <http://www.softtech.cz/webkamery/genius/152057>
- [36.] Web kamera Hercules. Dostupné z: <http://www.walmart.com/ip/Hercules-Webcam-with-VGA-Sensor-and-Integrated-Microphone/10363729>
- [37.] WIKIPEDIE. Konvoluce.[online]. [cit. 2012-05-23]. Dostupné z: <http://cs.wikipedia.org/wiki/Konvoluce>
- [38.] WINHOLDEM. Uniscraper profile anatomy. Uniscraper profile anatomy [online]. [cit. 2012-05-23]. Dostupné z: <http://forum.winholdem.net/wbb/viewtopic.php?t=5388>

Příloha A

Příprava cross-toolchain pro vývoj aplikací na platformě PC pro cílovou platformu ARM

Cross-toolchain je uložen na přiloženém CD k vývojové desce v adresáři `toolchain` pod názvem `codesourcery-armgcc-2099q1-repack.tar.bz2`.

- 1) Kopírování souboru `codesourcery-armgcc-2099q1-repack.tar.bz2` do adresáře `/opt`

```
cp codesourcery-armgcc-2099q1-repack.tar.bz2 /opt
```

- 2) Rozbalení pomocí příkazu:

```
tar xvf codesourcery-armgcc-2099q1-repack.tar.bz2
```

- 3) Nastavení cesty k používanému cross kompilátoru

```
export PATH=$PATH:/opt/bin/codesourcery-armgcc-2009q1/bin
```

- Následně je již možné používat cross kompilátor

- 4) Ukázka práce s cross kompilátorem:

- a) Vytvoření pracovního adresáře v GNU/Linux

```
mkdir /home/workspace
```

- b) Kopírování připravené ukázkové aplikace z adresáře na CD `/prilohy/priklady/pr_0`

```
cp /media/prilohy/priklady/pr_0.rar /home/workspace
```

```
tar xvf pr_0.rar
```

```
rm pr_0.rar
```

```
cd /home/workspace/pr_0
```

- c) Kompilace ukázkové aplikace pro platformu ARM

```
arm-none-linux-gnueabi-gcc -o helloWorld helloWorld.c
```

- d) Ověřit, zda aplikace `helloWorld` byla vytvořená pro platformu ARM

```
file helloWorld
```

```
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV),  
dynamically linked (uses shared libs), for GNU/Linux 2.6.14, not  
stripped
```

5) Pro přenesení binárního souboru do vývojového kitu bude využit Apache serveru

a. Práce v konzoli na PC

- i. Apache by měl být předinstalován na většině distribucí vycházejících z Debian, tedy i na distribuci Ubuntu. V případě potřeby je možné instalaci provést následujícím příkazem:

```
apt-get install apache2
```

- ii. Nastavení sítě

```
ifconfig eth0 192.168.1.10 netmask 255.255.255.0
```

- iii. Spuštění Apache serveru:

```
Service apache2 start
```

- iv. Nakopírování binárního souboru do adresáře `/var/www`

```
cp helloWorld /var/www
```

b. Práce v konzoli na ARM

- i. Vytvoření pracovního adresáře

```
mkdir /home/workspace_arm/
```

```
cd /home/workspace_arm/
```

- ii. Nastavení sítě

```
ifconfig eth0 192.168.1.20 netmask 255.255.255.0
```

- iii. Přenesení binárního souboru do ARM

```
wget http://192.168.1.10/helloWorld
```

- iv. Spuštění aplikace `helloWorld`

```
chmod 777 helloWorld
```

```
./helloWorld
```

root@sam9-l9261: První aplikace pro platformu ARM

Příloha B

Vývoj aplikací s knihovnou OpenCV-1.1

- 1) Knihovna je na přiloženém CD v adresáři
/prilohy/knihovna_OpenCV_1_1/opencv-1.1.0.tar
- 2) Kopírování opencv-1.1.0.tar do adresáře /opt

```
cp /media/prilohy/knihovna_OpenCV_1_1/opencv-1.1.0.tar /opt
```

- 3) Rozbalení pomocí příkazu:

```
cd /opt  
  
tar xvf opencv-1.1.0.tar  
  
cd /opt/opencv-1.1.0
```

- 4) Konfigurace kompilace pro cílovou platformu ARM

```
./configure --host=arm-Linux --without-gtk --without-carbon --  
without-quicktime --without-1394libs --without-ffmpeg --without-  
python --without-swig --enable-static --disable-shared --disable-apps  
CC=arm-none-Linux-gnueabi-gcc CXX=arm-none-Linux-gnueabi-g++  
CXXFLAGS="-fsigned-char -O2 -pipe" --prefix="/opt/OpenCV_1.1.0"  
LD=arm-none-Linux-gnueabi-ld CPP=arm-none-Linux-gnueabi-cpp  
CXXCPP=arm-none-Linux-gnueabi-cpp AR=arm-none-Linux-gnueabi-ar  
RANLIB=arm-none-Linux-gnueabi-ranlib NM=arm-none-Linux-gnueabi-nm  
STRIP=arm-none-Linux-gnueabi-strip AS=arm-none-Linux-gnueabi-as
```

- Důležitý přepínač, který určuje, kde bude po kompilaci knihovna vytvořena:

```
--prefix="/opt/OpenCV_1.1.0"
```

- 5) Kompilace a zkopírování knihovny a hlavičkových funkcí do adresáře
definovaného přepínačem --prefix v předešlém kroku:

```
make  
make install
```

- 6) V adresáři /opt se nacházejí vytvořené statické knihovny a hlavičkové soubory

```
/opt/OpenCV_1.1.0/lib  
/opt/OpenCV_1.1.0/include/opencv
```

7) Vytvoření jednoduché aplikace s využitím funkcí OpenCV

- a. Kopírování vytvořené ukázkové aplikace z CD v
/prilohy/priklady/pr_1 do pracovního adresáře

```
cp    /media/prilohy/priklady/pr_1.rar  /home/workspace
tar xvf pr_1.rar
rm pr_1.rar
cd    /home/workspace/pr_1
```

- b. Proces křížové kompilace pro platformu ARM
- Pomocí MAKEFILE

```
make
```

- c. Přenesení vytvořeného binárního souboru do vývojového kitu, stejným
postupem jako v příloze A v kroku č. 5

```
wget http://192.168.1.10/pr_1
```

- d. **Dále je nutné přenést do pracovního adresáře na vývojovém kitu
dynamickou knihovnu **libstdc++.so.6.0**,**
která je umístěná na CD v adresáři /prilohy/knihovna_OpenCV_1_1

```
cp    /media/prilohy/knihovna_OpenCV_1_1/libstdc++.so.6.0  /var/www
```

- Následné přenesení dle přílohy A kroku č. 5.

```
wget http://192.168.1.10/libstdc++.so.6.0
```

8) Spuštění vytvořené aplikace na vývojovém kitu:

```
./pr_1
```

```
root@sam9-19261: Prvni aplikace s využitím funkci OpenCV.
```

Příloha C

Vývoj aplikací s knihovnou GTK+

- 1) Prvním krokem při vytváření aplikací s grafickým uživatelským rozhraním bude instalace vývojových nástrojů GTK+ na platformě PC.

```
apt-get install libgtk.2.0-dev
```

- 2) Kopírování obrazu souborového systému z příloženého CD k vývojovému kitu do adresáře /opt a jeho následné rozbalení.

```
cp /media/SAM9-L9261_Rev.A/sources/sam9-l9261-rootfs.tgz /opt/  
tar xvf sam9-l9261-rootfs.tgz  
rm sam9-l9261-rootfs.tgz
```

- 3) Vytvoření jednoduché aplikace s grafickým uživatelským rozhraním

- a. Kopírování vytvořené ukázkové aplikace z CD v
/prilohy/priklady/pr_2 do pracovního adresáře

```
cp /media/prilohy/priklady/pr_2.rar /home/workspace  
tar xvf pr_2.rar  
rm pr_2.rar  
cd /home/workspace/pr_2
```

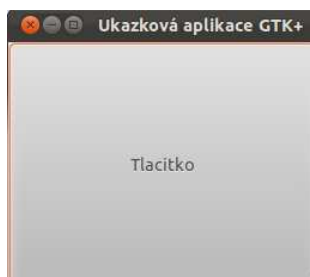
- b. Proces křížové kompilace pro platformu ARM
- Pomocí MAKEFILE
make

- c. Přenesení vytvořeného binárního souboru do vývojového kitu stejným
postupem jako v příloze A v kroku č. 5

```
wget http://192.168.1.10/pr_2
```

- 4) Spuštění vytvořené aplikace na vývojovém kitu:

```
./pr_2
```



Obr. č. 63- Aplikace využívající funkce knihovny GTK+

Příloha D

Připojení kamery

- 1) Prvním krokem při připojení kamery bude kopírování souboru s jádrem Linuxu z CD, které je přiložené k vývojovému kitu do adresáře /opt a jeho následné rozbalení.

```
cp /media/SAM9-L9261_Rev.A/sources/linux-2.6.30-olimex.tar.bz2 /opt/  
  
tar xvf linux-2.6.30-olimex.tar.bz2  
  
rm linux-2.6.30-olimex.tar.bz2
```

- 2) Kamera bude připojena na základě vytvoření modulu jádra a jeho následné dynamické zavedení pomocí LKM (Linux Kernel Modules)
- 3) Nejprve je nutné provést konfiguraci vlastností jádra a vytvořit moduly V4L, sn9c102, host_usb.

- Aby bylo možné provést konfiguraci jádra pomocí konfiguračního grafické rozhraní, je nejprve nutné nainstalovat balíček gt3:

```
apt-get install qt3-dev-tools
```

- Následná konfigurace:

```
cd /opt/linux-2.6.30-olimex  
export CROSS_COMPILE="arm-none-linux-gnueabi-"  
make ARCH=arm sam9_l9261_defconfig  
make ARCH=arm xconfig
```

- 4) Příkazem make ARCH=arm xconfig se otevřelo grafické konfigurační okno následnou důležitou konfiguraci prezentují obrázky, které jsou uloženy na přiloženém CD diplomové práce v adresáři /prilohy/img_konfig_jadra
- Po provedené konfiguraci jádra dle obrázků je možné přejít na proces vytvoření modulů jádra pomocí následujících příkazů:

```
ARCH=arm make modules  
  
make ARCH=arm modules_install
```

- Ukončením kompilace se moduly vytvoří v následném adresáři s koncovkou .ko
- ```
/lib/modules/2.6.30-olimex/kernel/drivers/media/video/
```



- 5) Podle následujícího seznamu příkazu se jednotlivé moduly zkomprimují do formátu .rar a zkopírují do adresáře /var/www

```
cd /lib/modules/2.6.30-olimax/kernel/drivers/media/video/tar -
cvf video.tar ./video
cp video.tar /var/www
rm video.tar
cd /lib/modules/2.6.30-olimax/kernel/drivers/usb
tar -cvf host.tar ./host
cp host.tar /var/www
rm host.tar
```

- 6) Následně se zkomprimované moduly přenesou do adresářů na vývojovém kitu a rozbali

- Práce v konzoli na vývojovém kitu:

```
cd /lib/modules/2.6.30-olimax/kernel/drivers/
wget http://192.168.1.10/video.tar
tar xvf video.tar
rm video.tar
cd /lib/modules/2.6.30-olimax/kernel/drivers/usb
wget http://192.168.1.10/host.tar
tar xvf host.tar
rm host.tar
```

- 7) Posledním krokem je provedení restartu systému a při jeho následném spuštění je možné připojit webkameru Genius Face Cam 1320 a následujícími příkazy zavést moduly:

```
cd /lib/modules/2.6.30-
olimax/kernel/drivers/video/sn9c102

modprobe sn9c102

cd /lib/modules/2.6.30-olimax/kernel/drivers/video/uv
modprobe uvcvideo
```

- 8) Ověření správného postupu lze provést příkazem `ls -l /dev/video*`, který se provede výpis dostupných video zařízení. V tomto okamžiku je již možné přistupovat ke kameře.

```
ls -l /dev/video*
```

```
crw-rw---- 1 root video 81, 0 Jan 1 00:05 /dev/video0
```

- 9) Vytvoření jednoduché aplikace s grafickým uživatelským rozhraním a zpracováním obrazu z kamery pomocí funkcí OpenCV:

- a. Kopírování vytvořené ukázkové aplikace z CD v  
/prilohy/priklady/pr\_3 do pracovního adresáře

```
cp /media/prilohy/priklady/pr_2.rar /home/workspace
```

```
tar xvf pr_2.rar
```

```
rm pr_2.rar
```

```
cd /home/workspace/pr_2
```

- b. Proces křížové kompilace pro platformu ARM  
- Pomocí MAKEFILE

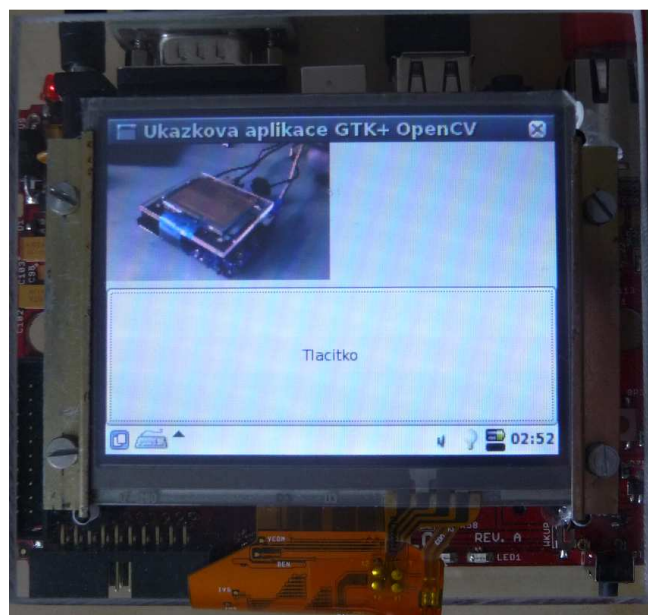
```
make
```

- c. Přenesení vytvořeného binárního souboru do vývojového kitu stejným postupem jako v příloze A v kroku č. 5

```
wget http://192.168.1.10/pr_3
```

- d. Spuštění vytvořené aplikace na vývojovém kitu:

```
./pr_3
```



Obr. č. 64 – Aplikace využívající funkce GTK+ a OpenCV