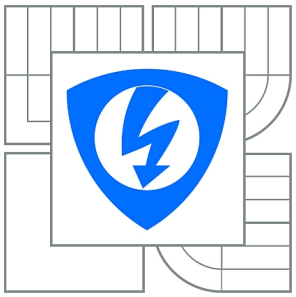
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

DETEKCE RYCHLOSTI PŘIBLIŽOVÁNÍ AUTOMOBILU NA ZÁKLADĚ ZPRACOVÁNÍ OBRAZU KAMERY

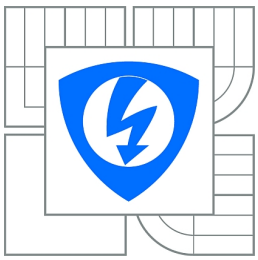
DETECTION OF CAR APPROACH SPEED USING CAMERA IMGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN KOVÁŘ
AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ ŠMIRG
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jan Kovář	***ID:***	83602
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Detekce rychlosti přibližování automobilu na základě zpracování obrazu kamery

POKYNY PRO VYPRACOVÁNÍ:

Diplomová práce se bude zabývat měřením rychlosti přibližování automobilů za pomocí kamery umístěne za předním sklem pohybujícího se auta. Program bude napsán v některém z programovacích jazyků (JAVA, C++, atd.)

DOPORUČENÁ LITERATURA:

[1] HLAVÁČ?, Václav, SEDLÁČEK, Miloš. Zpracování signálů a obrazů. Praha : ČVUT, 2005. 255 s. ISBN 80-01-03110-1.
[2] SCHLESINGER, Michail I., HLAVAČ?, Václav. Deset přednášek z teorie statistického a strukturního rozpoznávání-. Praha : ČVUT, 1999. 521 s. ISBN 80-01-01998-5.
[3] SONKA, Milan, HLAVAC, Vaclav, BOYLE, Roger. Image Processing, Analysis and Machine Vision. 3rd edition. Toronto : Thomson, 2008. 829 s. ISBN 978-0-495-08252-1.
[4] SVOBODA, Tomas, KYBIC, Jan, HLAVAC, Vaclav. Image Processing, Analysis and Machine Vision : A MATLAB Companion. Toronto : Thomson, 2008. 255 s. ISBN 978-0-495-29595-2.
[5] VERNON, David. Machine Vision : Automated Visual Inspection and Robot Vision. Hemel Hempstead : Prentice Hall International (UK) Ltd., 1991. 260 s. ISBN 0-13-543398-3.

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Ondřej Šmirg

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Tato práce pojednává o zpracování digitálního obrazu, od počátečního pořízení snímků digitálního obrazu, přes následné zpracování, segmentaci, až po algoritmy pro detekci tvarů na obrazové scéně.
Zpracování obrazu je velice rozsáhlé téma, proto zde jsou pro bližší pochopení rozebrány základní principy vnímání a zpracování obrazového signálu, reprezentací obrazu, jeho snímáním počínaje, přes filtry upravující digitální obraz pro zpracování až po metody detekující objekty v obraze. Jsou rozebrány metody pro statické a dynamické rozpoznávání objektů.
Dále je demonstrována závislost velikosti objektu v obraze na vzdálenosti od kamery, na jehož základě můžeme určovat rychlost přibližování či vzdalování od objektu. Ukážeme si, že pro konkrétní určení vzdálenosti nepotřebujeme v důsledku znát skutečnou velikost objektu. To je dáno tím, že poměr mezi velikostí objektu v závislosti na vzdálenosti je pro každý objekt stejný. Nakonec této práce jsou prezentovány výsledné snímky obrazu po implementaci pomocí knihovny OpenCV.

Klíčová slova:

zpracování obrazu, segmentace, filtry, detekce objektů, openCV

ABSTRACT

The thesis deals with digital image processing, from the initial acquisition of digital picture frames, subsequent processing segmentation and algorithms to detect visual shapes on the scene.

Image processing is a very broad topic, so here are analyzed for more understanding of the fundamental principles of perception and processing of video signals, image representation, his starting shooting through filters governing digital image processing methods to detect the objects in an image.

It is also demonstrated by the size dependence of the object in the image on the distance from the camera, whereby we can determine the speed of approaching or moving away from the object. We will show you the specific determination of the distance we need to know the actual result size of the object. This is because the ratio between the size of the object depending on the distance is the same for each object. Finally, this work presents the resulting image frames for implementation using OpenCV library.

Keywords:

imaging, segmentation, filters, object detection, OpenCV

BIBLIOGRAFICKÁ CITACE DÍLA

KOVÁŘ, J. *Detekce rychlosti přibližování automobilu na základě zpracování obrazu kamery.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 42 s. Vedoucí diplomové práce Ing. Ondřej Šmirg.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Detekce rychlosti přibližování automobilu na základě zpracování obrazu kamery" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 26.5.2010 .....

podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Ondřejovi Šmirgovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne 26.5. 2010 .....

podpis autora

Obsah

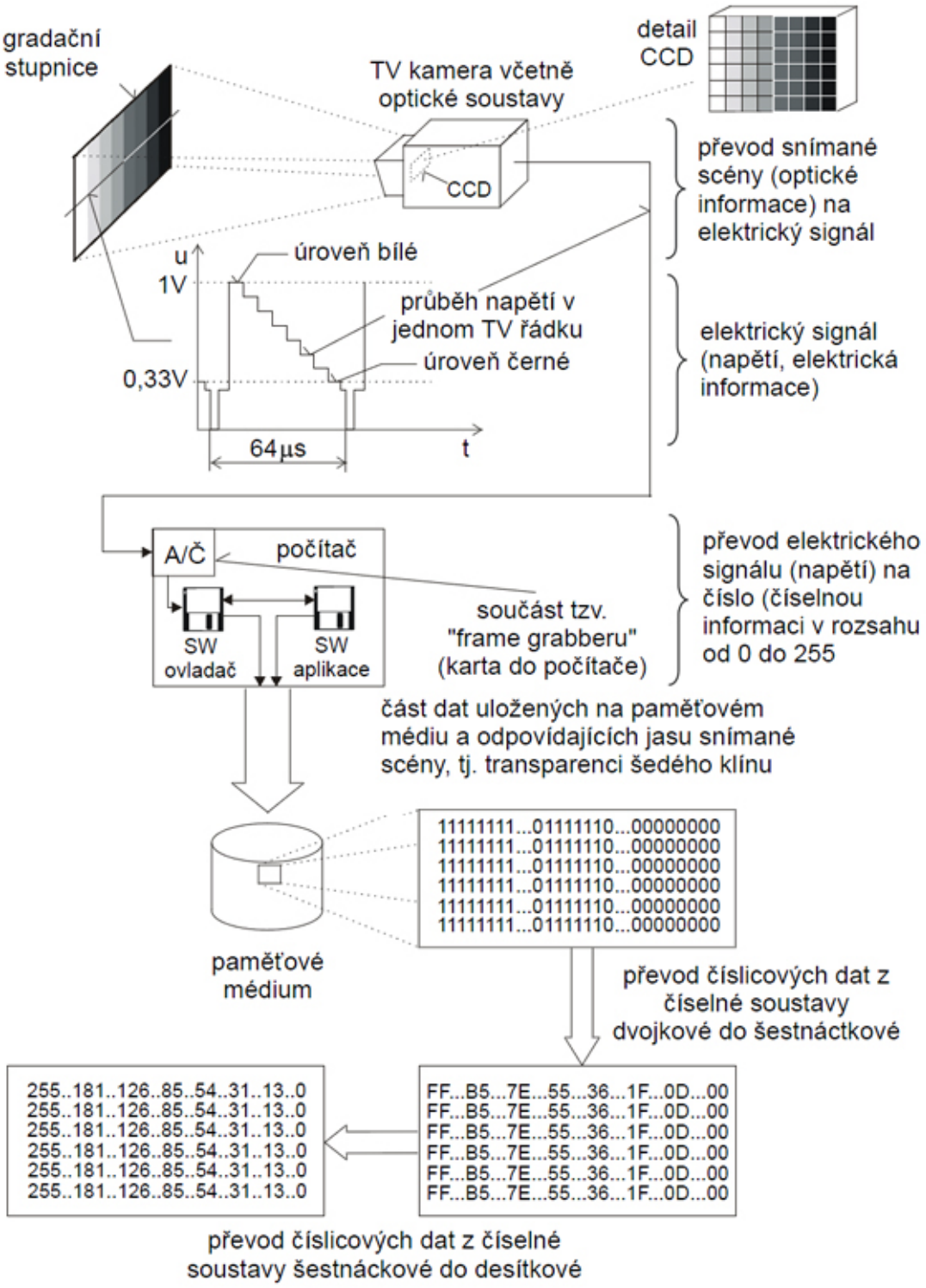
1. Úvod

Tato práce si bere za cíl teoretický rozbor zpracování digitálního obrazu, od počátečního pořízení snímků digitálního obrazu, přes následné zpracování, segmentaci, až po algoritmy pro detekci tvarů na obrazové scéně.
Zpracování obrazu je velice rozsáhlé téma, proto zde jsou pro bližší pochopení rozebrány základní principy vnímání a zpracování obrazového signálu, reprezentací obrazu, jeho snímáním počínaje, přes filtry upravující digitální obraz pro zpracování až po metody detekující objekty v obraze. Jsou rozebrány metody pro statické a dynamické rozpoznávání objektů.
Dále je demonstrována závislost velikosti objektu v obraze na vzdálenosti od kamery, na jehož základě můžeme určovat rychlost přibližování či vzdalování od objektu. Ukážeme si, že pro konkrétní určení vzdálenosti nepotřebujeme v důsledku znát skutečnou velikost objektu. To je dáno tím, že poměr mezi velikostí objektu v závislosti na vzdálenosti je pro každý objekt stejný. Nakonec této práce jsou prezentovány výsledné snímky obrazu po implementaci pomocí knihovny OpenCV.

2. Snímání a digitalizace obrazu

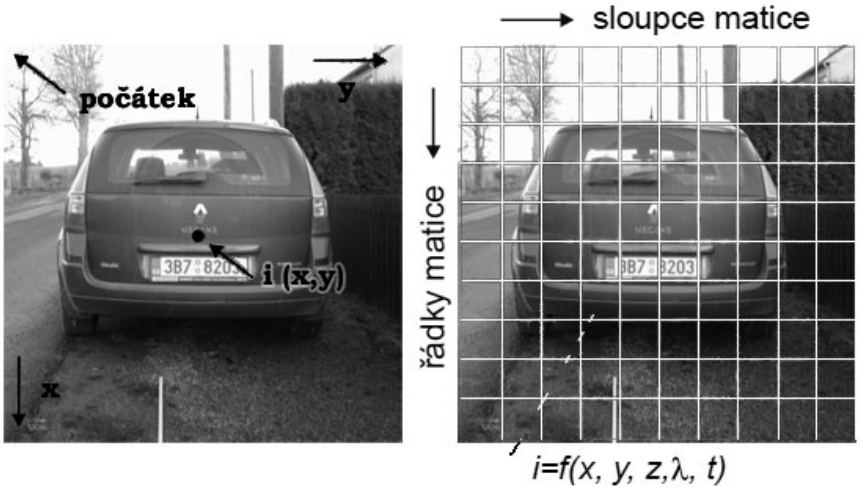
Pořízení obrazu je prvotní operací a kvalita obrazu je touto operací značně ovlivněna. Obraz vzniká tak, že světlo generované zdrojem se odráží od povrchu nějakého předmětu. Toto odražené světlo je zachyceno senzorem jako matice čísel. Na obraz pohlížíme jako na 2D funkci $f(x,y)$, kde x a y jsou prostorové souřadnice a funkční hodnota funkce f odpovídá jasu obrazu v daném místě. Funkci $f(x,y)$ pak definujeme pomocí dvou dílčích funkcí charakterizujících zdroj osvětlení (iluminace $i(x,y)$) a zobrazovaný objekt (odrazivost, resp. reflektance $r(x,y)$). Obor hodnot funkce $r(x,y)$ je v rozmezí 0, odpovídá absorpci světla, až 1, což odpovídá úplnému odrazu. Pro další digitální zpracování musíme obraz, doposud uvažovaný v jeho spojité podobě, diskretizovat pomocí vzorkování a

kvantování. Výsledkem je pak obraz sestavený z obrazových elementů neboli pixelů, přičemž každý z nich má svoji diskrétní polohu a hodnotu. Čerpáno z [6].



Obr. 1: Převod optické informace na číslicový obraz

Obrazový bod, pixel (z anglického „picture element"), je prkem matice daného obrazu. Na obr. 2 je znázorněn obrázek, kde jsou vyznačeny orientace jednotlivých os. Hodnota obrazového bodu, pixelu, je úměrná jasu snímané scény v daném bodě.



Obr. 2: Orientace os při digitální reprezentaci obrazu

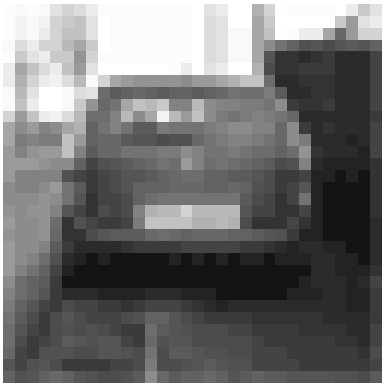
Tmavá místa ve scéně budou obsahovat malá čísla a naopak světlá místa budou obsahovat vyšší čísla. Např. pro bitovou hloubku 8b, tzn. rozsah od 0 do 255 bude číslo „0" pro černou a 255 pro bílou. Pro velkou většinu aplikací je takový počet úrovní dostačující, nehledě na fyziologii lidského oka, které takové množství úrovní není schopno rozlišit, což demonstruje obr. 3.

2.1 Vzorkování signálu a aliasing

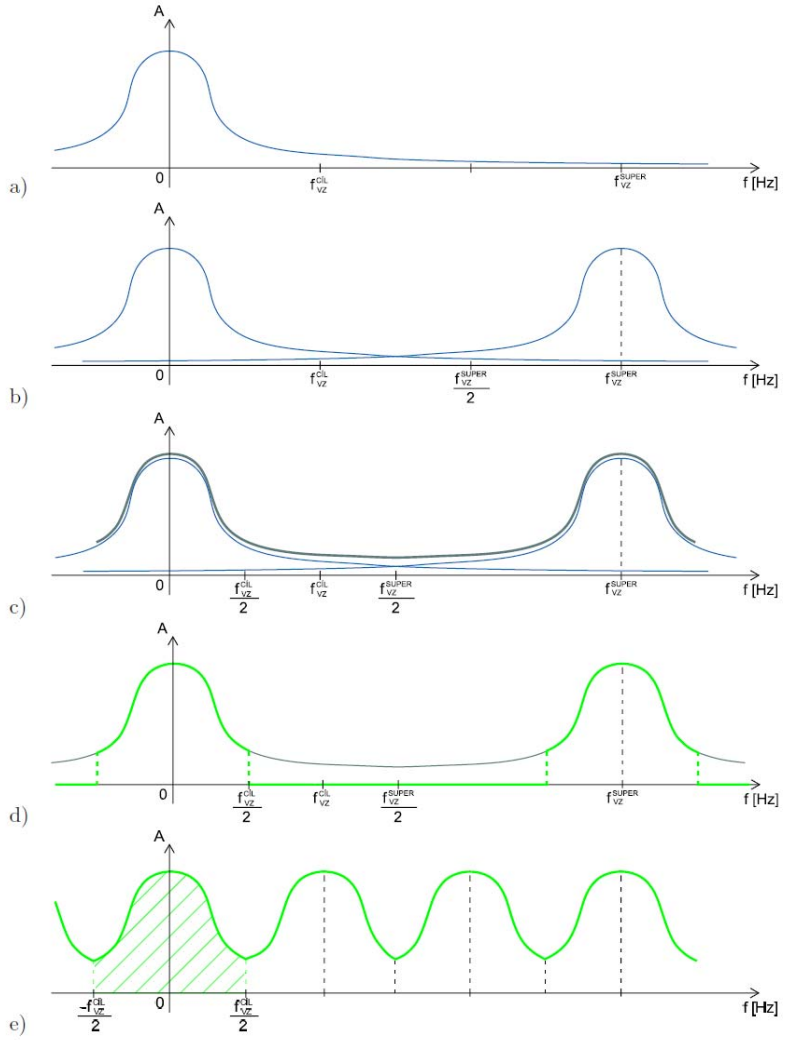
Vzorkování je převod signálu se spojitým časem na signál s diskrétním časem. U ideálního vzorkování jsou vzorky obrazu odebírány na nekonečně malé ploše. Bohužel u vzorkování nastává jev, který bezchybnou rekonstrukci obrazu znemožňuje, tzv. aliasing. Ten nastane, pokud obrazová funkce není kmitočtově omezená nebo není splněno tzv. Niquistovo kritérium, které je definováno vztahem

$$f_{vz} \geq 2 f_{\max}, \tag{1}$$

přičemž f_{vz} je vzorkovací kmitočet a f_{max} největší frekvence vzorkovaného signálu.

Vliv rozlišení obrazu	Vliv různého počtu odstínů šedé v obraze	
32x32 pixelů	2 odstíny šedé	4 odstíny šedé
		
64x64 pixelů	8 odstínů šedé	16 odstínů šedé
		
128x128 pixelů	32 odstínů šedé	64 odstínů šedé
		

Obr. 3: Ilustrace vlivu rozlišení a počtu zobrazitelných úrovní v obraze



Obr. 4: Schéma chování aliasingu při použití metody supersamplingu: a) modulové spektrum signálu se spojitým časem, b) zkopírování spektra při vzorkování s kmitočtem f_{vz}^{SUPER}, c) výsledné spektrum navzorkovaného diskrétního signálu – součet – pokud se kopie překrývají, vzniká aliasing, d) oříznutí spektra idealizovanou dolní propustí s mezním kmitočtem $f_{vz}^{CÍL}/2$, e) spektrum signálu pro převzorkování na kmitočet $f_{vz}^{CÍL}$, čerpáno z [5].

V reálném vzorkování není možné brát vzorek v nekonečně krátkém okamžiku. Teoretickým řešením vzniku aliasing je před vzorkováním uplatnit analogový tzv. antialiasingový filtr realizován jako doplní propust. V praxi však žádný filtr nedokáže aliasing potlačit úplně a proto se používá metoda tzv. supersampling. Analogový obraz je navzorkován s vyšším f_{vz}^{SUPER} , poté vyfiltrován diskrétní dolní propustí a pak teprve převzorkován na nižší cílový $f_{vz}^{CÍL}$. Kmitočet f_{vz}^{SUPER} musí být celočíselným násobkem $f_{vz}^{CÍL}$. Čerpáno z [5].

2.2 Kvantování

Diskretizace původně spojité škály hodnot signálu nebo převod diskrétního signálu na digitální nazýváme kvantování. Amplituda ve vzorkovaném obraze musí být pro zpracování počítačem vyjádřena jako digitální údaj. Počet kvantovacích úrovní musí být dostatečně velký, aby byly přesně vyjádřeny jemné detaily obrazu, nevznikaly falešné obrysy a aby se citlivost zařízení blížila citlivosti lidského oka. Většina systémů pro digitální zpracování obrazu používá kvantování do k stejných intervalů. Jestliže je pro reprezentaci informace o obrazovém elementu použito b bitů, je počet úrovní jasu $k=2^b$. Obvykle se používá 8 bitů na obrazový element. Největším problémem při kvantizaci je vznik falešných obrysů u obrazů kvantovaných do nedostatečného počtu jasových úrovní. Tento jev se stane člověku patrný pro počet jasových úrovní menší než 50, což je přibližný počet úrovní jasu, který je člověk schopen v monochromatickém obraze odlišit. Situaci lze poněkud zlepšit nelineárním kvantováním, které zvětšuje rozsah těch intervalů jasu, jejichž zastoupení není v obraze pravděpodobné, ale v praxi se používá jen zřídka. Tzv. kvantizační šum je rozdíl mezi hodnotou nekvantovaných a kvantovaných dat. Typický je pro něj trojúhelníkový průběh. Čerpáno z [5] a [1].

3. Reprezentace obrazu

Rozeznáváme dva základní způsoby, jak obraz prezentovat. Rastrově, resp. bitmapově, kdy je obraz definován určitým počtem obrazových bodů a každý bod nese informaci o určitém počtu bitů. Dle počtu bitů reprezentovaných v jednom pixelu rozlišujeme další druhy bitmapových obrazů. V druhém typu reprezentace je obraz popsán pomocí křivek, resp. matematicky popsatelných funkcí.

Monochromatický obraz - Tzv. binární obraz, jehož hodnota v každém pixelu může nabývat pouze dvou hodnot.

Stupně šedi - Hodnota pixelu vyjadřuje polohu jeho barvy na úsečce mezi bílou a černou.

True color - režim, kdy barva každého pixelu je reprezentovaná přímo hodnotami barevných složek.

Indexový mód - je spojen s použitím tzv. palety barev (palette, colormap). Hodnota pixelu není přímo jeho barva, ale ukazatel na pozici v paletě barev. Hodnota je obvykle reprezentována jedním bytem, což znamená maximální množství barev 256. Typ palety nazývaný *pseudo color* - paleta 8-8-8, tedy 3 B, 1 B/kanál. *Direct color* – pracuje se třemi hodnotami, přičemž každá pouze odkazuje na pozici v tabulce v barevné paletě, ovšem každá zvlášť pro jednotlivý barevný kanál. Vhodné např. při gamma korekci, přizpůsobování ICC profilu apod. - nepřepočítávají se hodnoty pixelů, nýbrž se pouze změní hodnoty RGB v tabulkách. Čerpáno z [5].

Vektorový formát - zatímco u bitmap je obraz prezentován množinou různobarevných pixelů umístěných v pravidelné rastrové mřížce, u vektorových formátu je obraz popsán analyticky jako množina geometrických tvarů. Rozlišujeme několik typů vektorových formátů a to dle toho, do jaké míry složitosti lze jednotlivé prvky – entity – matematicky popsat. Základní formáty zvládnou jen úsečky, ty složitější i oblouky, křivky či text a ty nejkomplexnější zvládnou hierarchické členění entit včetně možnosti jejich programového vytváření či modifikace (Postscript,SVG..). Dále pak také existují tzv. metaformáty, ve kterých se principy bitmapových a vektorových formátů kombinují (Postscript,Pdf…).

4. Předzpracování obrazu

Předzpracování obrazu je zcela nezbytnou operací, která se musí provést u všech zpracovávaných obrazů. To je dáno tím, že téměř nikdy se nepodaří pořídit

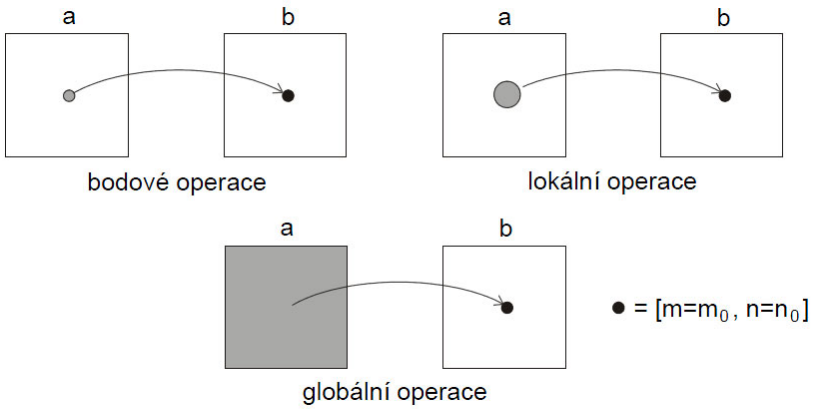
kvalitní obraz bez šumu, bez jakéhokoliv rušení, za optimálních podmínek a dalších možných vlivů. V rámci tohoto předzpracování se realizují takové operace, jako úprava jasu a kontrastu, úprava histogramu, průměrování několika za sebou jdoucích snímků za účelem potlačení šumu, různé druhy filtrací pro zaostření obrazu, pro odstranění rušení, operace zmenšení, zvětšení, otočení a posunu. Operací vztahujících se k předzpracování je celá řada a nelze zde uvést celý výčet. Souhrnně však lze říci, že předzpracování je tvořeno souborem základních postupů, které je nezbytné provést, abychom mohli vyhodnocovat informaci obsaženou v obraze, která má pro daný účel význam. Výstupem předzpracování pak může být např. obraz, jehož histogram je roztažen přes celý rozsah vstupních hodnot a neobsahuje žádné rušení. Čerpáno z [7].

4.1 Segmentace

Vyčlenění objektu zájmu neboli segmentace je dalším postupem, který má své významné postavení při zpracování obrazu. Jedná se o takové postupy, které zajistí, že se následující analýza obrazu bude vztahovat pouze k jeho určité části nebo částem, které přenášejí nejdůležitější informaci. Typickou metodou segmentace obrazu je prahování, detekce bodu, přímek, hran apod.

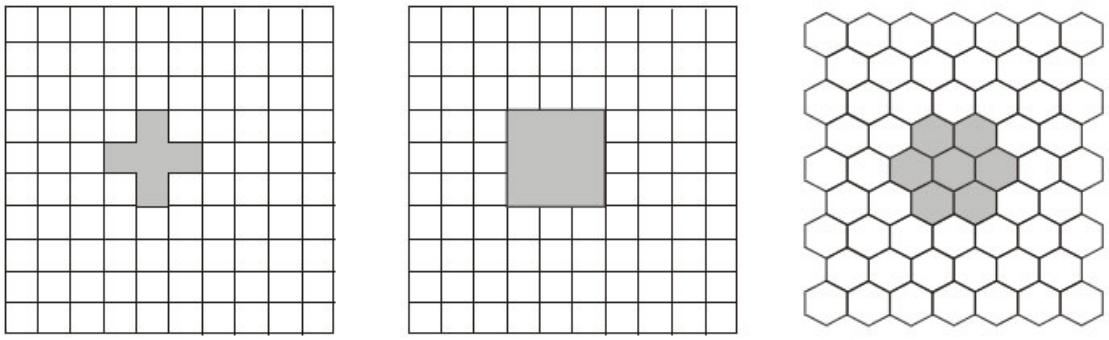
Typy operací, které mohou být aplikovány na digitální obraz můžeme rozdělit do tří skupin, které jsou zobrazeny na obr. 5, kde vstupní obraz je označen a[m,n] a výstupní obraz jako b[m,n]. Jedná se o následující operace:

1. bodové operace – výstupní hodnota pixelu s definovanými souřadnicemi je závislá pouze na hodnotě vstupního pixelu v téže poloze, ale ve vstupním obraze.
2. lokální operace – výstupní hodnota pixelu s definovanými souřadnicemi je závislá na hodnotách pixelů z definovaného okolí vstupních pixelů v téže poloze, ale ve vstupním obraze
3. globální operace – výstupní hodnota pixelu s definovanými souřadnicemi je závislá na všech hodnotách pixelů ve vstupním obraze. Čerpáno z [9].



Obr. 5: Kategorie operací aplikovaných na digitální obraz

Lokální operace jsou důležité z pohledu nejčastěji používaných operací v číslicovém zpracování obrazu. Záleží totiž na hledisku, jakým je obraz vzorkován, tj. z hlediska použití pravoúhlého či hexagonálního rastru a též z hlediska vazeb, které mohou být v závislosti na těchto rastrech použity v těsném okolí daného bodu, resp. sousedství. Uvedené příklady se uplatňují např. při konvoluční, mediánové filtraci ale i při morfologických operacích. Graficky je toto znázorněno na obr. 6. Hexagonální struktura se jeví jako optimální z hlediska využití prostoru, ale i z hlediska toho, že každý bod má stejnou vzdálenost od všech svých sousedů. Nicméně vlastní metody zpracování takových struktur nejsou zcela obvyklé a přináší některé obtíže. Tato struktura je mimo jiné totožná se strukturou oční sítnice. Čerpáno z [9].



Obr. 6: Pravoúhlý rastr 4 a 8 sousedství, hexagonální rastr 6 sousedství

4.2 Matematické a logické operace

Častým zpracováním obrazu je také použití základních **matematických operací** (sčítání, odečítání, násobení, dělení) a **logických operací** (logický součin, logický součet, doplněk a další odvozené). Zvláště aritmetické operace jsou východiskem pro další pokročilé a užitečné techniky počítačového zpracování obrazu.

Sčítání obrazů je nejpoužívanější aritmetickou operací z výše uvedeného výčtu. Vzhledem k tomu, že obraz můžeme považovat za matici dat, je sčítání dvou obrazů definováno tak, že prvek jednoho obrazu (matice) se sčítá v polohově odpovídajícím prvkem druhého obrazu (matice). Výsledkem je pak třetí obraz (matice), který je součtem předchozích dvou. To je však realizováno za základního předpokladu, že oba obrazy jsou tzv. registrovány (polohově sesouhlaseny). Může však dojít k situaci, kdy k původnímu obrazu chceme přičíst pouze konstantu. Situace je pak obdobná a výsledný obraz (matice), je pak vytvořen tak, že k původnímu obrazu, ke každému prvku takovéto matice, přičteme stejné číslo. V praxi to pak znamená, že přičtení konstanty odpovídá zvýšení jasu výsledného obrazu. Velmi užitečnou technikou pro zpracování obrazů s velkým šumem je průměrování. Čerpáno z [9].

Odčítání obrazů je obdobná operace jako sčítání. Tuto operaci lze opět použít pro ovlivnění jasu, tentokráte pro jeho snížení. Odečtením konstanty docílíme, že výsledný obraz bude celkově tmavší než vstupní.
Násobení obrazů je velmi užitečnou operací zvláště tam, kde chceme ovlivňovat kontrast obrazu (tj. násobení obrazu konstantou). V některých případech je možné využít násobení obrazu sebou samým, což může napomoci ke zvýraznění požadovaného rozsahu úrovní intenzit. Jedná se o násobení prvku jedné matice a polohově odpovídajícího prvku druhé matice. Čerpáno z [9].

Dělení obrazů je méně obvyklá operace, nicméně dělení obrazu konstantou je naopak velmi častá operace. Můžeme se s ní setkat jednak při ovlivňování kontrastu a jednak při tzv. normování dynamického rozsahu při vlastním

zobrazování (zachování původního rozsahu hodnot jako u vstupního obrazu). Čerpáno z [9].

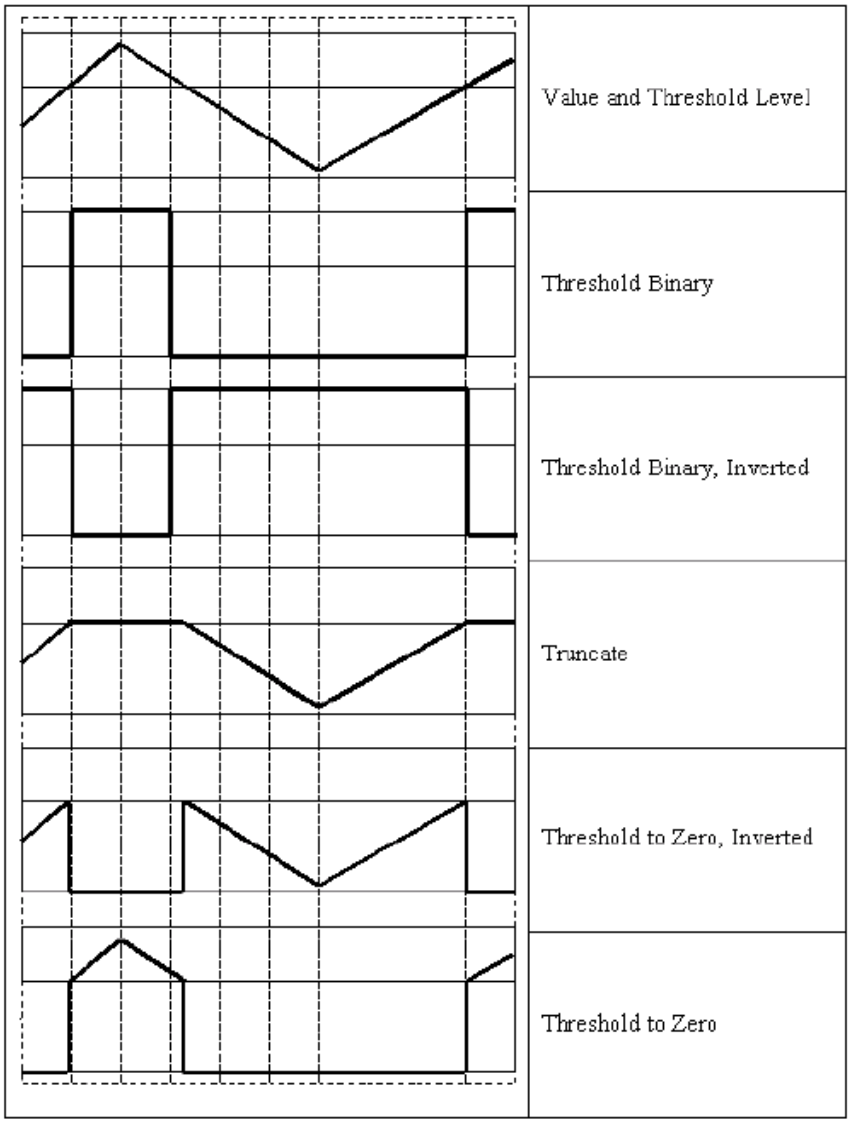
Nejzákladnějšími logickými operacemi používanými při zpracování obrazu jsou logický součin (AND – „a současně"), logický součet (OR – „nebo"), a doplněk resp. negace (NOT – „ne"). Základní rozdíl mezi aritmetickými a logickými operacemi je v tom, že aritmetické operace aplikujeme na šedotónové obrazy, zatímco logické operace především na dvouúrovňové obrazy (černobílé). Je to dáno tím, že u výše uvedených logických operací se jedná o využití dvojúrovňové logiky (tzv. Booleovské). Čerpáno z [9].

4.3 Prahování, erosion, dilatation

Prahování – segmentování na základě jasu. Pracuje na principu určité zvolené hodnoty jasu, přičemž všem pixelům pod touto hranicí je přiřazena hodnota 0, tzn. budou zobrazeny černě, a všem pixelům nad touto hranicí je přiřazena hodnota 1, tzn. bílá barva. Z uvedeného principu plyne malá výpočetní náročnost a tudíž i vysoká rychlost. Nevýhodou je vysoká degradace informací na snímku.

Erosion – aplikace na monochromatický obraz, používá se pro vyplnění celistvých tvarů jednou barvou, vhodné např. pro eliminaci nežádoucích pixelů vzniklých šumem.

Dilatation – opak filtru Erosion, zdůrazňuje části obrazu obsahující tvary. Čerpáno z [7].



Obr. 7: Různé typy prahování

5. Rozpoznávání objektů

Rozpoznávání objektů můžeme rozdělit na statické či dynamické. U statických metod je snímání pořizováno z jednoho místa. Statickou obrazovou scénu dělíme na dvě části – popředí a pozadí. Pozadí obrazu je definováno velice malou či žádnou změnou hodnot pixelů.

5.1 Metoda rozdílu snímku

Jak již název napovídá, princip je založen na odečítání jednoho nebo více po sobě jdoucích snímku. I když je tato metoda výpočetně vcelku nenáročná, je

samostatně nepoužitelná. Mezi její největší nedostatky patří její citlivost na volbu prahu v závislosti například na šumu nebo nasvětlení scény a pak, že se jako pixely popředí zobrazují pouze hodnoty na okraji objektu, přičemž hranice objektu bývá často neuzavřená. Čerpáno z [7].

5.2 Metody modelování pozadí

Často používaná metoda pro detekci pohybujících se objektů v obraze. Funguje na principu vytvoření modelu pozadí na úrovni pixelů. Ty vyjadřují nejpravděpodobnější barvu, nebo jas v daném okamžiku. Na základě změn daného pixelu v čase aproximuje jeho rozložení pravděpodobnosti. Princip pak spočívá v porovnávání aktuální hodnoty jednotlivých pixelů $I_n(x,y)$ s modelem pozadí reprezentovaného $B_n(x,y)$ a to podle následujícího vztahu:

$$I_n(x, y) - B_n(x, y) > T_h \qquad (2)$$

a tím určení zda daný pixel patří pozadí, nebo popředí. Parametr T_h je práh. Čerpáno z [7].

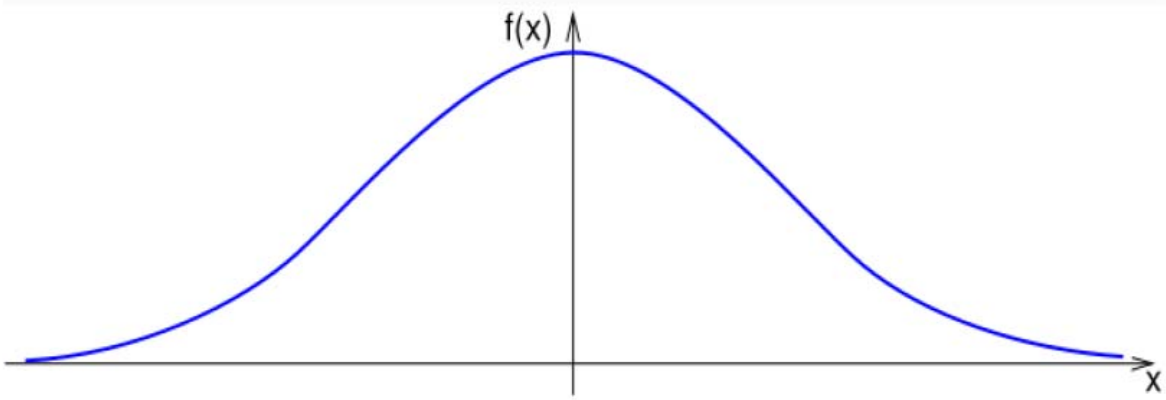
Modely pozadí se dají rozdělit podle časové tvorby pozadí na statické a dynamické. Rozdíl mezi statickým a dynamickým spočívá v tom, že zatímco u dynamického je model pozadí vypočítáván průběžně, tak ve statickém je toto provedeno pouze jednou, v první fázi učení. Statistické metody jsou jednoduché na matematické zpracování a s tím související zatížení procesoru. Na druhou stranu model pozadí se musí získávat jako sekvence snímku bez popředí a mají špatnou reakce na změnu osvětlení či na dlouho trvající změny scény. Dynamické lépe reagují na dlouho trvající změny scény a na změnu osvětlení. Nevýhodou je pak vyšší matematická náročnost. Čerpáno z [7].

5.3 Running gaussian average

Metoda vycházející z normálového rozložení pravděpodobnosti. Využívá se předpokladu, že hodnoty pozadí se náhodně mění přibližně podle normálového rozdělení. Pro každý pixel se vypočítá jeho střední hodnota a směrodatná odchylka. Pokud je hodnota pixelu uvnitř gaussovy křivky, jedná se o pixel

pozadí, v opačném případě se jedná o hodnotu popředí. Normálové rozložení pravděpodobnosti je dáno funkcí dle následujícího vzorce:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} . \tag{3}$$



Obr. 8: Funkce normálového rozložení pravděpodobnosti (gaussián) – čerpáno z [7].

Pokud je pixel klasifikován do pozadí, je střední hodnota přepočítána podle vzorce:

$$\mu_{t+1} = \alpha I_t(x,y) + (1-\alpha)\mu_t . \tag{4}$$

kde α je časová konstanta stanovená na základě dynamiky snímané scény. Parametr μ určuje střední hodnotu normálového rozložení v čase t a parametr I je hodnota aktuálního pixelu, nejčastěji hodnota jasu. Čerpáno z [7].

Pro určení gaussovy křivky je potřeba parametr směrodatné odchylky. Ta je přepočítávána podle vzorce:

$$\mu^2{}_{t+1} = \alpha(I_t(x,y) - \mu_t)^2 + (1-\alpha)\sigma_t^2 . \tag{5}$$

S každým novým pixelem, který nepatří do popředí, je přepočítána gaussovská křivka a tím i upravován model pozadí. Čerpáno z [7].

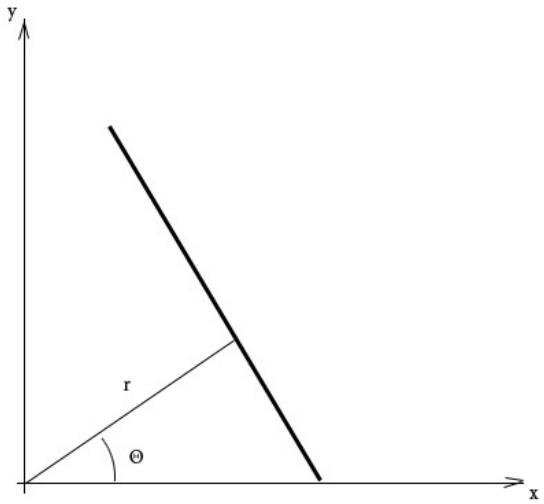
5.4 Mixture of gaussians

Vychází z předchozí metody a řeší některé její nedostatky. Princip metody je založen na modelování jednotlivých pixelu pozadí za pomocí několika gaussových rozložení (tzv.gaussiánu), které reprezentují vlastnosti jednotlivých pixelů pozadí. Tzn. že pixel není popsán jedním, nýbrž několika parametry, z čehož vyplývá, že tato metoda je náročnější na výpočetní výkon. Čerpáno z [7].

Houghova transformace je metoda, kterou můžeme řadit do dynamických metod rozpoznávaní objektů. Používá pro detekci hledaného tvaru parametrického popisu. Proto se používá pro hledání jednoduchých tvarů (přímky, kružnice apod.), resp. pro segmentaci objektů, jejichž hranice lze popsat jednoduchými křivkami. Výhodou této metody je robustnost vůči nepravidelnostem či porušení hledané křivky. Vstupem pro Houghovu transformaci bývá snímek, nejlépe s omezeným počtem úrovní, přičemž výsledkem je potom poloha a orientace hledaných tvarů.
Čerpáno z [3].
Princip metody si můžeme vysvětlit na detekci přímky v obraze. Houghova transformace pracuje buď se směrnicovým nebo normálovým popisem. V našem případe použijeme pro model přímky rovnici z [8]

$$x\cos\Theta + y\sin\Theta = r\,, \tag{6}$$

kde r je délka normály od přímky k počátku souřadnic, Θ je úhel mezi normálou a osou x, viz. obr. 6.1

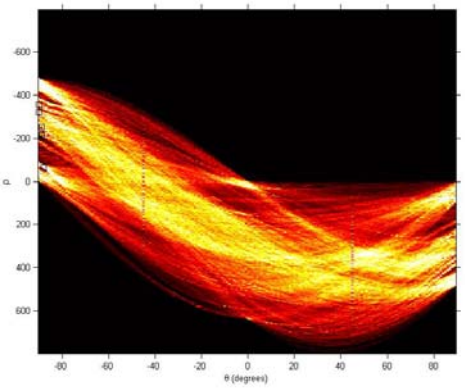


Obr. 9: Parametrický popis přímky

Pokud do předchozí rovnice dosadíme souřadnice nějakého bodu (x_i,y_i), pak množina všech možných řešení (r,Θ) vytvoří v Houghově prostoru spojitou křivku. Jestliže si tímto způsobem promítneme do Houghova prostoru všechny body ležící na nějaké přímce p, pak uvidíme, že křivky odpovídající jednotlivým bodům (x_i,y_i) se protnou v jediném bodě (r_{max}, Θ_{max}). Tato dvojice jsou námi hledané parametry přímky p. Čerpáno z [3].

Houghova transformace je v praxi implementována tak, že v prvním kroku je Houghův prostor diskretizován. Každý element tohoto prostoru, který nazýváme akumulační buňka, je vynulován. Každý bod (x_i,y_i) je transformován do diskretizované křivky (r, Θ), přičemž hodnota akumulačních buněk podél této křivky je inkrementována. Souřadnice maxima v akumulační rovině (r_{max}, Θ_{max}) jsou parametry hledané přímky v obraze. Čerpáno z [2].

a)



b) c)

Obr. 10: *a)* fotografie auta , *b)* aplikace Houghovy transformace v Matlabu a *c)* příklad hledání přímek Houghovou transformací

6. Určení rychlosti pohybujících se objektů.

Nyní se zaměříme na obecný způsob, jak vůbec určit rychlost nějakého pohybujícího se objektu. Vycházejme z předpokladu, že k dispozici máme detektor – fotoaparát, kameru - a známe pouze čas mezi jednotlivými pořízenými snímky. Základní vzorec pro určení rychlosti říká, že rychlost lze vypočítat poměrem mezi ujetou vzdáleností a časem, za který tuto vzdálenost objekt urazil.

$$v = \frac{s}{t}\,. \tag{7}$$

Z uvedeného vyplývá, že potřebujeme nějakým způsobem určit výše zmíněnou ujetou vzdálenost. Nabízí se porovnávání velikosti detekovaného objektu

s nějakou, předem danou hodnotou. Např. porovnávání velikosti automobilu, koncových světel apod. Zde ovšem nastává problém. Každé auto je specifické, má svoje rozměry, rozměry světel, oken apod. Naštěstí pro nás, na každém automobilu, které používá pozemní komunikace ČR v souladu se zákonem, najdeme jeden objekt, který je normovaný, tzn. má pevně danou velikost, a tou je státní poznávací značka, resp. tabulka s registrační značkou, jak zní její aktuální definice.

Ze stránek Ministerstva dopravy České republiky vyplývá, že existují rozměrově dva základní typy tabulek s registrační značkou lišících se pouze barevným provedením podkladu, případně písma např. pro nákladní vozidla, traktory, apod.. My se zaměříme na typ tabulky užívaný nejčastěji. Velikost tabulky s registrační značkou základního typu 101 uvádí následující obrázek:

Obr. 11: Typ tabulky registrační značky 101, její rozměr
zdroj: internetové stránky Ministerstva dopravy ČR

V praxi se můžeme setkat ještě s druhým, starším typem tabulky, který neobsahuje modrý pruh Evropské Unie, rozměrově se ale shoduje. Navíc pevně daný rozměr značky pro nás není až tak důležitý (viz dále).

6.1 Odhad vzdálenosti auta od objektivu

Pomocí fotoaparátu Canon typu IXUS 55 jsem nafotil několik zkušebních snímků z různých vzdáleností, abych zjistil, jak se mění velikost značky v závislosti na vzdálenosti od fotoaparátu. Nejprve jsem zvolil základní nejmenší rozlišení fotoaparátu tj. 640x480 bodů.

a **b** **c**

Obr. 12: Vzdálenost auta od objektivu a) 1m, b) 4m, c) 15m

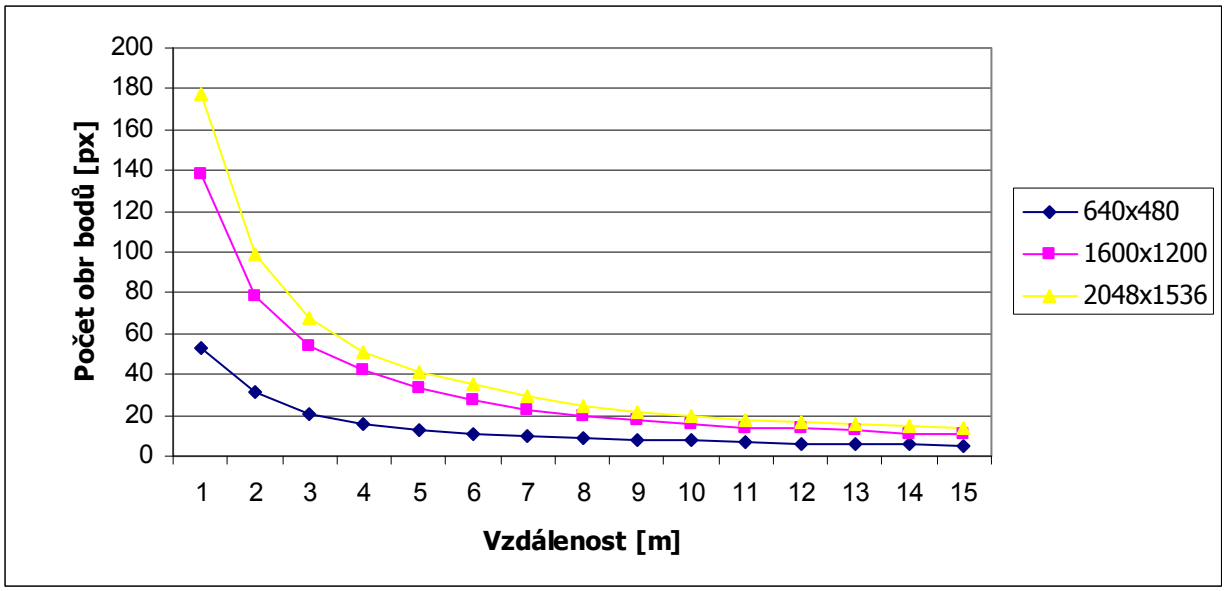
Ovšem jak ukazuje následující obrázek, toto rozlišení je nedostatečné, neboť při již relativně krátké vzdálenosti, tj. 7m od fotoaparátu, zaujímala tabulka s registrační značkou v digitální podobě výšku pouhých 10 pixelů, přičemž další vzdalování objektu mělo jen minimální vliv na změnu šířky tabulky v pixelech.

Obr. 13: Vzdálenost automobilu 7m od objektivu a zvětšenina tabulky s registrační značkou na úroveň pixelů

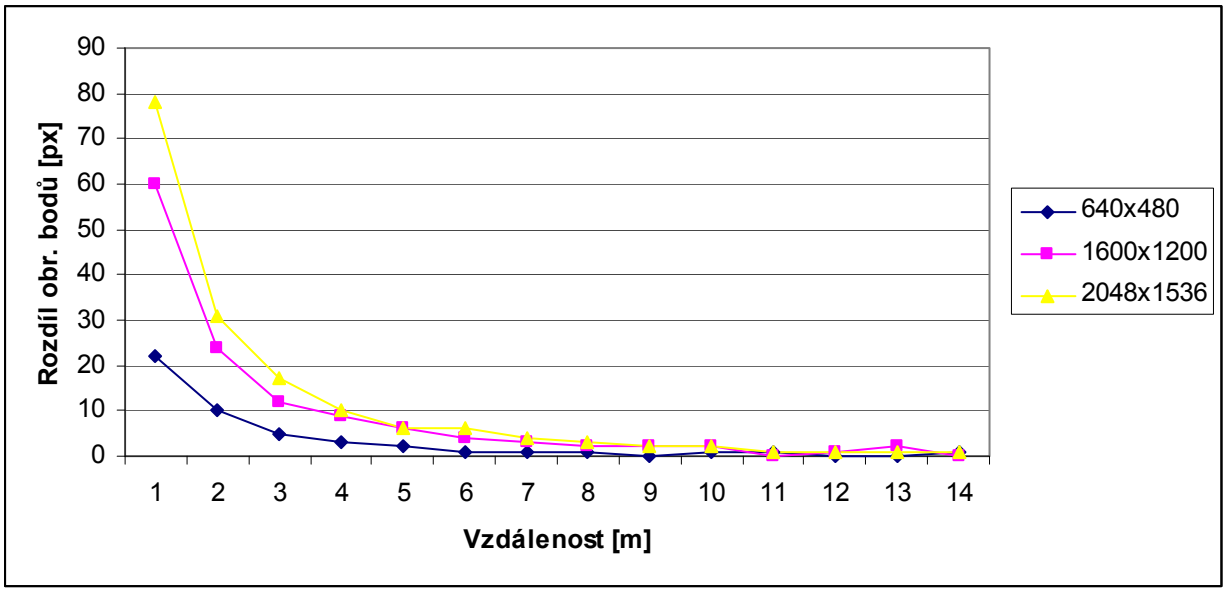
Vzdálenost [m]	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Výška tabulky [px]	53	31	21	16	13	11	10	9	8	8	7	6	6	6	5

Tab. 1: Počet pixelů na výšku v tabulce s reg. značkou v závislosti na vzdálenosti

Na základě tohoto zjištění jsem pořídil další sérii fotografií s rozlišením 1600x1200 bodů a 2048x1536 bodů. Následující obrázek demonstruje grafické závislosti počtu pixelů určujících šířku tabulky registrační značky a matematický rozdíl v počtu pixelů v závislosti na vzdálenosti od objektivu.



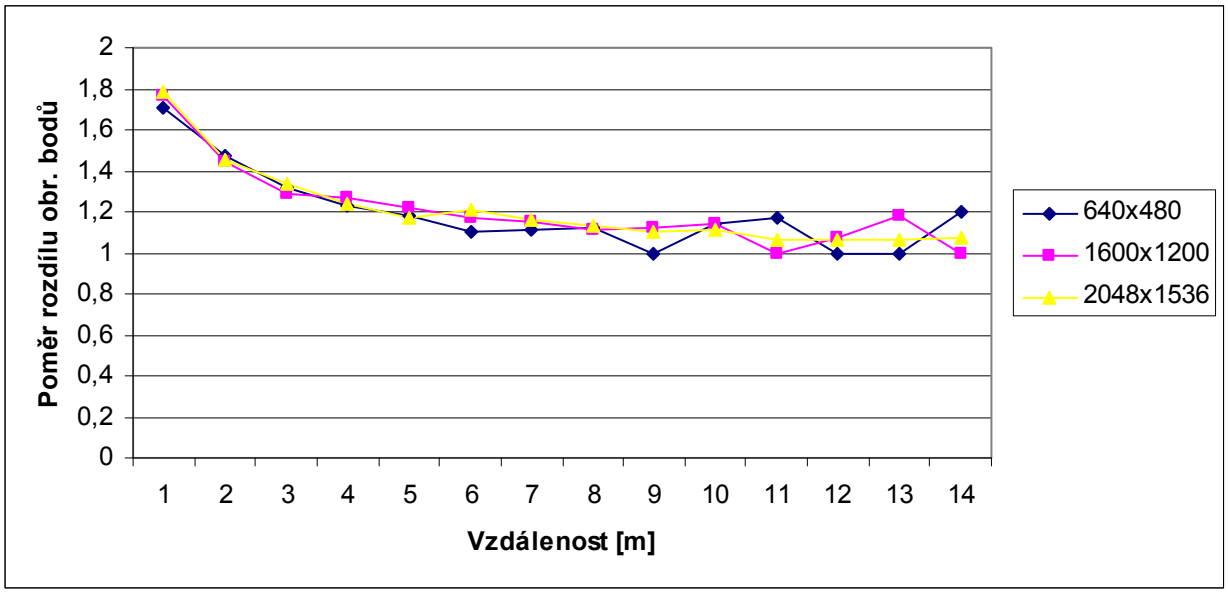
Obr. 14: Počet pixelů určujících šířku tabulky registrační značky v závislosti na vzdálenosti od objektivu pro jednotlivá rozlišení fotoaparátu



Obr. 15: Matematický rozdíl v počtu pixelů v závislosti na vzdálenosti od objektivu pro jednotlivá rozlišení fotoaparátu

Z uvedených grafů plyne, že pro vzdálenosti větší než 7m je rozdíl v počtu pixelů pro každou vzdálenost téměř konstantní. To je pro nás nepříjemné, neboť je zde vysoké riziko vzniku chyby při určování vzdálenosti a tedy pro vzdálenost automobilu větší než 7m jsou takové snímky v podstatě nepoužitelné. Tento problém by se mohl vyřešit např. použitím přídavné optické soustavy, která by celkovou scénu, resp. objekty v ní, zvětšila.

Uvedené hodnoty pixelů pro jednotlivé rozlišení fotoaparátu jsem ještě porovnal z hlediska poměru mezi pixely v závislosti na vzdálenosti.



Obr. 16: Poměr rozdílu obr. bodů v závislosti na vzdálenosti od kamery

Tento graf je pro nás mnohem zajímavější než předchozí. Z uvedeného grafu totiž plyne, že bez ohledu na velikost objektu, v našem případě reprezentovaným počtem pixelů, je poměr změny velikosti v závislosti na vzdálenosti vždy stejný. Toho můžeme s výhodou využít při implementaci.

7. Implementace

Celá problematika byla v praxi implementována na pořízené snímky pomocí knihovny OpenCV v jazyce C++. OpenCV (Open source Computer Vision) je svobodná a otevřená multiplatformní knihovna pro práci a manipulaci s obrazem původně vyvíjená firmou Intel. Je zaměřena na počítačové vidění a zpracování obrazu. Obsahuje množství algoritmů a metod pro zpracování obrazu. Je k dispozici pro operační systémy Windows i Linux, knihovnu lze přímo používat z C a C++ a pomocí wrapperů i z dalších jazyků, jako je Python. [10]

Na zpracovávaný obraz je z hlediska předpzracování obrazu aplikován filtr prahování.

Obr. 17: Původní snímky a snímky po aplikaci filtru prahování

Funkce prahování v OpenCV umožňuje 6 druhů prahování, přičemž v rámci testování se nejlépe osvědčil typ CV_THRESH_TOZERO demonstrovaný na obrázku 17. Pro samotnou detekci tvarů je pak využíváno funkce pro detekci tvarů cvFindContours() a pomocných funkcí pro bližší specifikaci hledaného tvaru.

V našem případě se jedná o specifický tvar obdélníku, tedy tyto funkce blíže specifikují vlastnosti tvaru na obdélník. Využívá se výpočtu cosinu úhlu mezi jednotlivými stranami detekovaného tvaru tak, že pokud je cosinus úhlu blízko nule, má tento úhel cca 90°, tedy je pravoúhlý.

Následující obr. 18 znázorňuje aplikaci detekce tvarů na pořízené snímky. Z obrázku je patrné, že detekovaných tvarů je mnohem více než jenom námi hledaná poznávací značka. Je tedy nutné ještě blíže specifikovat, které objekty jsou námi hledané a vykreslit je. Využívá se jednoduché funkce pro určení vzdálenosti mezi dvěma body. Počítáme výšku a šířku tvaru a na základě poměru těchto dvou stran pak tvar vykreslíme a nebo nikoliv, pokud poměr stran značky se pohybuje okolo 4,5. Program doplněný o tuto podmínku rozhodování pak vykreslil detekované tvary již téměř přesně, jak znázorňuje obr. 19

Obr. 18: Aplikace základní detekce tvaru

Obr. 19: Finální detekce tvaru ze vzdálenosti 1 až 7m

8. Závěr

Problematika zpracování obrazu, respektive detekce objektů v něm, je natolik rozsáhlá, že pro jednotlivé případy je potřeba hledat jiný přístup v závislosti na daných podmínkách, který mi jsou např. použitá optická soustava pro detekci, použitelný výpočetní výkon pro zpracování a s tím související třeba i ekonomické náklady pro zpracování.

V mém zadání bylo úkolem zjistit, jakým způsobem můžeme určit rychlost přibližování auta jedoucího před námi za předpokladu, že v autě máme umístěnou kameru, kterou pořizujeme snímky. Jelikož známe čas mezi snímky a známe počet pixelů registrační značky, resp. poměry mezi vzdálenosti od objektivu, stačí již jen registrační značku na snímku obrazu detekovat.

Každá tabulka s registrační značkou používaná v souladu s vyhláškou o provozu na pozemních komunikacích 361/2000 Sb. je černá, tzv. na snímku je značka ostře ohraničená. To jsem mohl s výhodou využít při následné implementaci programu k detekci registrační značky. Na obraz byl aplikován filtr prahování a dále zpracován funkcemi knihovny OpenCV (zdrojový kód použitého skriptu je uveden v příloze). Bohužel jsem již nestihl implementovat algoritmus na videosekvenci a zpracovávat tak obraz v reálném čase. Navíc algoritmus i přes použité pomocné funkce není zcela přesný. Z testovaných snímků do vzdálenosti cca 6-7 metrů správně detekoval značku z 95 procent, na delší vzdálenost již nebyl schopen značku rozpoznat. Nicméně pro samotné určení vzdálenosti to již nehraje velkou roli. Z předchozí kapitoly totiž vyplývá, že nezáleží na použitém rozlišení pořízených ani na velikosti detekovaného objektu z hlediska počtu pixelů obsažených v detekovaném tvaru. Testování proběhlo jak na snímcích o rozlišení 640x480, tak i na snímcích s vyšším rozlišením, přičemž pro samotnou detekci rozlišení nemělo žádný větší vliv. Změna rozlišení se projevila především v rychlosti zpracování, kde snímky s rozlišením 1600x1200 byly zpracovávány znatelně pomaleji.

LITERATURA

[1] *Digitalizace obrazu* [online]. [2008] [cit. 2009-11-20]. Dostupný z WWW: <http://e-learning.tul.cz/cgi- bin/elearning/ elearning.fcgi? ID_tema=67& ID_obsah=1176& stranka=publ_tema&akce=polozka_vstup>

[2] DOBEŠ, Michal. *Zpracování obrazu a algoritmy v C#*. Praha : BEN, 2008. 144 s. ISBN 978-80-7300-233-6.

[3] *Houghova transformace* [online]. 2008 , 2008 [cit. 2009-11-05]. Dostupný z WWW: <http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv5/hough.htm>.

[4] Piccardi, M. Background subtraction techniques: a review,Sydney, University of technology,2004,30s. Dostupný z WWW: <http://www-staff. it. uts. edu.au/ ~massimo/BackgroundSubtractionReview-Piccardi.pdf>

[5] RAJMIC, Pavel. *Multimediální a gracké procesory*. [s.l.] : [s.n.], 2009. 120 s.

[6] ŘÍHA, Kamil. *Pokročilé techniky zpracování obrazu*. [s.l.] : [s.n.], 2007. 102 s.

[7] ŠMIRG, Ondřej. *Detekce lidské postavy v obrazové scéně*. [s.l.], 2008. 56 s. VUT FEKT. Vedoucí diplomové práce Ing. Michal Kohoutek.

[8] *Přímka - příklady* [online]. [2006] [cit. 2009-12-12]. Dostupný z WWW: <http://www.aristoteles.cz/matematika/analyticka_geometrie/primka/primka.php>.

[9] Hozman, J., Bernas, M., Klíma, M., Dvořák, P.: *Zpracování obrazové informace.* (Skriptum). České vysoké učení technické v Praze. Vydavatelství ČVUT: Praha, 1996. 177 s

[10] OpenCV [online]. 2010 [cit. 2010-05-26]. Dostupné z WWW: <http://opencv.willowgarage.com/wiki/>.

Seznam použitých zkratek

ICC	International Color Consortium
RGB	Barevný model skládající se ze složek červená, zelená, modrá
ASM	Aktive shape models, model pro určování objektů v obraze
PCA	Algoritmus používající ASM

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1: Zdrojový kód

```
#ifdef _CH_
#pragma package <opencv>
#endif

#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <math.h>
#include <string.h>

int thresh = 50;
IplImage* img = 0;
IplImage* img0 = 0;
CvMemStorage* storage = 0;
const char* wndname = "Detekce znacky";

int length(CvPoint a, CvPoint b)
{ //hleda vzdalenost mezi 2 body
    //return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
    return a.x-b.x;
}
// pomocná fce hledající cosinus uhlu mezi
// vectory z bodu pt0->pt1 a pt0->pt2
double angle( CvPoint* pt1, CvPoint* pt2, CvPoint* pt0 )
{
    double dx1 = pt1->x - pt0->x;
    double dy1 = pt1->y - pt0->y;
    double dx2 = pt2->x - pt0->x;
    double dy2 = pt2->y - pt0->y;
    return (dx1*dx2 + dy1*dy2)/sqrt((dx1*dx1 + dy1*dy1)*(dx2*dx2 + dy2*dy2) +1e-10);
}



// fce ktera vraci sekvenci detekovanych ctvercu
CvSeq* findSquares4( IplImage* img, CvMemStorage* storage )
{
    CvSeq* contours;
    int i, c, l, N = 11;
    double p;
    //pocatecni inicializace, vytvoreni novych obrazu z puvodniho
    CvSize sz = cvSize( img->width & -2, img->height & -2 );
    IplImage* timg = cvCloneImage( img );
    IplImage* gray = cvCreateImage( sz, 8, 1 );
    IplImage* pyr = cvCreateImage( cvSize(sz.width/2, sz.height/2), 8, 3 );
    IplImage* tgray;
    CvSeq* result;
    double s, t;

    //vytvori prazdnou sekvenci obsahujici body
    // 4 body na ctverec
    CvSeq* squares = cvCreateSeq( 0, sizeof(CvSeq), sizeof(CvPoint), storage );

    // nastaveni max. ROI (region of interest) v obraze

        cvSetImageROI( timg, cvRect( 0, 0, sz.width, sz.height));



    // zmenšení a zvìtšení obrazu k vyfiltrování šumu
    cvPyrDown( timg, pyr, 7 );
    cvPyrUp( pyr, timg, 7 );
    tgray = cvCreateImage( sz, 8, 1 );

    // hledání obrysù v každé složce barev
    for( c = 0; c < 3; c++ )
    {
        // extrahování c-té barevné složky
        cvSetImageCOI( timg, c+1 );
        cvCopy( timg, tgray, 0 );


        for( l = 0; l < N; l++ )
        {
```

```
              // aplikování prahování
               cvThreshold( tgray, gray, (l+1)*255/N, 255, CV_THRESH_TOZERO);

            // hledání obrysu
            cvFindContours( gray, storage, &contours, sizeof(CvContour),
                CV_RETR_LIST, CV_CHAIN_APPROX_NONE, cvPoint(0,0) );

            // testovani kazdeho obrysu
            while( contours )
            {

                // vraci aproximovane obrysy priblizne odpovidajici
                // hledanym proporcim
                result = cvApproxPoly( contours, sizeof(CvContour), storage,
                    CV_POLY_APPROX_DP, cvContourPerimeter(contours)*0.03, 0 );

                // obrysy by meli mit 4 hrany po aproximaci
                // a meli by byt konvexni

                if( result->total == 4 &&
                    fabs(cvContourArea(result,CV_WHOLE_SEQ)) > 1 &&
                    cvCheckContourConvexity(result) )
                {
                    s = 0;

                    for( i = 0; i < 5; i++ )
                    {
                         // hleda minimalni uhel mezi stranami
                        // (maximalni kosinus)
                        if( i >= 2 )
                        {
                            t = fabs(angle(
                            (CvPoint*)cvGetSeqElem( result, i ),
                            (CvPoint*)cvGetSeqElem( result, i-2 ),
                            (CvPoint*)cvGetSeqElem( result, i-1 )));

                            s = s > t ? s : t;

                        }
                    }

                    // jestlize je cosinus maly, je uhel okolo 90 stupnu

                    if( s < 0.3 ) {
                        for( i = 0; i < 4; i++ ) {

                                cvSeqPush( squares,
                                (CvPoint*)cvGetSeqElem( result, i ));

                                } }
                }

                // vemzi dalsi obrys pro zpracovani
                contours = contours->h_next;
            }
        }
    }

    // uverejneni vsech obrazu
    cvReleaseImage( &gray );
    cvReleaseImage( &pyr );
    cvReleaseImage( &tgray );
    cvReleaseImage( &timg );

    return squares;
}



//fce vykreslujici vsechny obrysy
void drawSquares( IplImage* img, CvSeq* squares )
{
    CvSeqReader reader;
    IplImage* cpy = cvCloneImage( img );
    int i;
    double cx[2];
    double pomer;
```

```
    // inicializace readeru
    cvStartReadSeq( squares, &reader, 0 );

    // naèti 4 strany obdelniku

    for( i = 0; i < squares->total; i += 4 )
    //for( i = 0; i = 2; i += 4 )
    {
        CvPoint pt[4], *rect = pt;
        int count = 4;


        CV_READ_SEQ_ELEM( pt[0], reader );
        CV_READ_SEQ_ELEM( pt[1], reader );
        CV_READ_SEQ_ELEM( pt[2], reader );
        CV_READ_SEQ_ELEM( pt[3], reader );

        //vrati delku obdelniku
        cx[0]=length(pt[2],pt[1]);
        //vrati sirku obdelniku
        cx[1]=length(pt[1],pt[0]);
        cx[2]=length(pt[3],pt[2]);

        double pomer=abs(cx[0]);



        // kresli obdelnik a ukaz delku v pixelech
        // v pripade ze, delka je vetsi 4 krat
       if((cx[0] > (cx[1]*4)) && (cx[2] > (cx[2]*4)) && (cx[0] > (cx[1]*4)) &&(cx[1]>0) )
{
             printf("(%f)\n", pomer);
           cvPolyLine( cpy, &rect, &count, 1, 1, CV_RGB(0,255,0), 1, CV_AA, 0 );
         }

    }

    // ukaz vysledny obrazek
    cvShowImage( wndname, cpy );
    cvReleaseImage( &cpy );
}


char* names[] = { "auta1.JPG", "auta2.jpg",
"auta3.jpg","auta4.jpg","auta5.jpg","auta6.jpg","auta7.jpg","auta8.jpg", 0 };

int main(int argc, char** argv)
{
    int i, c;
    // vytvori ulozny prostor pro detekovane tvary
    storage = cvCreateMemStorage(0);

    for( i = 0; names[i] != 0; i++ )
    {
        // load i-th image
        img0 = cvLoadImage( names[i], 1 );
        if( !img0 )
        {
            printf("Couldn't load %s\n", names[i] );
            continue;
        }
        img = cvCloneImage( img0 );


        cvNamedWindow( wndname, 1 );

        // najdi a kresli obdelnikz
        drawSquares( img, findSquares4( img, storage ) );


        c = cvWaitKey(0);

        cvReleaseImage( &img );
        cvReleaseImage( &img0 );

        cvClearMemStorage( storage );
        if( (char)c == 27 )
```

```
            break;
        }

        cvDestroyWindow( wndname );

        return 0;
    }
```