UNIVERZITA PARDUBICE
DOPRAVNÍ FAKULTA JANA PERNERA
KATEDRA ELEKTROTECHNIKY, ELEKTRONIKY A ZABEZPEČOVACÍ TECHNIKY

ORIENTACE VOZIDLA POMOCÍ VESTAVĚNÉ OPTICKÉ KAMERY

DISERTAČNÍ PRÁCE

AUTOR PRÁCE: Ing. Josef Šroll
ŠKOLITEL: prof. Ing. Vladimír Schejbal, CSc.

2009

UNIVERSITY OF PARDUBICE
JAN PERNER TRANSPORT FACULTY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING AND SIGNALING IN TRANSPORT

ORIENTATION VEHICLES USING EMBEDDED OPTICAL CAMERA

DISSERTATION

AUTHOR: Ing. Josef Šroll
SUPERVISOR: prof. Ing. Vladimír Schejbal, CSc.

2009

Prohlašuji:

Tuto práci jsem 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 se 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 Univerzity Pardubice.

V Pardubicích dne 31. 8. 2009

Ing. Josef Šroll

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě
Akademický rok: **2007/2008**

ZADÁNÍ DISERTAČNÍ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ing. Josef ŠROLL**
Studijní program:	**P3710 Technika a technologie v dopravě a spojích**
Studijní obor:	**Dopravní prostředky a infrastruktura**
Název tématu:	**Orientace vozidla pomocí vestavěné optické kamery**

Zásady pro vypracování:

Základní myšlenkou je nasnímání obrázků optickou kamerou při prvním průjezdu vozidla danou trasou, např. s řidičem. Při dalších průjezdech se vozidlo orientuje na trase již samo pomocí dat získaných při prvním průjezdu porovnáním s daty aktuálními. V těch částech trasy, kde není jednoznačná identifikace polohy tímto způsobem, bude poloha určována vyhodnocením aktuální obrazové informace z kamery bez identifikace jejího obsahu jen pro získání rychlosti dopředné a úhlové. S prvotním nasnímáním obrázků se současně nebo dodatečně k obrázků uloží souřadnice. Požaduje se,aby řešený systém dokázal na základě aktuálního obrazu z kamery a obrázků uložených v předchozí jízdě určit souřadnice místa, kde se právě vozidlo s kamerou nalézá. Odvodit příslušné algoritmy, pomocí nichž je možno získat souřadnice vozidla porovnáním obrazu z optické kamery a obrazy se souřadnicemi dříve uloženými.
Cíle disertace jsou:
Najít vhodnou metodu pro určení posice vozidla opakovaně se pohybujícího po určité trase pomocí optické kamery
Odvodit příslušné algoritmy, pomocí nichž je možno získat souřadnice vozidla porovnáním obrazu z optické kamery a obrazy se souřadnicemi dříve uloženými.
Testovat odvozené algoritmy jako důkaz jejich správnosti
Vytvořené algoritmy realizovat v jazyce C++ ve formě knihovny a zpřístupnit ji.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování disertační práce: **tištěná**

Seznam odborné literatury:

[1] **SONKA,Milan. - Hlavac, Václav. Image Processing, Analysis, and Machine Vision. Roger Boyle, PWS Boston 1999, 2nd edition; ISBN 0-534-95393-X**

[2] **PRATT W. Digital image processing (3ed., Wiley, 2001. 738 s. ISBN 0471374075.**

[3] **ZHIGANG Zhu. - GUANGGYOU, Xu, BO Yang. – DINJI, Shi. – XUEIN, Lin. VISATRAM: a real-time vision system for automatic traffic monitoring. Image and Vision Computing 18 (781 - 794)**

[4] **ESCALERA, A. - ARMINGOL, J. M. – MATA, M. Traffic sign recognition and analysis for intelligent vehicles Image and Vision Computing 21 (247-258)**

[5] **STILLER, C. – HIPP, J. – ROSSING, C. – EWALD, A. Multisensor obstacle detection and tracking. Image and Vision Computing 18 (389 – 396)**

Vedoucí disertační práce: **prof. Ing. Vladimír Schejbal, CSc.**
Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě

Datum zadání disertační práce: **3. října 2002**

Termín odevzdání disertační práce: **31. srpna 2008**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

prof. Ing. Vladimír Schejbal, CSc.
vedoucí katedry

V Pardubicích dne 31. srpna 2008

SUMMARY

The dissertation contains an analytical section and an original section. The aim was to find suitable algorithms that help to find a vehicle coordinates based on the optical camera images. It is assumed that the vehicle firstly drives through a determined route creating a sequence of optical camera images that will be complemented by relevant coordinates. At repeated drives-through, the system already compares the optical camera images with those saved at the first drive-through. Based on this comparison, current vehicle coordinates can be determined. The task appeared to be much more demanding then it looked in the beginning. This problem has probably not been dealt with, therefore it was impossible to find the required support. The originally expected openCV programme libraries assistance has not been helpful. For this reason, it was necessary to solve numerous other problems to be able to complete the original task. However, new algorithms were found, that have not been solved so far.

The set goal has been reached. The algorithms derived in this paper were launched into operation at a small vehicle model (using a children building box) with a web camera (Herkules de Luxe). Regarding extensive cost of a bit-maps processor, the vehicle can only move quite slowly to enable the computer to process the camera images timely. Speed increase is only a matter of the processor equipment, either by its speed increase or by a parallel run of more processors. The later solution is feasible, which is referred to in the paper.

The application was programmed in C++ language. The Borland C++ Builder compiler was used for this purpose. Standard libraries of the compiler are already equipped with programmes for JPEG images conversion into bit-maps.

The above mentioned facts demonstrate that the task has been fully completed, and so far unpublished theories and algorithms have been created as a by-product

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

Zkratka	Význam
C/C++	Programovací jazyky; C++ je určen pro objektové programování
C++ Builder	Použité vývojové prostředí fy Borland
DCT	Diskrétní kosinová transformace
dopravní cesta	Původně spojnice dvou uzlů; zde obecněji – nemusí se jednat o klasickou komunikaci (silnice, železnice), ale o obecně zamýšlenou trasu, která může vést i volnou přírodou či uvnitř jiné struktury (kanály, tunely)
GPS	Global Position System – americký družicový systém pro určování polohy
GALILEO	Evropský družicový systém pro určování polohy (ve výstavbě)
HSV	Hue, Saturation, Value – obrazový systém potlačující stíny
INS	Inerciální navigační systém
JPEG	Joint Photographics Experts Group – ztrátová komprimační metoda obrazů, založená na DCT transformaci
openCV	Knihovna algoritmů počítačového vidění fy Intel
pixel	Jeden obrazový bod bitové mapy
RGB	Systém barevného obrazu, kde základními barvami je červená(Red), zelená (Green) a modrá (Blue)
TColor	Formát bitové mapy používaný v C++ Builderu
Markant	Výrazný detail v obrazu vhodný k dalšímu zpracování

1 ÚVOD

Určování polohy je základní požadavek při řízení vozidla. Klasická metoda je prostřednictvím vnímání osoby ovládající vozidlo. Při pokusech nahradit tohoto řidiče je jedním z rozhodujících faktorů nahrazení jeho vnímání jinými možnostmi. U osob se jedná převážně o vnímání světelných informací zrakem, méně již dalšími smysly - okolním hlukem např. v tunelech, vibracemi vozidla vyvolané nerovnostmi vozovky a pod. Pro určování polohy se používají různé technické prostředky, založené na různých principech. Podle toho mají různou přesnost měření a různou použitelnost. Systémy pracující s radiovými signály (LORAN, GPS) lze použít pouze v prostředí pro tyto signály dostupné, a to bez odrazů. Inerciální navigační systémy sice nejsou závislé na vnějším prostředí, ale vzniká u nich chyba, která se časem zvětšuje (kumuluje).

Předpokládá se použití kamery s výhledem před vozidlo, případně boční kamera s výhledem do strany. V tomto zorném poli se nachází část vozovky, oblast, ve které by se měl nacházet horizont a různě velké předměty, které se nachází v okolí cesty.

Jedná se o rozsáhlou problematiku. Řešený systém nejspíš nikdy nebude dominantním navigačním systémem, může však být nenahraditelný v některých specifických případech, jako je orientace robotů v uzavřených prostorách, s použitím miniaturní kamery i ve velmi stísněných místech, kde jiné systémy již pracovat nemohou (potrubí, navádění sondy např. při laparoskopii).

V dnešní době jsou k disposici různé kamery v miniaturním provedení. Bohužel mají vždy zakódovaný výstup do nějakého formátu, který je sice vhodný pro přenos, nikoliv však pro další zpracování obrazu. Prvním krokem bude vždy jeho rozkódování z použitého formátu (JPEG, AVI, televizní norma) do prosté bitové mapy. Naštěstí pro tento úkon je dostupná podpora, takže kromě požadavků na příslušné vybavení to nepřináší žádné další potíže.

Pro zpracování obrazů zde požadovaným způsobem však jakákoliv podpora chybí. Určité naděje byly vkládány do knihoven openCV (CV - computer vision, tedy počítačové vidění) fy Intel. V ní se nachází veliké množství algoritmů pro identifikaci objektů, převody obrazů atd., ovšem zde řešenou úlohu zřejmě považují za nedůležitou a ignorují ji.

2 CÍLE DISERTACE

Základní myšlenkou je nasnímání obrázků optickou kamerou při prvním průjezdu vozidla danou trasou, např. s řidičem. Při dalších průjezdech se vozidlo orientuje na trase již samo pomocí dat získaných při prvním průjezdu porovnáním s daty aktuálními. V těch částech trasy, kde není jednoznačná identifikace polohy tímto způsobem, bude poloha určována vyhodnocením aktuální obrazové informace z kamery bez identifikace jejího obsahu jen pro získání rychlosti dopředné a úhlové. S prvotním nasnímáním obrázků se současně nebo dodatečně k obrázků uloží souřadnice. Požaduje se, aby řešený systém dokázal na základě aktuálního obrazu z kamery a obrázků uložených v předchozí jízdě určit souřadnice místa, kde se právě vozidlo s kamerou nalézá. Odvodit příslušné algoritmy, pomocí nichž je možno získat souřadnice vozidla porovnáním obrazu z optické kamery a obrazy se souřadnicemi dříve uloženými.

Cíle disertace jsou:

- Najít vhodnou metodu pro určení posice vozidla opakovaně se pohybujícího po určité trase pomocí optické kamery
- Odvodit příslušné algoritmy, pomocí nichž je možno získat souřadnice vozidla porovnáním obrazu z optické kamery a obrazy se souřadnicemi dříve uloženými.
- Testovat odvozené algoritmy jako důkaz jejich správnosti
- Vytvořené algoritmy realizovat v jazyce C++ ve formě knihovny a zpřístupnit ji.

3 TEORETICKÝ ZÁKLAD ŘEŠENÍ

Tato kapitola je zařazena pro úplnost, jen aby práce měla určitou celistvost. Jde o výtah z populárních publikací o digitálním zpracování obrazu [8], [15]. Termíny zde vysvětlené se v další části této práce používají bez dalšího vysvětlení, které je proto zde.

3.1 Rozložení světelné energie

C(x,y,t, λ) kde jsou (x,y) souřadnice, t je čas a λ vlnová délka. Spektrální citlivost optického snímacího prvku (oka, kamery atd.) je S(λ).
Funkce jasu

$$f(x,y,t) = \int_0^{\infty} C(x,y,t,\lambda) S(\lambda) d\lambda \tag{3.1}$$

Při vícebarevném zobrazení máme více jasových složek:

$$\vec{f} = \left(f_1(x,y,t), f_2(x,y,t), ..., f_n(x,y,t)\right) \tag{3.2}$$

například rudá, zelená, modrá.

U statických obrazů jsou jasové složky dané pouze souřadnicemi x, y. Při počítačovém zpracování pracujeme se souřadnicemi omezenými jen v určitém prostoru R:

$$R = \left\{(x,y), 0 \le x \le x_m, 0 \le y \le y_n\right\} \tag{3.3}$$

3.2 Rozložení Diracovo a konvoluce

Teorie Diracova rozložení má zásadní význam při vzorkování signálu a tedy i videosignálů. Tyto teorie by byly potřebné v případě, že se budeme zajímat o zpracování signálů přímo ve videokameře. Pro nás je k dispozici již signál digitalizovaný ve formátu JPEG. Nicméně tento popis je nutný pro úplnost.

Diracovo rozložení $\delta(x,y)$ je definováno

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(x,y) dx dy = 1 \tag{3.4}$$

a $\delta(x,y) = 0$ pro všechna $x, y \neq 0$

Vlastnost posunutého Diracova rozložení

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x-\lambda,y-\mu)dxdy = f(\lambda,\mu) \tag{3.5}$$

To je základem vzorkování spojitého procesu f(x,y).
Pro vzorkování v celém prostoru obrazu je

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)\delta(a-x,b-y)dadb = f(x,y) \tag{3.6}$$

Důležitou operací je konvoluce g definovaná pro dvojrozměrné funkce f a h

$$g(x,y) = \int_{-\infty}^{\infty} f(a,b)h(x-a,y-b)dadb =$$

$$\int_{-\infty}^{\infty} f(x-a,y-b)h(a,b)dadb =$$

$$= (f * h)(x,y) = h(* f)(x,y) \tag{3.7}$$

Hvězdičkou * označujeme operátor konvoluce.

Fourierova transformace funkce dvou proměnných

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(xu+yv)}dxdy \tag{3.8}$$

Zpětná Fourierova transformace

$$F(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{2\pi i(xu+yv)}dudv \tag{3.9}$$

Jako operátor Fourierovy transformace se používá $\mathcal{F}$

$$\mathcal{F}\{f(x,y)\} = F(u,v) \tag{3.10}$$

3.2.1 Diskrétní kosinová transformace

Diskrétní kosinová transformace (DCT) [5], zejména její dvojrozměrná varianta [3], je součástí obrazového formátu JPEG (Joint Photographic Expert Group) [25] a dalších aplikací. V tomto formátu nám kamera poskytuje nasnímané obrazy. První výsledky byly dosaženy s použitím

elektronických obrázků právě ve formátu JPEG. Význam DCT tkví v možnosti vynechání části koeficientů při inverzní transformaci bez významného snížení kvality přenášené informace. Při trasformaci se pracuje pouze s reálnými čísly, což podstatně zjednodušuje konkrétní použití DCT. Používá se několik verzí této transformace, které bývají označovány jako DCT I až DCT VIII. Všechny obrazy z optické kamery přichází do řešeného systému právě ve formátu JPEG.

- **Definice**

Jednorozměrná DCT je definována:

$$y(k) = c(k)\sum_{n=1}^{N} x(n)\cos\left(\frac{\pi(2n-1)(k-1)}{2N}\right) \tag{3.11}$$

kde

$$c(k) = \begin{cases} \dfrac{1}{\sqrt{N}} & k = 1 \\ \sqrt{\dfrac{2}{N}} & 2 \le k \le N \end{cases} \tag{3.12}$$

K této transformaci existuje transformace inversní, mající velmi podobný tvar:

$$x(n) = \sum_{k=1}^{N} c(k)y(k)\cos\left(\frac{\pi(2n-1)(k-1)}{2N}\right) \tag{3.13}$$

kde opět je

$$c(k) = \begin{cases} \dfrac{1}{\sqrt{N}} & k = 1 \\ \sqrt{\dfrac{2}{N}} & 2 \le k \le N \end{cases} \tag{3.14}$$

Dvojrozměrná DCT pro čtvercové matice je definována:

$$t(i,j) = c(i,j)\sum_{m=1}^{N}\sum_{n=1}^{N} x(m,n)\cos\left(\frac{\pi(2i-1)m}{2N}\right)\cos\left(\frac{\pi(2j-1)n}{2N}\right) \tag{3.15}$$

kde

$$c(i,j) = \begin{cases} \dfrac{1}{N} & i = 0 \\ \dfrac{1}{N} & j = 0 \\ \dfrac{2}{N} & i \ne j \end{cases} \tag{3.16}$$

K této dvojrozměrné transformaci přísluší transformace inversní, která je popsaná takto:

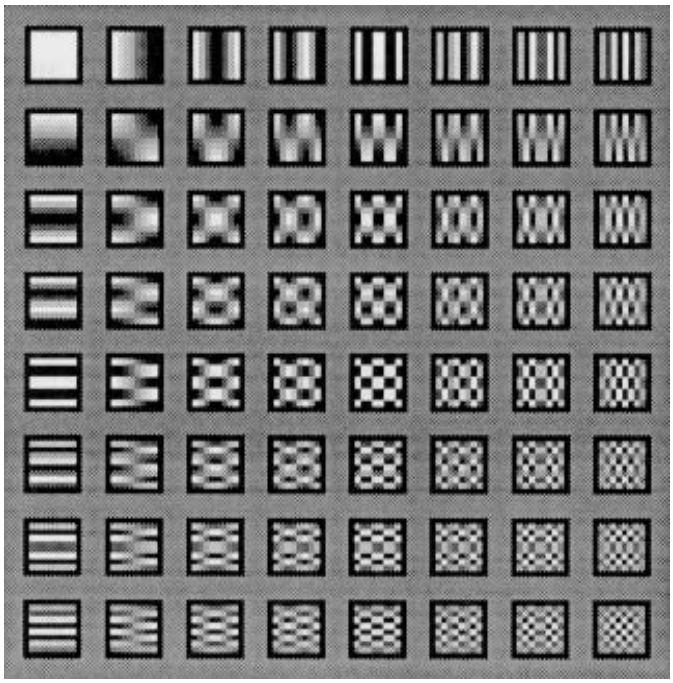
$$x(m,n) = \sum_{i=1}^{N}\sum_{j=1}^{N} c(i,j)t(i,j)\cos\left(\frac{\pi(2m-1)i}{2N}\right)\cos\left(\frac{\pi(2n-1)j}{2N}\right) \tag{3.17}$$

kde je opět:

$$c(i,j)=\begin{cases}\frac{1}{N} & i=0\\ \frac{1}{N} & j=0\\ \frac{2}{N} & i\neq j\end{cases} \tag{3.18}$$

- **Aplikace**

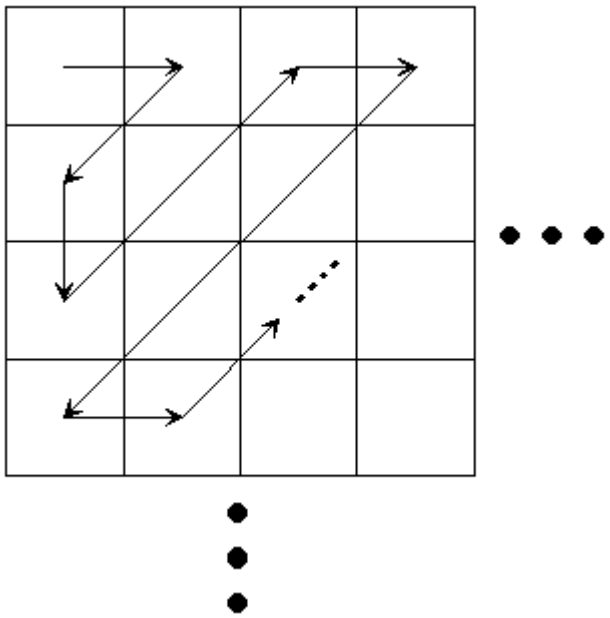
Transformační funkce a k ní příslušná funkce inverzní obsahují vždy stejný součin funkcí kosinus a koeficientů c. Tento součin funkcí bývá nazývám báze (basis), který nabývá jen omezeného počtu hodnot. Proto při konkrétním použití bývá nahrazován tabulkou. Pro dvojrozměrnou transformaci a často používanou matici 8x8 bývá tato tabulka zobrazována v grafické formě dle Obr. 1:



Obr. 1 Grafické znázornění bázových funkcí DCT

Při dvojrozměrné transformaci matice rozměrů např. 8x8 = 64 prvků získáme transformací 8x8=64 koeficientů. Stěžejní význam této transformace tkví v tom, že ne všechny koeficienty mají pro zpětnou transformaci stejný význam. V praxi to znamená, že se méně významné koeficienty ignorují neboli se nezahrnují do výstupního souboru určeného např. pro přenos nebo pro zápis na příslušné médium. Před inverzní transformací se chybějící koeficienty nahradí nulami. Tím sice dochází k určitým ztrátám informačního obsahu, ale pro řadu aplikací vzniklá úspora kapacity média (např. u grafiky) může mít větší význam, než částečné zhoršení kvality výstupu. Z opačného pohledu získáme větší kvalitu (u grafiky např. rozlišovací schopnost) výstupního souboru, který byl podroben uvedenému procesu, než by měl soubor s rozlišovací schopností redukovanou na dosažení stejné délky . Význam této transformace je ten, že je jako součást grafického formátu JPEG používán pro přenos většiny grafiky na internetu a existuje pro něj i hardwarová podpora např. v návrhových systémech hradlových polí atd. Pracovní postup je takový, že obraz určený pro transformaci do formátu JPEG se rozdělí na políčka 8x8 pixelů, každé políčko se samostatně podrobí DCT a ze vzniklých koeficientů se použije jen část. Do výstupního souboru se však koeficienty

transformovaných políček nezapisují v pořadí podle řádků obrazu, ale systémem cik-cak, jak je znázorněno na Obr. 2.

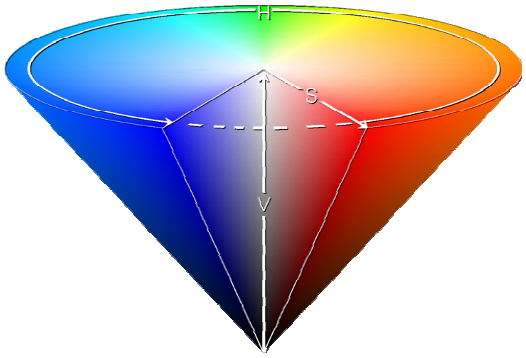


Obr. 2 Pořadí ukládání koeficientů ve formátu JPEG

Z uvedeného je patrné, že snaha určovat polohu přímo podle koeficientů formátu JPEG by nevedla k cíli.

3.2.2 Transformace barev HSV

Převádí klasický obraz RGB do HSV. Celá metoda odstranění stínů [9] se zakládá na teorii, že osvětlená scenerie a tatáž scenerie ve stínu má stejnou barvu, která se liší jen v jasové složce . Snahou je odstranění stínů z obrazu. Formát HSV obsahuje 3 složky:
(Hue, Saturation, Value) nebo-li (Barva, Saturace-sytost, Jas)
Hue H (barva) je jediným nositelem barevné informace, udává barvu a je z rozsahu 0° - 360°
Saturation S (sytost) je z rozsahu 0% – 100% a udává sytost dané barvy, tedy jak hodně je daná barva „barevná“ (0-bezbarvá – černobílá)
Value V (jas), neboli jasová složka, která nabývá rozsahu 0% – 100% (0%-černá 100%-bílá)



Obr. 3 Znázornění významů kanálů HSV

Barevná hodnota je tedy dána úhlem H, který jako jediný nese informaci o samotné barvě pixelu. Z Obr. 3 by mělo být patrné, že s klesající hodnotou V (jas) se barevný kužel zužuje a klesá tak rozdíl mezi jednotlivými barvami. Při nízkém osvětlení jsou tedy vzájemné rozdíly menší a hůře rozlišitelné, což odpovídá realitě – při nedostatečném osvětlení vidí hůře lidské oko i kamera.

Převod z RGB do HSV:

$$H = \begin{cases} nedefinovano & pokud \quad MAX = MIN \\ 60 \cdot \dfrac{G-B}{MAX-MIN} + 0 & pokud \quad MAX = R \; a \; G \geq B \\ 60 \cdot \dfrac{G-B}{MAX-MIN} + 360 & pokud \quad MAX = R \; a \; G < B \\ 60 \cdot \dfrac{B-R}{MAX-MIN} + 120 & pokud \quad MAX = G \\ 60 \cdot \dfrac{R-G}{MAX-MIN} + 240 & pokud \quad MAX = B \end{cases} \tag{3.19}$$

$$S = \begin{cases} 0 & pokud \quad MAX = 0 \\ 1 - \dfrac{MIN}{MAX} & jinak \end{cases} \tag{3.20}$$

$$V = MAX \tag{3.21}$$

kde $H \in \langle 0;360 \rangle \qquad S,V \in \langle 0;1 \rangle$

MAX a MIN zastupují barevné kanály s MAXimální a MINimální hodnotou.

Pro převod z formátu RGB do HSV je v OpenCV funkce ***cvCvtColor(const CvArr* src, CvArr* dst, int code)*** kde *src* je zdrojový obrázek, *dst* je cílový obrázek a code značí způsob transformace (z jakého formátu do kterého). Bohužel tato transformace sice zlepší korelační koeficienty, ale zdaleka nedosahuje takového účinku, jako je zde použité prosté odečtení střední hodnoty.

3.2.3 Použité formáty bitové mapy

Z použité optické kamery přichází obraz ve formátu JPEG [25]. Tento formát používá ztrátovou kompresi založenou na DCT. Před dalším zpracováním se převádí zpět na bitovou mapu a ve formátu bitové mapy se i ukládá. Převod JPEG – bitová mapa se provádí jednoduše: Použitý programovací systém C++ Builder příslušný algoritmus obsahuje a proto se prostě použije [4]. Vstupní formát JPEG po při přiřazení příslušnému okénku transformuje do formátu TColor, což je tříbajtový typ obsahující jednotlivé barevné složky. Z něj jsou extrahovány všechny tři barevné složky a ukládány ve vlastním formátu, což je jen trojice barevných složek, viz kapitola 6.1.

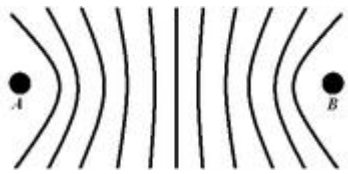
4 ANALÝZA SOUČASNÉHO STAVU PROBLEMATIKY

4.1 Historické systémy navigace

Některé systémy navigace jsou známe již od starověku. Navigace se prováděla podle polohy Slunce (výška Slunce nad horizontem se měřila sextantem) a hvězd na nebi a pomocí magnetických kompasů. Pro určení zeměpisné délky podle astronomických objektů je třeba znát přesný čas výchozího bodu (nultého poledníku). Do doby, než vznikly přesné lodní chronometry to byla výrazná slabina těchto systémů.

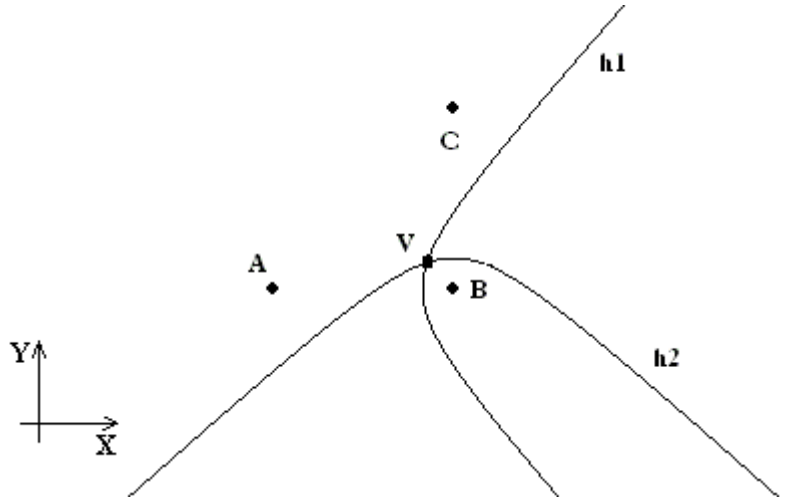
4.1.1 Navigační systém LORAN

Loran (Loomis radio navigation) [24] vznikl během 2. světové války pro potřeby navigace americké armády. Jedná se o síť silných vysílačů. Přijímač, který se nachází někde v jejich dosahu, vyhodnotí rozdíl časů mezi příjmem signálů jednotlivých vysílačů. Geometrickým místem konstantního rozdílu časů příjmu těchto signálů je hyperbola. Proto se tento a podobné navigační systémy někdy označují jako hyperbolické.



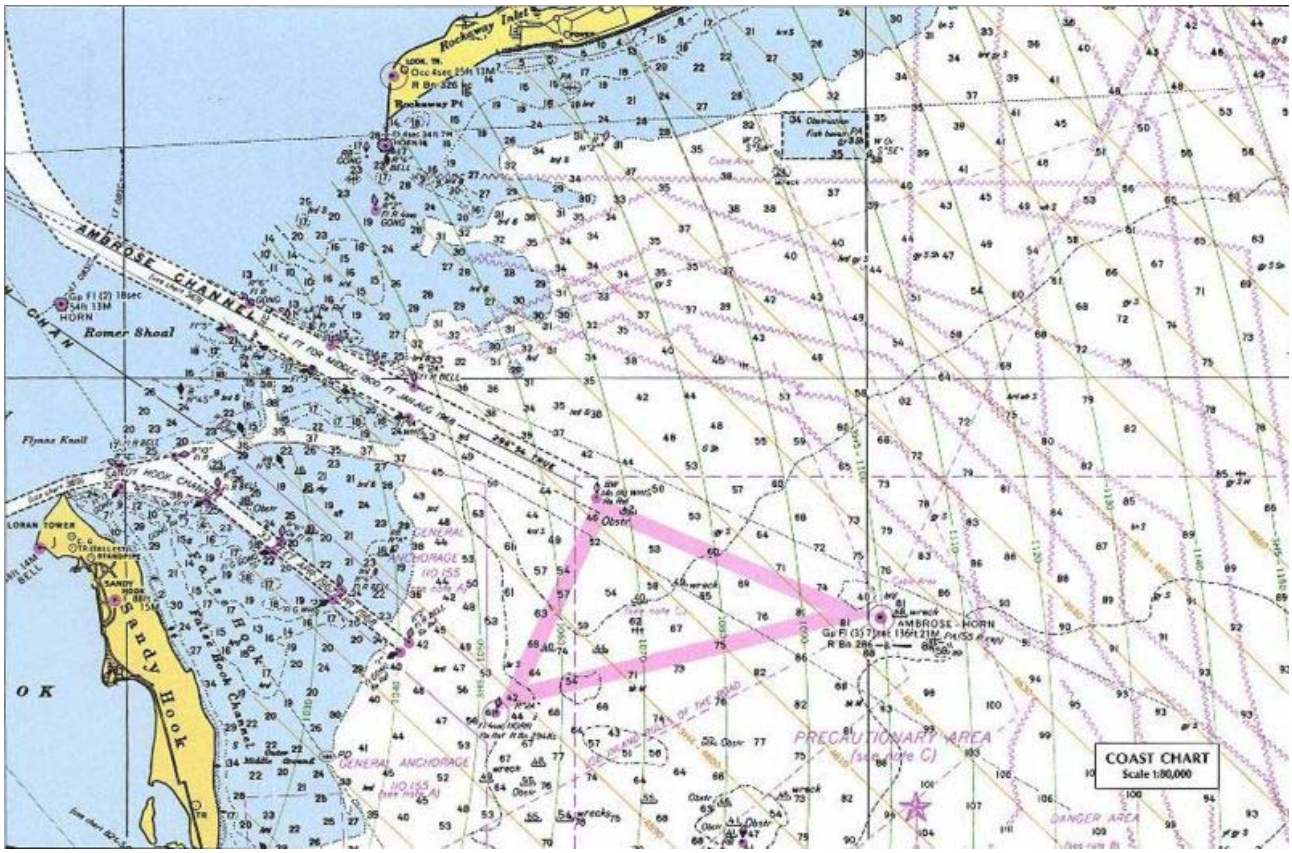
Obr. 4 Křivky konstantního rozdílu časů

Pro každý další signál vznikne nová síť hyperbol. V průsečíku hyperbol na Obr. 5 stejných rozdílů se nachází přijímač. Používají se kmitočty od 130kHz do několika MHz podle verze systému. Pro zajištění dostatečného dosahu se používají výkony vysílačů 100kW až 4MW. Pro určení skutečné polohy se používaly mapy se zakreslenými hyperbolami konstantního časového rozdílu přijímaných kmitočtů. Dosah byl asi 400 mil (644km).



Obr. 5 Stanovení polohy průsečíkem hyperbol u hyperbolického systému

Princip hyperbolického systému: V bodech A, B, C jsou zdroje vysílaných signálů (impulzů), které jsou přijímány přijímači na vozidle V. Zařízení vyhodnotí časový rozdíl v příjmu signálu z bodu A a z bodu B. Z časového rozdílu plyne, že vozidlo V se musí nacházet někde na hyperbole h1. Obdobně rozdíl signálů z bodu B a bodu C určí hyperbolu h2. Průsečík těchto hyperbol určí jednoznačně polohu vozidla V. Pokud jsou průsečíky dva, musí se vytvořit třetí hyperbola z rozdílu signálů v bodě A a C. Otočením směru šíření signálu (vozidlo vysílá impulzy a v bodech A, B a C jsou přijímače) dostaneme princip pasivního radiolokátoru (TAMARA). LORAN ovšem neměl k disposici elektronickou výpočetní techniku, proto používal speciální mapy se zakreslenými hyperbolami příslušnými jednotlivým vysílačům, viz Obr. 6.



Obr. 6 Mapa pro navigaci v systému LORAN

4.1.2 GPS

Systém GPS (global position system) [7] vznikl ze systému LORAN. Jeho vysílače jsou umístěny na satelitech, v současné době je jejich počet 24. Pracují s mnohem menšími výkony, než systém LORAN na kmitočtech 1GHz až 2GHz. Přesnost určení polohy je pro civilní použití několik metrů. Pro širokou veřejnost nabízí řada firem cenově přijatelné přijímací jednotky GPS v cenách od $70 (březen 2008).

Obr. 7 Přijímače GPS pro motoristy a turisty

Tyto jednoduché přijímače GPS běžně používají turisté, cyklisté, piloti všech typů letadel a další. V automobilu je možné k těmto přístrojům připojit laptop s mapami a tím výrazně zvýšit komfort využití tohoto systému. Zde je zmiňován zejména proto, že je využitelný pro vkládání prvotních dat (souřadnic) do řešeného systému.

4.1.3 Galileo

Systém GALILEO je budovaná evropská verze systému amerického GPS. S jeho nasazením do provozu se počítá po r. 2010. V současnosti se projekt potýká s velkými finančními problémy a není jisté, zda bude vůbec dokončen.

4.1.4 Inerciální navigační systém

Metoda zvaná inerciální navigace (inertial navigation) [7] používá tzv. inerciálních senzorů, jako jsou gyroskopy a akcelerometry pro měření rychlosti pohybu vozidla, letadla, satelitu… a následné pozice, přičemž primární měřenou veličinou je zpravidla zrychlení. Celý princip tedy spočívá v efektu, že známe-li startovní pozici objektu a zaznamenáme-li změny zrychlení ve všech osách, které jsou pro měření podstatné, jsme schopni vypočítat současnou rychlost a pozici. Princip metody je jednoduchý, ale praktická realizace je velmi obtížná, protože s integrací užitečného signálu je integrována i chyba. Inerciální navigace využívá obvykle tří gyroskopů (určení náklonů) a tří akcelerometrů (určení zrychlení) pro určení rozdílů polohy a natočení. Protože to není nekonečně přesné a v nekonečně krátké době, hromadí se chyba a proto je systém využíván pro určení polohy v mezích mezi dvěma určeními polohy GPS (obvykle 0,5 s). Dnes jednotky INS najdeme v každém vojenském letadle i v tancích.

4.2 Navigace podle korelace obrazu

4.2.1 Korelace obrazu

Pro funkce jedné proměnné je vzájemná korelační funkce definována [29]

$$c(\tau) = \int_{-\infty}^{+\infty} f(x)g(x-\tau)dx \tag{4.1}$$

Tato operace bývá nazývána konvoluce a jako operátor se používá znak * (hvězdička).

$$c(\tau) = f(x) * g(x - \tau) \tag{4.2}$$

Zpracovávaný obraz má však obvykle 2 rozměry, proto se musíme zajímat o korelaci funkcí dvou proměnných. Pro 2 funkce dvou proměnných je vzájemná korelační funkce definována [29]

$$c(i, j) = \iint_D f(x, y) g(x - i, y - j) dxdy \tag{4.3}$$

Pokud pro vstupní obraz volíme označení f, potom g bude funkce, ktterá určuje zpracování obrazu.

Protože však je náš obraz digitalizovaný, musíme pracovat s posloupnostmi bodů. Potom dostaneme jednotlivé korelační koeficienty [15]:

$$F(m_1, m_2) = \sum_{n_1} \sum_{n_2} F(m_1 - n_1 + 1, m_2 - n_2 + 1) G(n_1, n_2) \tag{4.4}$$

Rozsah indexů n_1 a n_2 nikdy nemůže být nekonečný vzhledem k fyzikální postatě problému. Pro konkrétní účely, jako je dále uvedená derivace obrazu, může mít G výrazně menší rozsah indexů n_1 a n_2. G bývá nazývána jádrem operace. V kapitole 5.1.3 je zdůvodněno, proč pro praktické použití korelace v obraze je nutno nejprve z obrazu odstranit střední hodnoty barevných složek.

4.2.2 Použití derivace v obrazu

Pro derivaci v obraze se často používá metoda Sobelova. Pro detekci svislých hran má matice G tvar:

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{4.5}$$

pro vodorovné hrany:

$$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.6}$$

Ukázka je použitím metody Sobelovy na Obr. 8 (Portál dopravní fakulty v r.2004 před přestavbou). Pro matematické vyjádření se obvykle používá matice, někdy zvaná jádro, viz dříve odvozený vztah (4.4). U barevného obrazu se derivace provádí samostatně pro každou barevnou složku. Předpokládá se, že na hranách objektů svou barvu nemění, což je v běžném provoz téměř vždy splněno. Ve výsledném obrazu očekáváme síť hran a lze očekávat, že posloupnosti korelací poskytnou výrazné

extrémy. V příkladu na Obr. 8 je výchozí obraz určený k derivaci, na Obr. 9 a Obr. 10 vidíme ony derivace.

Obr. 8 Obrázek určený k derivaci

Obr. 9 Derivace obrazu ve svislém směru

Obr. 10 Derivace ve vodorovném směru

Často bývá požadavek na detekci všech hran bez ohledu na jejich směr Obr. 11. Potom se obvykle používá 8 matic, které pokryjí všechny směry po 45^0 . Pro Sobelovu metodu jsou tedy k dispozici kromě uvedených matic (4.5) a (4.6) ještě dalších 6 matic pro zbývající směry, které bývají nazývány podle světových stran [15]:

$$\frac{1}{4}\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \quad \text{Northeast} \tag{4.7}$$

$$\frac{1}{4}\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \text{Northwest} \tag{4.8}$$

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{West} \tag{4.9}$$

$$\frac{1}{4}\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad \text{Southwest} \tag{4.10}$$

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & -2 & -1 \end{bmatrix} \quad \text{South} \tag{4.11}$$

$$\frac{1}{4}\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \quad \text{Southeast} \tag{4.12}$$

Obr. 11 Všesměrová derivace obrazu

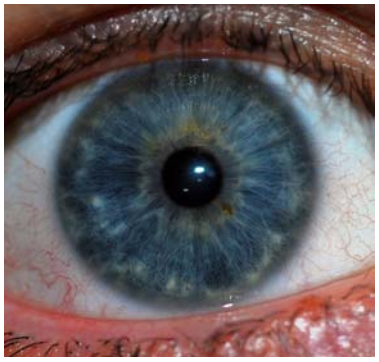
Pro každý pixel obrazu se provede operace samostatně se všemi osmi maticemi a z obdržených koeficientů se vybere ten maximální. Pro obrazy s velkým rozlišením nemusí být řešení derivace pouze pomocí nejbližších pixelů dostatečné, kro tento účel jsou k dispozici rozměrnější matice [15]. V oblíbené knihovně programů OpenCV pro tento účel je funkce cvSobel(…). Hrany, které jsou rovnoběžné s osou pohybu a tudíž jsou v obraze vodorovné, ničím k určení polohy vozidla však nepřispívají. Proto má význam jen případné použití detekce svislých hran.
Kromě metody Sobel se používá pro derivaci hran v obraze řada dalších metod: Prewitt, Roberts, Canny, Kirch, Robinson a další.

4.3 Navigace podle nalezených markantů

Identifikací markantů se zabývá mnoho prací pro svůj velký praktický význam. V obrazech se vyhledávají některé jednoduché geometrické obrazce charakterizující reálné objekty a dále markanty, které indikují přítomnost člověka a identifikují jeho totožnost. Nejznámější je identifikace pomocí papilárních linií, dříve prováděná ručně a identifikace podle oční rohovky.

4.3.1 Identifikace podle duhovky

Pozorováním bylo zjištěno, že každá oční duhovka je [31] unikátní, pravděpodobnost výskytu dvou stejných duhovek je o mnoho řádů nižší, než pravděpodobnost výskytu stejných otisků prstů.



Obr. 12 Duhovka lidského oka

Z hlediska navigace mohou mít algoritmy identifikace duhovek význam jen v antikolizních systémech a jako součást negativního výběru – objekt, v němž je rozpoznáno oko, je pro navigaci nepoužitelný, neboť se přemisťuje. Teoreticky by bylo možné je použít pro identifikaci osob vstupujících do vozidel, taková aplikace však zatím není známa. Ovšem ještě před několika lety se pro přístup do střežených prostor, kde se dnes používá rozpoznávání podle duhovky, používaly pouze klíče a magnetické karty.

4.3.2 Identifikace podle otisku prstu

Otisky prstů jsou unikátní a proto v kriminalistice dosud hojně používané [31]. V knihovně OpenCV [13] je k dispozici řada funkcí pro nalézání rohů, hran, obdélníků atd. v obrazu. Výsledkem hledání jsou souřadnice markantu a jeho typ. Díky daktyloskopii bylo zjištěno a poté usvědčeno na

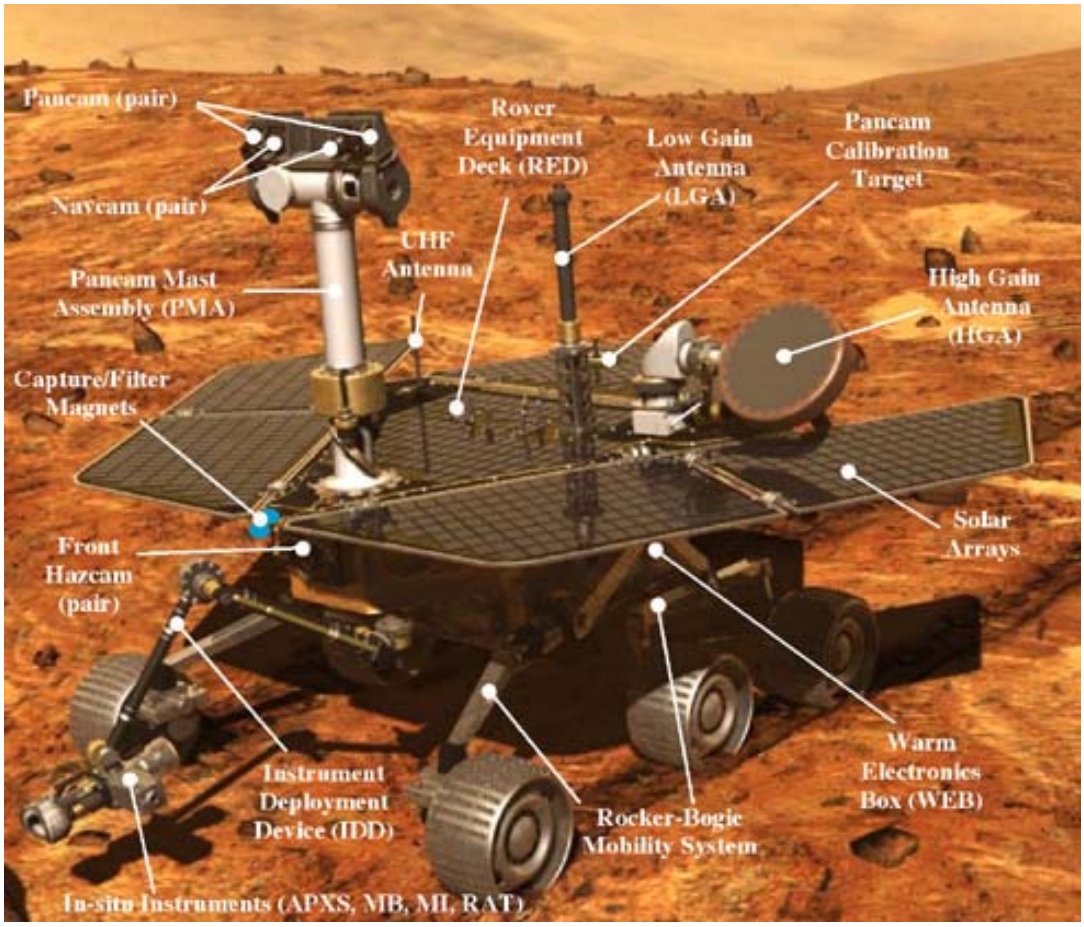
desítky tisíc zločinců a ztotožněno mnoho neznámých mrtvol. Pro nás jsou zajímavé tím, že byly první, které začaly být zpracovávány strojně. Byla vytipována řada markantů [31], které se v každém otisku vyhledávají a jejich výskyt se katalogizuje. Proto existují algoritmy pro ukládání, vyhledávání a porovnávání právě podle těchto markantů.

Obr. 13 Markanty v otisku prstu

Lze se domnívat, že některé algoritmy zpracování obrazů z této problematiky by byly použitelné i v této práci. Bohužel nejsou veřejně dostupné. Lze však očekávat, že znalost některých algoritmů používaných v této problematice mohla být přínosem i pro tuto práci.

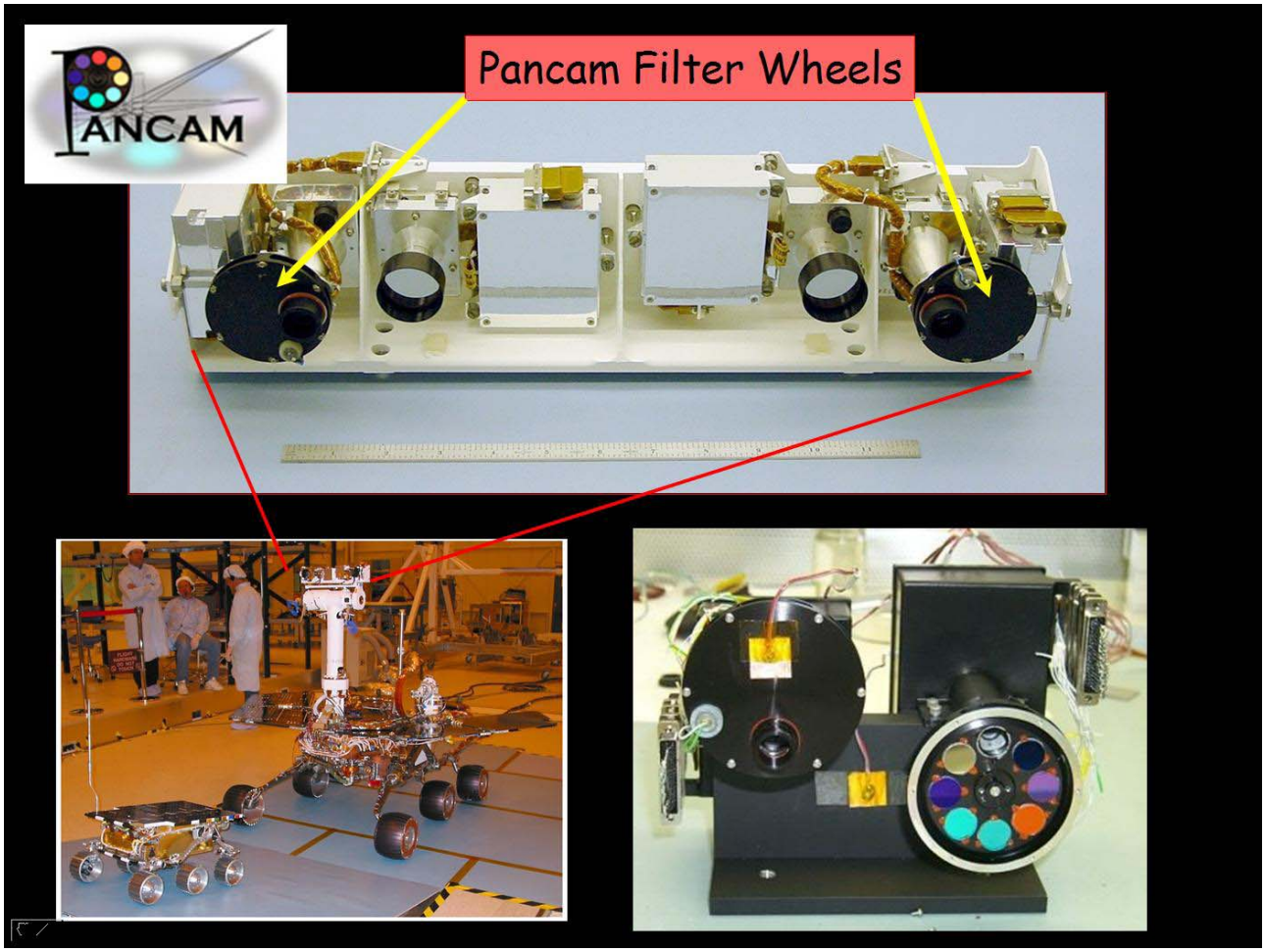
4.4 Robotická vozidla Spirit a Oportunity

Jsou to vozidla (rovery) použitá pro výzkum povrchu Marsu [32]. Mohou se částečně pohybovat autonomně pomocí obrazů z kamer. Vozidlo Spirit přistálo 4. ledna 2004 v kráteru Gusev a Opportunity 25. ledna 2004. Na rozdíl od předchozího vozítka Sojourneru, který za celou dobu trvání jeho průzkumu okolí Pathfinderu najezdil okolo 100 metrů, byly nové rovery schopny urazit tuto vzdálenost během jednoho solu - marťanského dne. Každý z nich nesl vědecké přístroje pro hledání důkazů o přítomnosti tekoucí vody na povrchu Marsu v minulých dobách. Oba rovery byly naprosto identické, ale přistály v různých částech Marsu. Každé vozidlo má 9 kamer [Obr. 14].



Obr. 14 Vozidlo Oportunity

Všechny kamery používají CCD prvky 1024x1024 pixelů. Hlavní kamery jsou určeny pro získání vlastních vědeckých informací a mají pro získání stereoefektu rozteč 280mm. Získaný obraz bývá nazýván hyperstereoskopický, protože rozteč kamer je větší, než rozteč očí u lidí (54 až 72mm). Hlavní dvojice kamer má světelnost objektivu f/20 a jsou pevně zaostřeny na 3000mm, takže obraz je ostrý od 1500mm do nekonečna. Hlavní kamery jsou vybaveny osmi barevnými filtry Obr. 15.

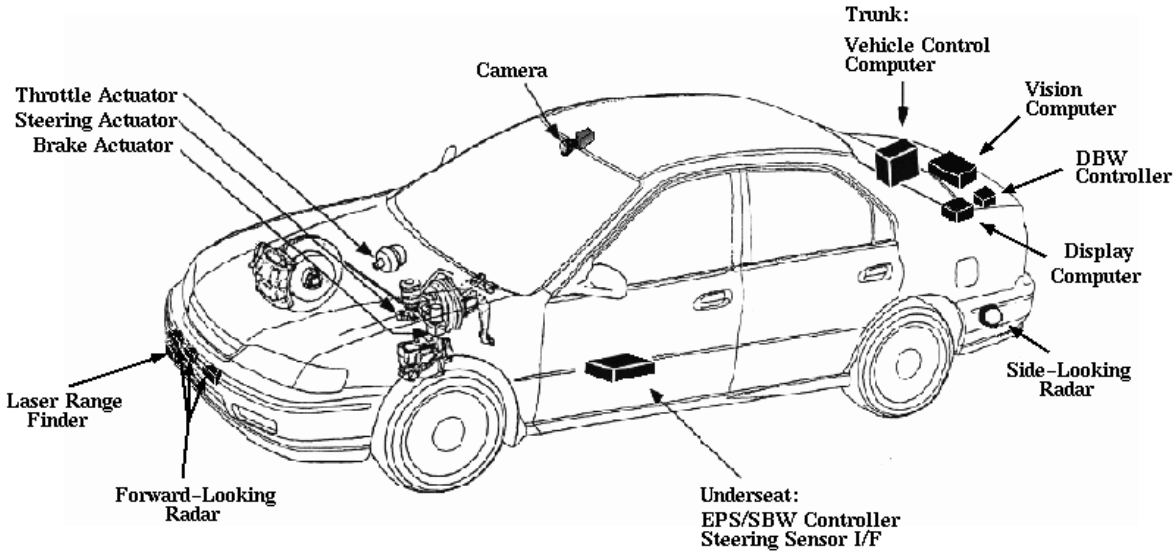


Obr. 15 Dvojice kamer na otočném rameni

Dvojice kamer určených pro navigaci je monochromatická a má rozteč 200mm, zaostřeny jsou na 1000mm, takže poskytují ostrý obraz od 500mm do nekonečna.
Kolizní kamery jsou dva páry vpředu a dva páry vzadu. Jejich objektivy jsou typu "rybí oko" se zorným úhlem přes úhlopříčku 180 stupňů a stran 124 stupňů. Jsou zaostřeny na vzdálenost 400mm a poskytují ostrý obraz od 200mm.
Mimo těchto kamer je na vozidle řada dalších, např. širokoúhlá kamera určující směr ke Slunci, aby bylo možno správně nasměrovat antény pro rádiový přenos.
Navigační software a schopnosti vyhodnotit nebezpečí kolize palubního počítače umožňuje postupovat samostatně v zadaném směru podle denního souboru povelů. Rychlost přesunu je až 50 mm/s (180 m/h) na rovném a tvrdém povrchu, ale při činnosti automatického navigačního systému je průměrná rychlost asi pětinová.
Pod úrovní paluby jsou namontovány dvě stereoskopické kamery identifikace kolizí - jedna vpředu a jedna vzadu. Kromě navigační funkce přední kamera snímá rovněž práci robotizované ruky. Další dva páry stereoskopických kamer se nacházejí na stěžni vystupujícím nad palubu. Panoramatická kamera je součástí souboru vědeckých přístrojů, širokoúhlá kamera s nízkým rozlišením je další součástí navigačního systému. Na stěžni se nachází ještě další z vědeckých přístrojů - miniaturní spektrometr měřící tepelné vyzařování.
Počítač používá 32 bitový procesor Rad 6000, který je odolnější proti radiaci (rychlost 20 Mips, 128 Mbyte RAM, 256 Mbyte flash memory) a malou paměť uchovávající data při výpadku proudu.. Počítač zpracovává data z vědeckých přístrojů a autonomně řídí pohyb roveru s využitím prvků umělé inteligence. Tento počítač rovněž řídil sondu při přeletu ze Země na Mars.
Na spodní straně roveru jsou umístěny dvě dvojice kamer HazCam (Hazard Avoidance Camera) o zorném úhlu 120°, které slouží k detekci překážek do vzdálenosti 4 metrů. Jedna dvojice těchto kamer se nachází vpředu, druhá vzadu. Přední dvojice navíc kromě navigace ještě slouží ke snímání práce robotické ruky. Další pár navigačních kamer se označuje jako NavCam (Navigation Camera) a nachází se na stěžni PMA (Pancam Mast Assembly), který vystupuje nad palubu. Kamery NavCam mají zorný úhel 45°. Kromě toho lze pro navigaci použít i vědeckou panoramatickou kameru rovněž umístěnou na stěžni PMA.
Spirit i Oportunity jsou dosud (červenec 2009) v provozu. Původní předpokládaná životnost byla 90 solů (marťanských dní = 24hod. 39min.), avšak nyní má za sebou na Marsu dosud již 1551. sol a Spirit dosud urazil přes 7,5 km, OPPORTUNITY přes 11,7 km po povrchu Marsu.

4.5 Pokusné vozidlo centra CITR univerzity Ohio

Robotické řízení vozidel centra CITR (Center for Intelligent Transportation Research) na universitě OHIO [33] používá optickou kameru jako jedno z čidel, detekující kolizní situaci. Vozidlo je dále vybaveno mikrovlnnými radary vpředu a po stranách vozidla, laserovými čidly pro měření vzdálenosti překážek před vozidlem a dalšími čidly pro řízení vozidla.



Obr. 16 Pokusné vozidlo centra CITR

Optická kamera snímá monochromatické obrázky s rozlišením 512 x 512 pixelů s 8 bitovým rozlišením jasu. Obrazová frekvence je 17 Hz. Hlavním úkolem kamery je navigace pomocí vodících čar na dálnici, dosažená přesnost je +-5cm s dosahem 6m. V uvedené aplikaci se tedy kamera využívá ke sledování vodících čar na vozovce, plných i přerušovaných. Je nepochybné, že informace z jediného čidla nestačí k bezpečnému provozu vozidla. Situace se zobrazovala na LCD panelu.

4.6 Bioroboti pro práci v zemědělství

V Japonsku jsou vyvíjeni "bioroboti" [34], kteří mají v zemědělství nahradit ubývající počet manuálních pracovníků. Na podvozku je umístěno hydraulické rameno s čtyřčlenným manipulátorem. CCD kamera je umístěna nad manipulátorem a poskytuje informaci jak o okolním prostředí, tak i o práci manipulátoru. Používá víceprocesorový systém transputerů, kterému se před časem připisovala velká budoucnost, ale rychlý vývoj procesorové techniky nedal příležitost k většímu rozvinutí této technologie.

Obr. 17 Japonský robot pro práci v zemědělství.

CCD kamera poskytuje barevný obraz s rozlišením 512 x 480 pixelů. Je umístěna 1,8m nad terénem a její zorné pole zahrnuje oblast 2,4 x 2,3m. Obraz zpracovává 8 transputerů, o jejichž rychlosti pramen nic neuvádí; zřejmě proto, že jejich náhrada jedním běžným procesorem s dnes běžnou rychlostí řádu GHz by byla příliš aktuální. Jako jeden z důvodů tohoto řešení je použití neuronové sítě modelované z těchto prvků.

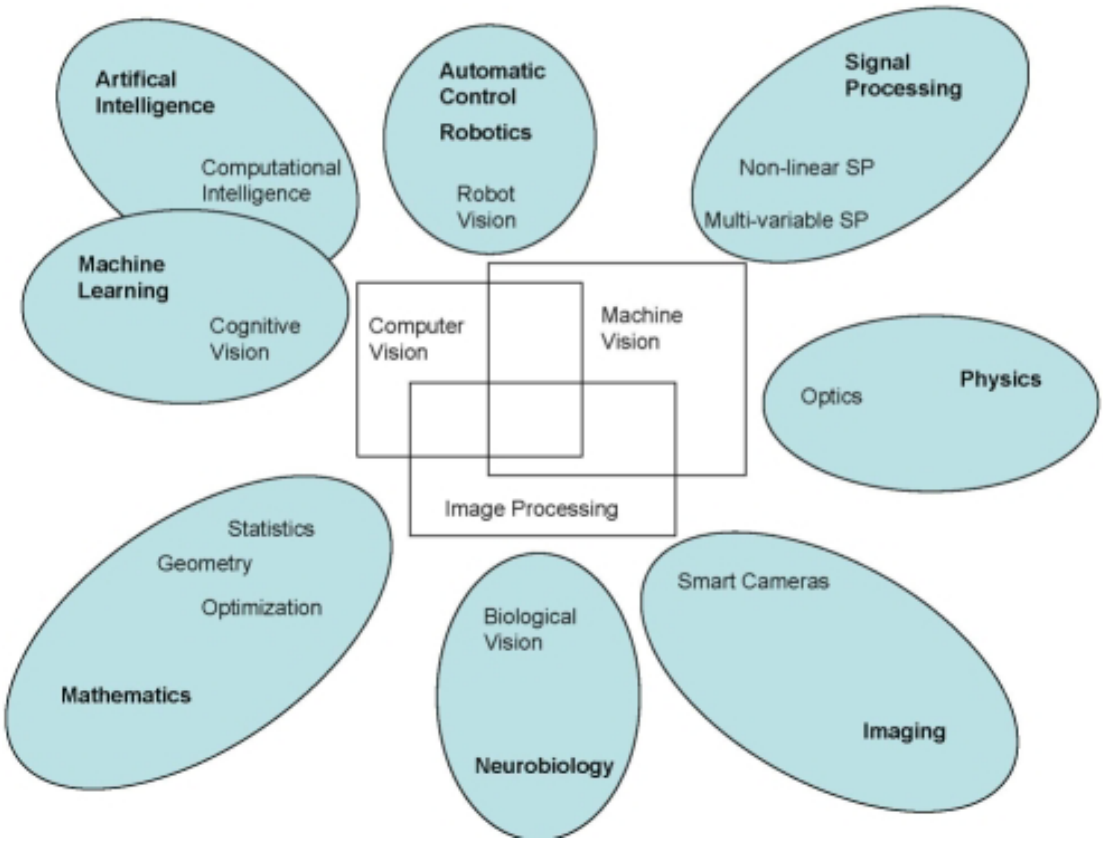
4.7 Programové prostředky

Společnost Intel pracuje na vývoji technologií, které by počítačům umožnily přirozenou interakci s okolním světem, podobnou lidské. OpenCV (Open Source Computer Vision Library) [12], [13], [14] je volně šiřitelná knihovna pro analýzu obrazu, původně vyvinutá firmou Intel. Umožňuje zpracovávat obraz z kamery nebo z videosouboru v reálném čase. Zdrojový kód knihoven je k dispozici v jazyce C++, knihovna je multiplatformní, běží jak pod Windows, tak i pod Linuxem a MacOS. Velmi zjednodušuje programování aplikací pracujících s obrazem v reálném čase tím, že sjednocuje přístup k obrazovým zařízením a nabízí velké množství funkcí pro analýzu a zpracování obrazu. Hnacím motorem je využití výkonných procesorů, které Intel vyvíjí, vyrábí a prodává. Povzbuzení vývoje aplikací vyžadujících použití výkonných procesorů je jednou z cest. Knihovna OpenCV je používána při vývoji aplikací počínaje hračkami a konče průmyslovou výrobou [10], [11]. Tato softwarová knihovna obsahuje zdrojový kód všech funkcí v jazyku C a bezplatnou licenci k další distribuci. Přehled oblastí dosud zpracovaných je na http://www.intel.com/technology/computing/opencv/overview.htm Obsah knihovny OpenCV je zhruba rozdělen do 4 kategorií:

- Základní zpracování obrazu (filtrování, detekce hran, interpolace, konverze barev, histogramy atd.)
- Analytické funkce (strukturální analýza, analýza pohybu, detekce objektů atd.)
- Grafické prvky (kreslení čar, kružnic a dalších grafických prvků)
- Prvky uživatelského rozhraní (okno, zobrazení obrázku, spolupráce s myší a klávesnicí atd.)

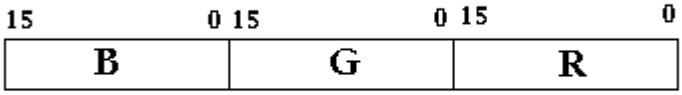
Knihovna je volně k disposici na internetu na adrese http://www.intel.com/technology/computing/opencv/index.htm.

Hodně práce bylo provedeno na rozpoznávání objektů, různé druhy zpracování obrazů atd. Navigace podle optické kamery je zřejmě pokládána za nedůležitou a tudíž je opomíjena.



Obr. 18 Struktura knihovny openCV

Tato knihovna, těžící z výhodných vlastností jazyka C/C++, je mezi návrháři aplikací zpracovávající obrazová data, velice oblíbená [10], [11]. Z výše uvedených důvodů však nebyly algoritmy z této knihovny použity.



Obr. 19 Rozložení barevných složek ve struktuře TColor

5 ŘEŠENÍ ÚLOHY

5.1 Možnosti využití korelace obrazu získaného kamerou umístěnou na vozidle.

5.1.1 Co kamera zobrazuje

Zobrazené objekty můžeme dělit podle různých kritérií: malé/velké, světlé/tmavé, kulaté/hranaté... pro nás je důležité dělení podle použitelnosti:

a) Objekty použitelné k identifikaci polohy vozidla. jsou to zejména objekty blízké, které svou polohu mezi jednotlivými jízdami nemění. jsou to např. budovy, sloupy, stromy, terénní zvláštnosti. Na Obr. 20 je příklad dobré sekvence snímků pro určení polohy vozidla. Obsahují jak blízké rozměrnější objekty, tak i hrany kolmé na směr pohybu (tedy svislé hrany). Dále objekty na snímcích jsou pevné, je předpoklad, že se na stejných místech budou nacházet i při opakovaných průjezdech vozidla s kamerou.

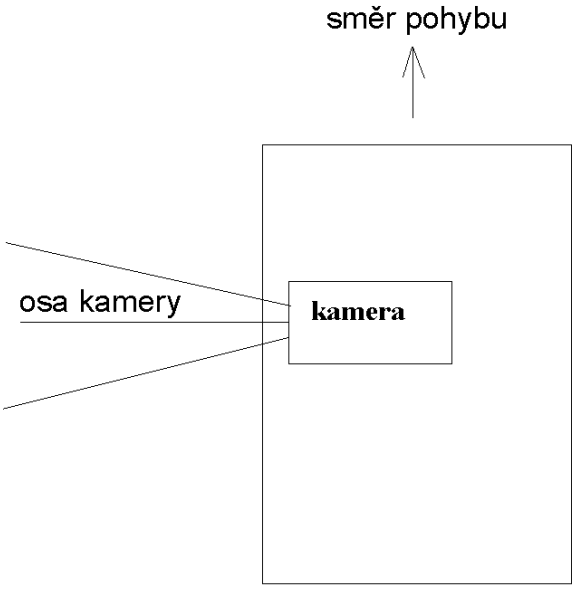
b) objekty nepoužitelné pro identifikaci polohy vozidla. jsou to zejména objekty, které se pohybují, např. jiná vozidla, mraky, stíny objektů závislé na poloze Slunce. Dále objekty příliš vzdálené, jejichž obraz se při změně polohy kamery o jeden krok změní o méně, než jeden pixel. Na Obr. 21 je příklad sekvence snímků, podle kterých lze jen obtížně nebo vůbec určit polohu vozidla. Blízké okolí kamery neobsahuje markanty, zbytky snímků obsahují zobrazení vozidla, které při příštím průjezdu vozidla s kamerou může být přemístěno či vůbec chybět.

Obr. 20 Sled snímků vhodných pro orientaci vozidla

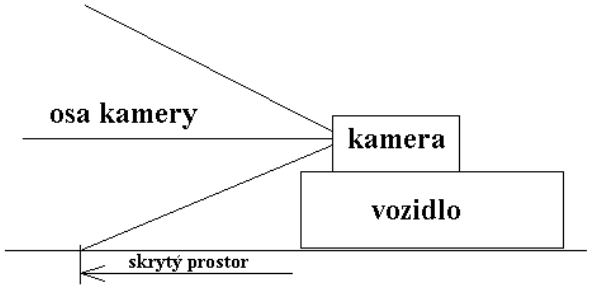
Obr. 21 Sled snímků nevhodných pro určení polohy vozidla

5.1.2 Kamera se zorným polem do boku vozidla

Přepokládá se, že kamera je umístěna na vozidle v takové výšce nad terénem, že je osa objektivu vodorovná a kolmá na směr pohybu viz Obr. 22 a Obr. 23.



Obr. 22 Vozidlo s boční kamerou, pohled shora



Obr. 23 Vozidlo s boční kamerou, pohled zezadu

Při takto nastavené kameře bude zobrazovaný horizont procházet středem snímku. V dolní polovině snímku bude okolní terén, v horní polovině snímku se zobrazí objekty, jejichž výška přesahuje osu kamery a tudíž se zobrazí nad horizontem. Při pohybu vozidla s kamerou se bude vzdálený horizont (pokud je viditelný) jevit jako stojící, objekty v blízkosti kamery budou na snímcích postupovat v závislosti na rychlosti vozidla a vzdálenosti objektu od vozidla, jak vidíme na Obr. 20 a Obr. 21.

Její zorné pole nezahrnuje vozovku. Celý nasnímaný obraz lze použít k dalšímu zpracování. Objekty, které se v obraze vyskytují, můžeme rozdělit do 3 základních skupin:

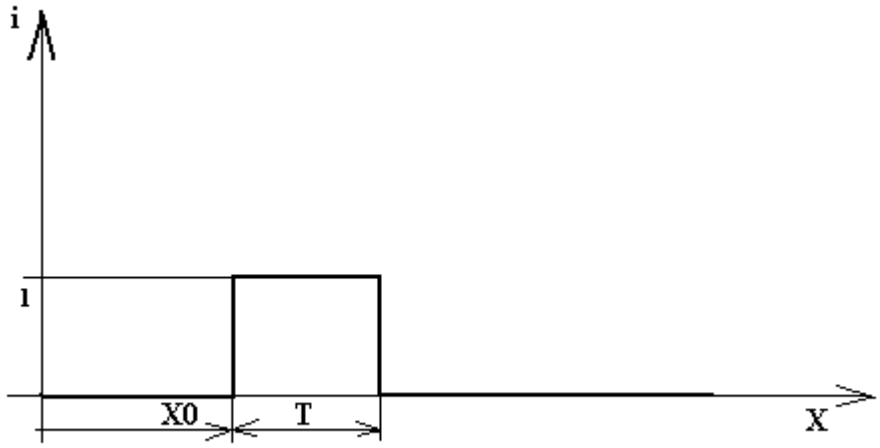
a) objekty blízké, použitelné k přesnému určení polohy vozidla. Jsou to např. sloupy, dopravní značky, stromořadí.

b) Objekty ve střední vzdálenosti. Lze je použít jak k určení posice vozidla, tak k přibližnému zjištění polohy. Jsou to např. domy, ploty, jiné stavby.

c) Objekty vzdálené. Jejich zobrazení prakticky nepřispívá k určení polohy, protože jsou příliš vzdálené a jejich obraz se při významné změně polohy vozidla změní méně, než je rozlišovací schopnost použitého zobrazení.

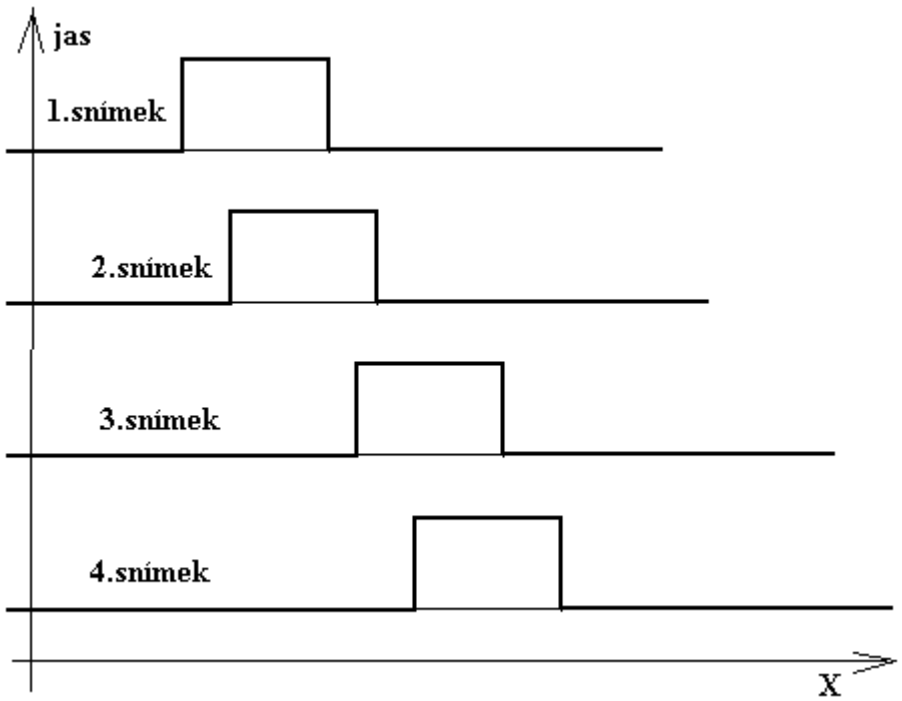
Pokud vozidlo míjí nějaký objekt, potom se jeho pozice v obraze při snímání boční kamerou posouvá. Porovnání s posloupností snímků dříve uložených provedeme pomocí korelačních koeficientů. Jako příklad budeme uvažovat jeden řádek obrazu, jehož pozadí má nulovou hodnotu úrovně jasu a objekt, který se zobrazí v řádku zvýšením jasu v délce T. Při porovnání se snímky z předchozího průjezdu budeme tedy porovnávat obrazy s různě posunutým zobrazením objektu délky T. Pokud odhlédneme od způsobu získání obrazů – současný a obrazy z minulého průjezdu, pak se vlastně jedná o vytváření koeficientů korelační funkce (4.4). Následující příklad to osvětlí:

V zorném poli kamery předpokládáme pozadí, které má pro jednoduchost nulovou hodnotu jasových složek. Dále předpokládáme v zorném poli předmět, který se na obrázku zobrazí jako útvar šířky T. Pro další zjednodušení popíšeme jen zpracování v jednom řádku. Průběh jasu na jednom snímku z prvního průjezdu vozidla s kamerou je zobrazen na Obr. 24.



Obr. 24 Průběh jasu v jednom řádku

Při opakované jízdě získáme podobné obrázky, vzhledem k okamžité poloze vozidla se však onen předmět zobrazí na různých místech snímku Obr. 25:



Obr. 25 Průběhy jasu na nových snímcích

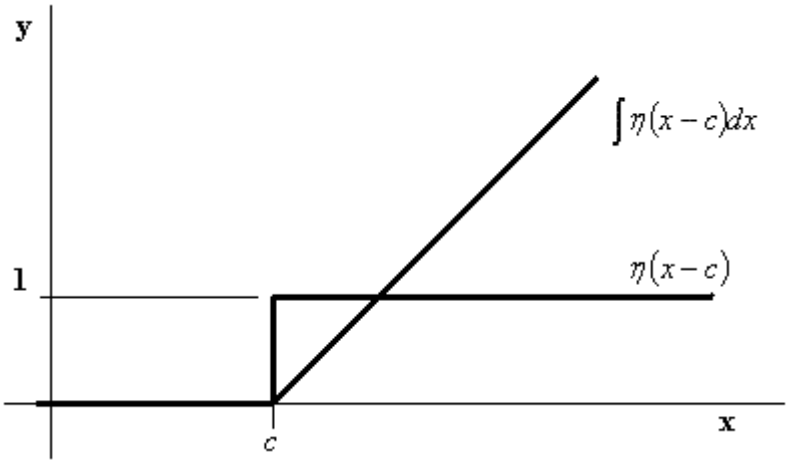
Zajímá nás vzájemná korelace (korelační koeficienty) funkce jasu na prvotním snímku na Obr. 24 a snímků získaných při opakovaném průjezdu vozidla s kamerou na Obr. 25.

Nejprve tedy popíšeme průběh na Obr. 24. Pro popis použijeme funkci jednotkového skoku, někdy též nazývanou Heavisidovou funkcí[29].

$$\eta(x)=\begin{cases}0, & x<0\\ 1, & x\geq 0\end{cases} \tag{5.1}$$

A dále budeme potřebovat integrál posunuté Heavisidovy funkce Obr. 26:

$$\int \eta(x-c)dx=\begin{cases}=0 & x<c\\ =x-c & x\geq c\end{cases} \tag{5.2}$$



Obr. 26 Posunutá Heavisidova funkce a její integrál

Pomocí této funkce můžeme popsat funkci na Obr. 24:

$$f(x,x_0)=\eta(x-x_0)-\eta(x-x_0-T) \tag{5.3}$$

Funkce jasu na obrázcích získaných při opakovaném průjezdu budou jen posunuté do x_1:

$$f(x,x_1)=\eta(x-x_1)-\eta(x-x_1-T) \tag{5.4}$$

Vzájemná korelační funkce je podle vztahu dříve uvedeného (4.40)

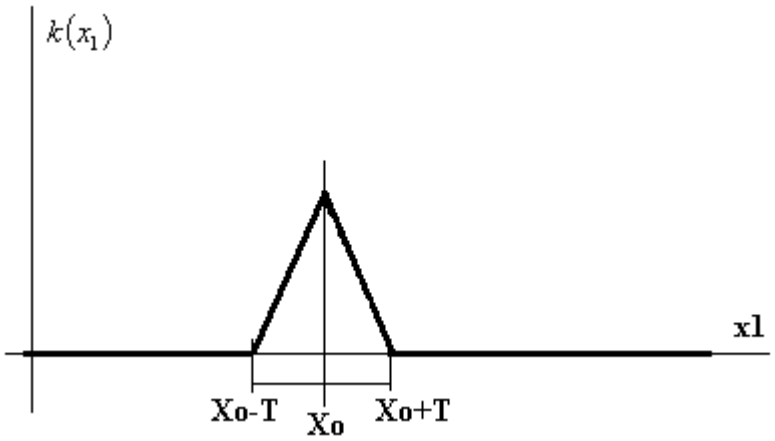
$$k(x_0,x_1)=\int_0^\infty f(x)g(x-\tau)dx=\int_0^\infty [\eta(x-x_0)-\eta(x-x_0-T)][\eta(x-x_1)-\eta(x-x_1-T)]dx \tag{5.5}$$

Protože funkce (5.3) má nenulovou hodnotu jen v intervalu x_0 až x_0+T a ve zbývající části je jednotková, můžeme zkrátit integrační meze v rovnici (5.5):

$$k(x_0,x_1)=\int_{x_0}^{x_0+T}[\eta(x-x_1)-\eta(x-x_1-T)]dx \tag{5.6}$$

Zbývající funkce je jednotková v intervalu $<x_1,x_1+T>$, takže můžeme najít řešení pouhým rozdělením intervalu s použitím (5.2) :

$$k(x_0,x_1)=\begin{cases}=0 & x_1<x_0-T\\ =x-x_0+T & x_1\in<x_0-T,x_0>\\ =x_0+T-x & x_1\in<x_0,x_0+T>\\ =0 & x_1>x_0+T\end{cases} \tag{5.7}$$

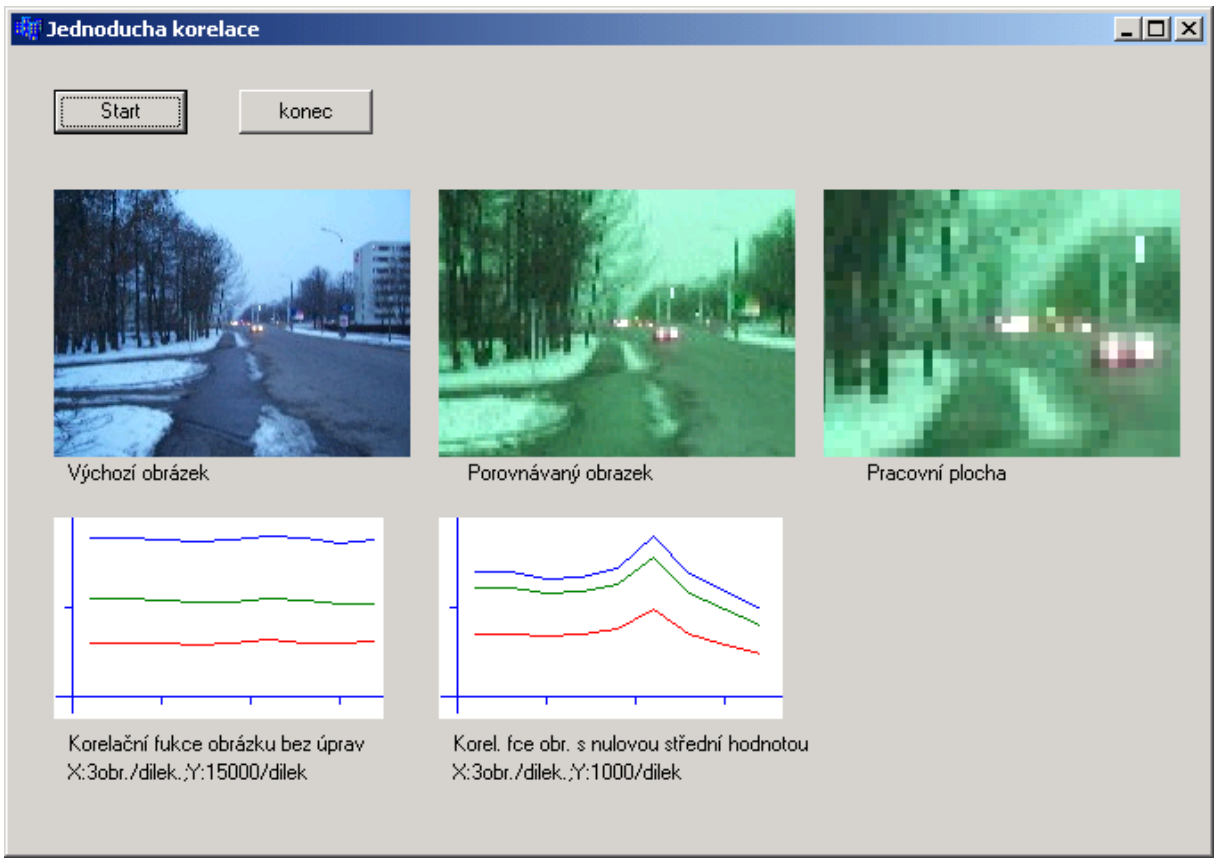


Obr. 27 Průběh korelačního koeficientu v závislosti na posunutí obrázků

Podle očekávání má koeficient $k(x_1)$ maximum pro $x_1 = x_0$, tedy v případě, že se obrázky přesně překrývají. To je základ určení polohy vozidla pomocí obrazů z kamery, prostě musíme hledat maxima této funkce. V polovině převýšení $k(x_1)$ vidíme, že šířka je právě T. To jsme sice intuitivně mohli předpokládat, v (5.7) máme k disposici matematické vyjádření.

5.1.3 Kamera se zorným polem před vozidlo

Předpokládáme kameru umístěnou v ose vozidla se zorným polem ve směru pohybu. Pro první přiblížení problematiky byl pohyb vozidla nahrazen programovým zoomem obrázku. Získané obrázky sice neodpovídají přesně obrázkům, které bychom dostali z kamery pohybujícího se vozidla, ale na druhou stranu nemusíme eliminovat další vlivy, kterými se budeme zabývat až později.
U korelační funkce je nezávislým parametrem posunutí funkce, v našem případě obrazu. To je v tomto jednoduchém případě nahrazeno parametrem zvětšení obrázku, které simuluje pohyb kamery.
Pokusné obrázky mají velikost 200x150 pixelů, každá ze tří barevných složek má úroveň od 0 do 255. Použití obrázku s vyšším rozlišením dává prakticky stejné výsledky, doba zpracování je však výrazně delší. Původní originální obrázky měly velikost 2000x2500 pixelů. Korelace se vždy prováděla s obrázkem digitálně zvětšeným na dvojnásobek tak, aby bylo možno maximum korelační funkce očekávat uprostřed grafu zobrazující průběh korelačního koeficientu v závislosti na digitálním zvětšení, které zde supluje za reálný posuv kamery.
Hned při prvních pokusech zjistíme, že korelační funkce funkcí, které mají výrazně nenulovou střední hodnotu, je prakticky nepoužitelná. V místě, kde očekáváme maximum, tj. v místě, kde se porovnávají 2 stejné obrázky vznikne jen těžko pozorovatelné navýšení. Z toho plyne první důležitý výsledek – pro další zpracování musíme používat pouze funkce (obrázky) s nulovou střední hodnotou. Pracovní postup je takový, že se změří střední hodnota příslušné barvy celého obrázku a ta se pak od každého bodu odečte. Výsledné funkce mají potom téměř polovinu hodnot záporných, což sice neodpovídá fyzikální realitě, ale výsledný průběh korelační funkce již dává přesvědčivější výsledky. V následujícím se bude vždy jednat o obrázky s nulovou hodnotou střední úrovně barevných složek. Střední hodnoty jsou jedněmi z atributů příslušných tříd, takže je možno s jejich pomocí zrekonstruovat původní obraz. To je zapotřebí např. při rekonstrukci obrazu při tvorbě posunutého vzoru pro detekci stranové odchylky. Odstranění středních hodnot je aplikováno jako metoda všech tříd a šablon, kde mohou vznikat, tj. při načtení obrázku, jeho rozdělení apod. Sekundárním efektem je zmenšení vlivu osvětlení na další zpracování obrazových informací.



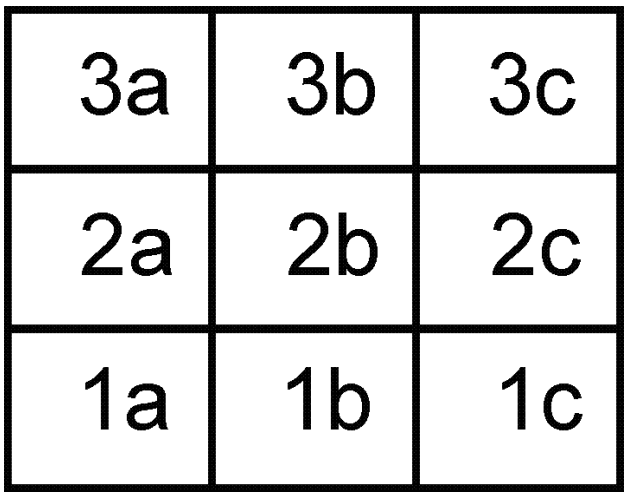
Obr. 28 Korelace neupraveného a upraveného obrázku

Obr. 29 Jednotlivé fáze při digitální úpravě obrázku před korelací

Přesto však vidíme, že vrchol korelační funkce je sice dobře parný, ale jeho okolí je výrazně nenulové. Příčinu spatřujeme přímo v korelovaných obrázcích – i při posunutí si jsou podobné. Tuto podobnost najdeme i při korelaci obrázků získaných z různých komunikací, technologicky i prostorově vzdálených – železniční trať v lese dá nenulovou korelaci se silnicí ve městě.
Na obrázcích vidíme, že např. v horní třetině je téměř vždy vidět jen obloha, ve středu dolní části pak část komunikace. Tyto části obrazu sice ničím nepřispívají k možnosti určení polohy vozidla, ale výrazně zvyšují hodnotu korelační funkce. Vzniká tedy otázka, zda není lepší se těchto neužitečných částí zbavit.

5.2 Rozdělení obrazu

Obrázek, reprezentovaný jako bitová mapa RGB, má nenulové střední hodnoty jednotlivých barevných složek. Ty je nutno odstranit. Velikosti středních hodnot jednotlivých obrázků jsou sice ukládány jako atributy příslušných tříd obrazů, ale jen pro případnou rekonstrukci obrazu pro nějakou demonstraci. Střední hodnota se dále na zpracování obrazu nijak nepodílí. Dalším krokem je rozdělení obrazu na 9 dílčích částí Obr. 30. Vychází se z toho, že vestavěná kamera na vozidle poskytuje obrazy, které mají některé shodné charakteristiky s různým stupněm použitelnosti ke stanovení polohy vozidla.



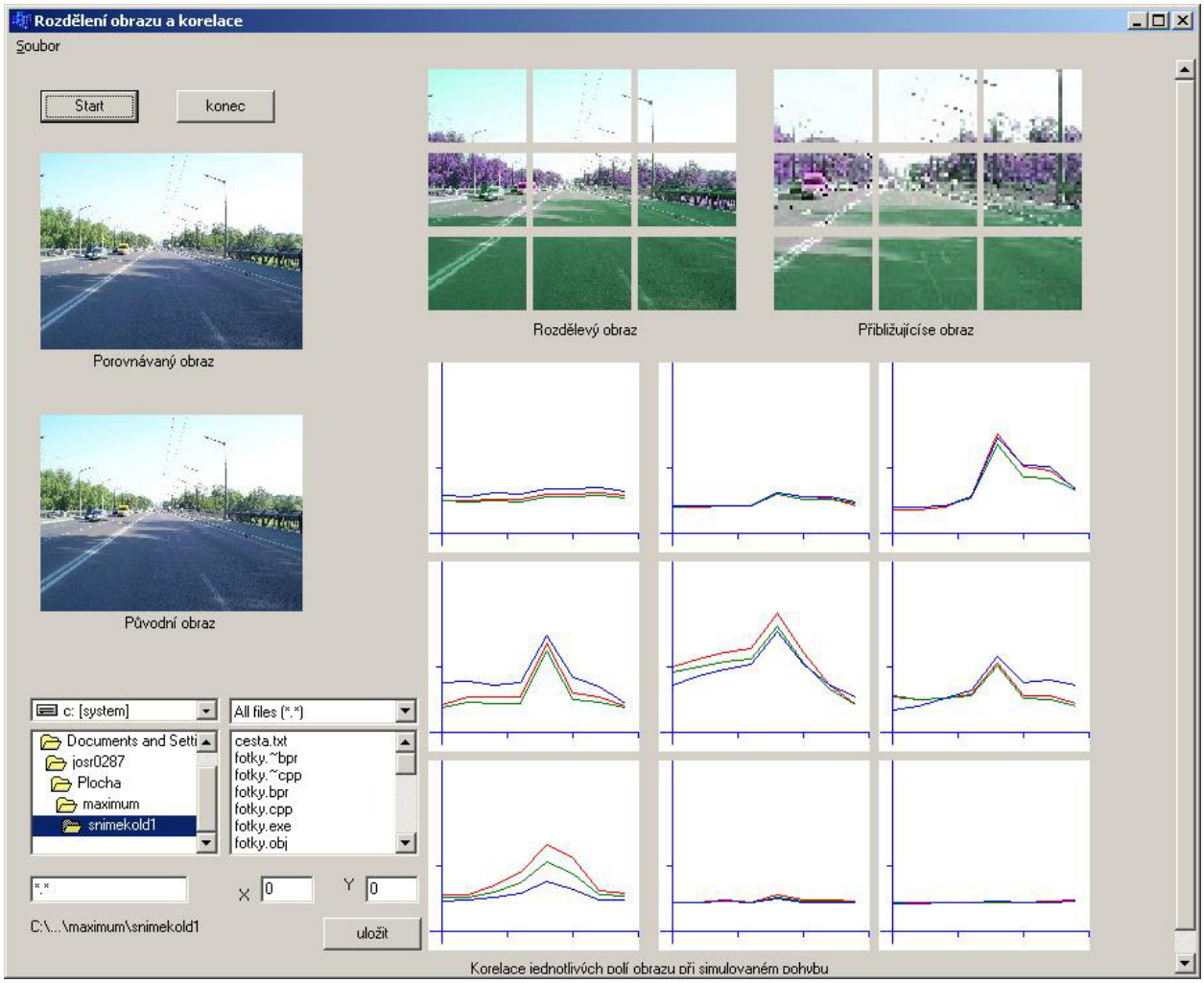
Obr. 30 Rozdělení obrázku vestavěné kamery na 9 polí

V další části se tedy pracuje s obrazem rozděleným na 9 částí pro posouzení vhodnosti těchto částí pro plnění zadané úlohy. Podle očekávání se průběh korelačních funkcí jednotlivých částí obrazu při simulovaném pohybu liší. Testováním souboru několika set obrázků bylo zjištěno, že v horní střední části obrázku je většinou část oblohy, mimo případů průjezdu pod mosty. Pro identifikaci polohy má přínos pouze onen případ průjezdů pod mostem. Ve střední části obrázku bývá korelace výrazně narušena momentální dopravní situací, ze které je přínos k určení polohy kamery prakticky nulový. Ve spodní střední části je zachycena vozovka, na které nebývají obrazové elementy použitelné k určení polohy, vyjma vodorovného dopravního značení. Z uvedeného plyne, že přínos jednotlivých částí zorného pole pro identifikaci polohy vozidla není stejný. V prvním přiblížení je možno střední pole ze zpracování vypustit, optimálnější je však použití váhových koeficientů pro zužitkování korelačních koeficientů uvedených částí obrazu.

Pro ověření této hypotézy byl sestaven program, který vytvářel korelační funkce obrazu původního a obrazu upraveného digitálním zoomem či obrazem získaným z jiného místa.

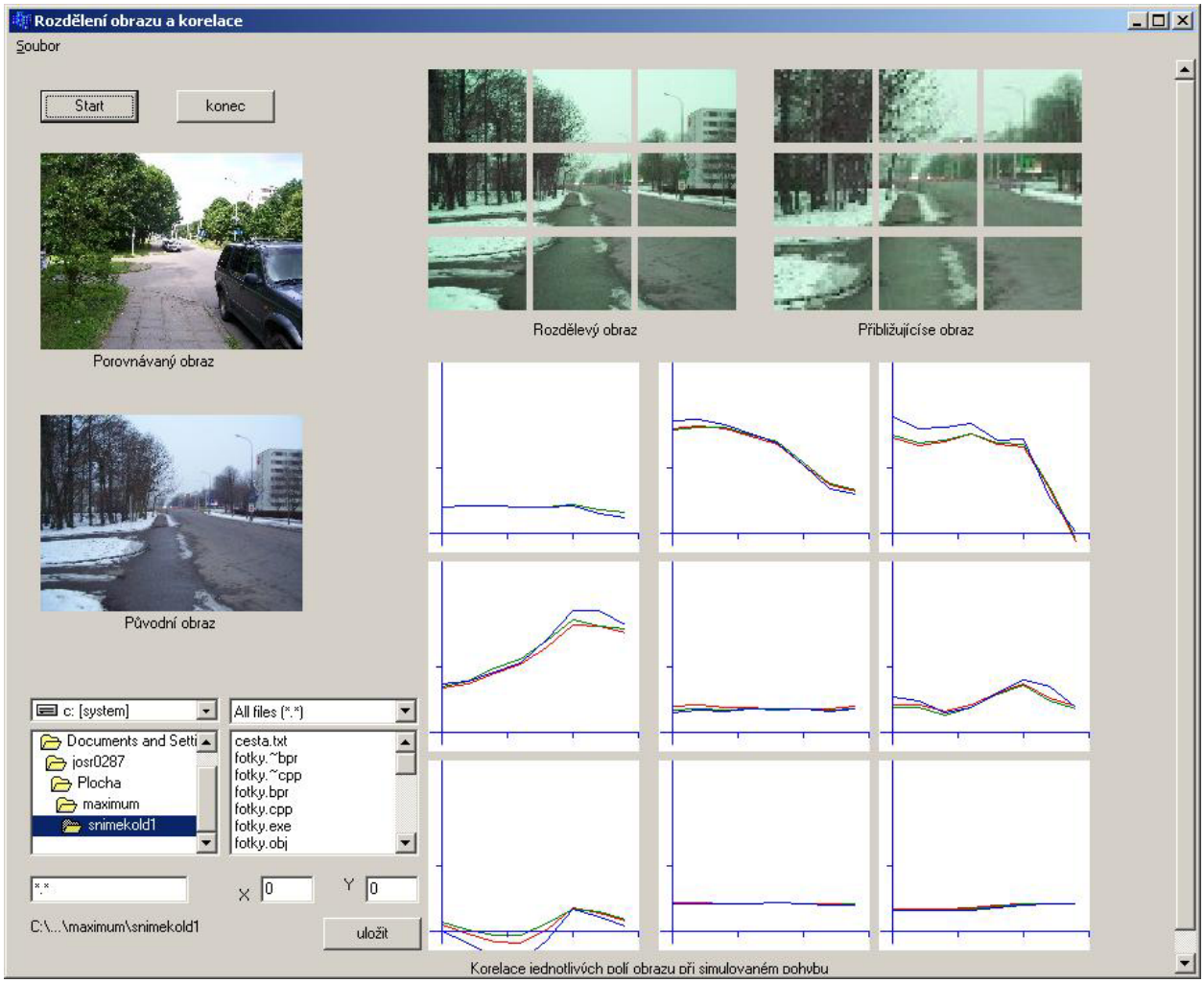
Podle předpokladů se skutečně objevila lokální maxima korelačních funkcí některých částí obrazu, i když původní obraz pocházel z úplně jiného místa.

5.3 Vybrané příklady korelací rozděleného obrazu.



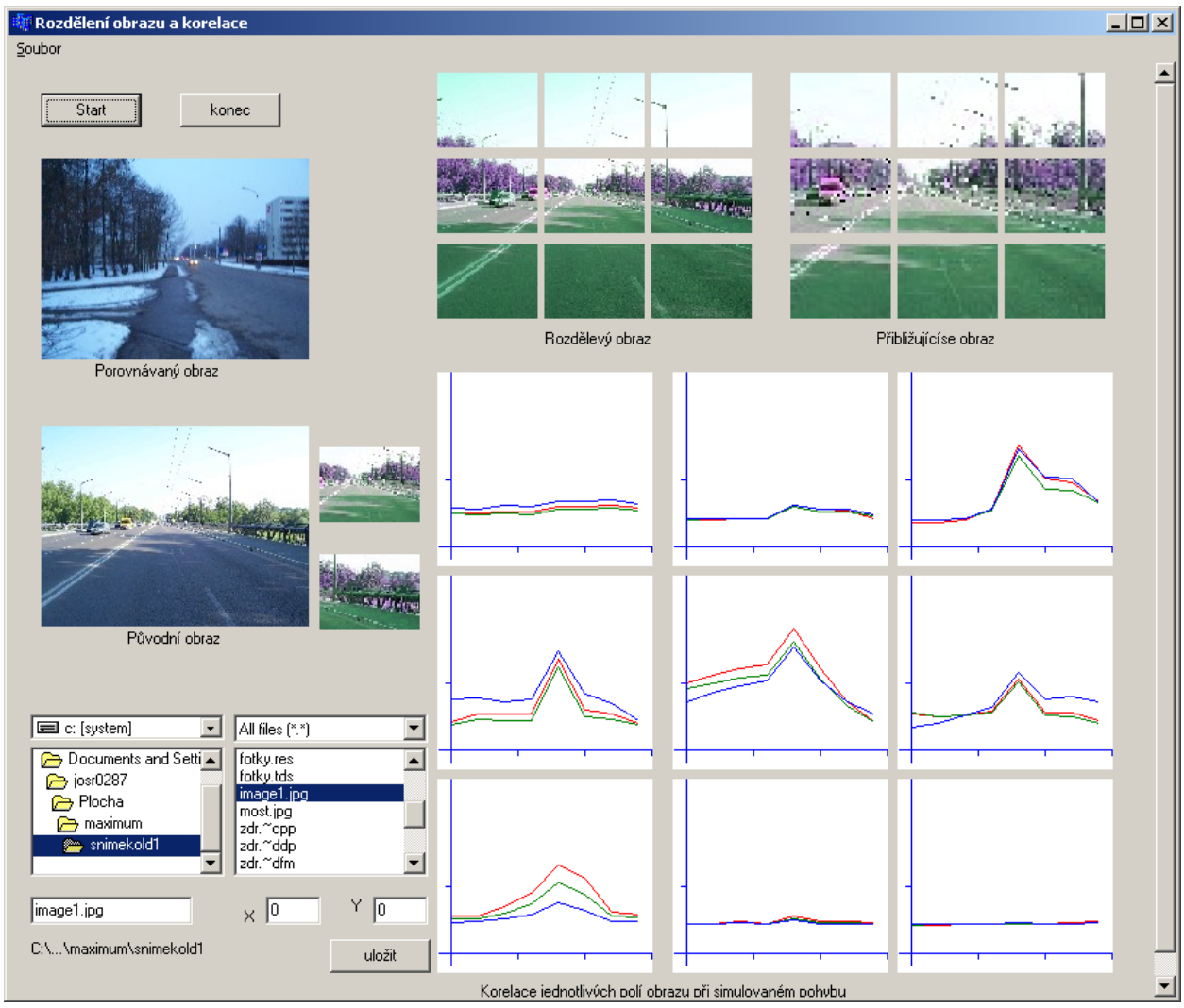
Obr. 31 Typický případ korelace jednotlivých částí rozděleného obrazu

Na Obr. 31 typický případ korelačních funkcí jednotlivých částí rozděleného obrazu. Části 3a a 3b nemají žádný vrchol na korelační funkci. Na obrázku vidíme, že se jedná o část oblohy, kde se nevyskytují žádné markanty. V prostřední vodorovné řadě korelací jsou jasně patrná korelační maxima, která jednoznačně identifikují polohu vozidla při tomto simulovaném posunutí. Dolní třetina snímku zobrazuje část vozovky s nevýrazným povrchem. Proto se na korelačních funkcích částí 1b a 1c neobjevují vrcholy. Plochý vrchol v části 1a je způsoben přítomností vodorovného dopravního značení (dvojitá čára), které se zde nachází.



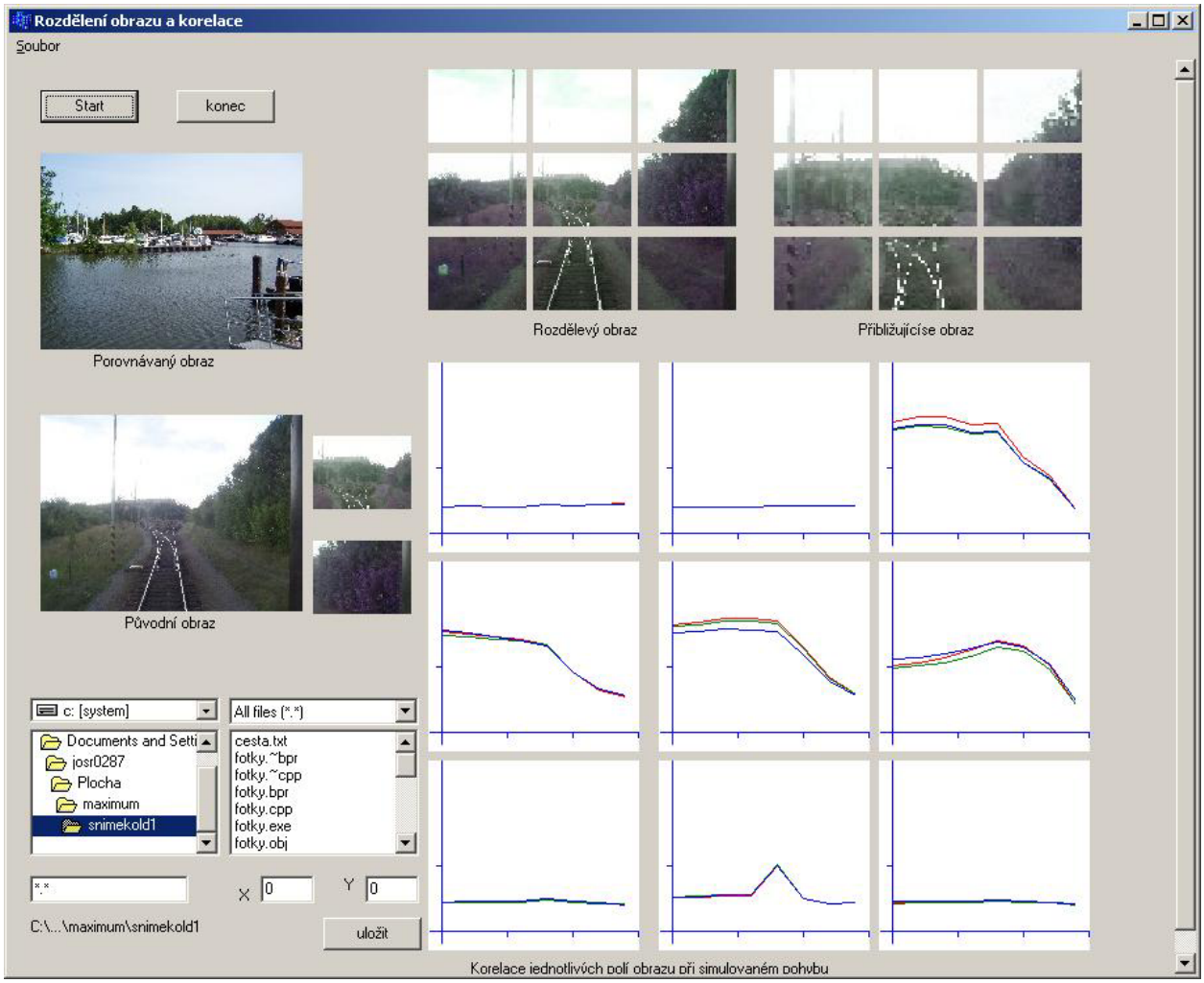
Obr. 32 Korelace obrazu, který se v čase změnil

Na Obr. 32 je zachycena situace scéna v zimě (únor 2006) a v létě. Snímky jsou pořízeny ze stejného místa, ale vrcholy na korelačních funkcích se hledají velmi těžko. Scéna během půl roku výrazně změnila svou tvář – zmizel sníh, na stromech se objevilo listí. Budova v pozadí, která je v zimě vidět skrz stromy, v létě úplně zmizí. Obraz je navíc narušen dopravní situací (vozidlo). V částech 3b a 3c se vyskytují vysoké ploché vrcholy korelační funkce způsobené horizontem.



Obr. 33 Korelace dvou nesouvisejících obrázků

Na Obr. 33 je příklad korelačních funkcí u dvou nesouvisejících obrazů, které přesto mají v příslušných místech vrcholy na křivkách. Jsou způsobeny ubíhajícími okraji vozovky, které se jeví jako velice podobné.



Obr. 34 Příklad korelace nesouvisejících obrázků

Na Obr. 34 jsou dva nesouvisející obrazy mající v sektorech 2c a 3c plochá lokální maxima na korelačních funkcích, ač je zřejmé, že celkové obrazy spolu vůbec nesouvisí. Další maxima často nacházíme v oblasti 3b, která na většině snímků zachycuje oblohu, a na oblast 1b, která na většině snímků zachycuje vozovku či trať. Naopak oblast 2b zachycuje měnící se dopravní situaci a často vypadá naprosto odlišně i v záběrech z téhož místa. Proto bylo provedeno statistické šetření pro získání věrohodnosti korelačních koeficientů jednotlivých polí. V průběhu 2 let byly pořizovány snímky různých situací na silnicích a železnici. Pomocí uvedené metody (digitální zoom) se provádělo šetření, jak by mohly vypadat průběhy korelačních funkcí při zpracování snímků z kamery umístěné na jedoucím vozidle. Samozřejmě, zoom nemůže generovat přesný obraz, který by zachytila kamera z jiného místa, ale pro další postup řešení tato metoda byla velmi užitečná, neboť umožnila jednoduše zpracovat velké množství obrázků z mnoha míst, neboť ne vždy bylo k disposici vozidlo pro snímání za pohybu. Pokusy o vytvoření filmů pomocí série snímků z kamery nesené v ruce daly nepoužitelné výsledky. Příčinou bylo nepřesné směrování kamery pomocí hledáčku a udržení roviny horizontu. Použití filmových záběrů poskytne sice výsledky přesnější, ale je mnohem pracnější a nebylo by možno zpracovat tak veliké množství dat. Následující tabulky zobrazují pravděpodobnost, že na korelační funkci se vyskytuje použitelné lokální maximum. Byly sestaveny na základě vyhodnocení velkého množství snímků. Původní podrobné tabulky jsou v příloze jako Tab. 3 a Tab. 4. V nich jsou pouze koeficienty 1 = korelační funkce má použitelné maximum a 0 = korelační funkci pro identifikaci nelze použít. Koeficient byl očekáván uprostřed grafu korelační funkce, kde se porovnávaly stejné obrázky ve stejném měřítku. Další požadavek byl,

aby graf v tomto místě zobrazoval ostré maximum s úrovní nejméně 20% nad zbývající částí grafu. Vyhodnocování se provádělo zpočátku ručně, později pomocí jednoduchého programu.

Četnost			Koeficienty		
0,62	0,17	0,68	0,120	0,033	0,132
0,76	0,37	0,81	0,148	0,072	0,157
0,67	0,28	0,79	0,130	0,054	0,153

Tab. 1 Věrohodnost korelačních koeficientů u silničních vozidel

Četnost			Koeficienty		
0,63	0,28	0,64	0,131	0,058	0,133
0,66	0,59	0,67	0,137	0,123	0,140
0,58	0,16	0,59	0,121	0,033	0,123

Tab. 2 Věrohodnost korelačních koeficientů u kolejových vozidel

5.4 Zpracování korelačních funkcí

Nejprve se vytvoří korelační funkce tvořená korelačními koeficienty obrazu kamery se snímky dříve uloženými. Korelační funkce se generuje pro každou barevnou složku samostatně. Výsledná korelační funkce se vytvoří jako eukleidovská vzdálenost barevných složek v každém bodě funkce:

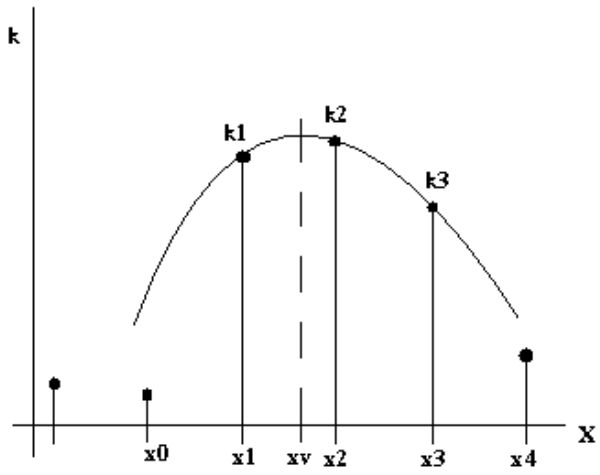
$$k_V = \sqrt{k_R^2 + k_G^2 + k_B^2} \qquad (5.8)$$

kde
k_V je výsledný korelační koeficient
k_R, k_G a k_B jsou korelační koeficienty jednotlivých barevných složek

Na snímcích je patrné, že ve většině případů se korelační křivky jednotlivých barev podobají. Výraznější absence některé složky u výrazně barevných (jednobarevných) objektů nejsou překážkou. U obrazů velmi rozdílných bude vrchol korelační funkce malý, u obrázků z původního místa bude největší. Lze předpokládat, že nejen obrázek, který byl uložen v místě, kde se momentálně nalézá kamera, bude mít velký vrchol funkce, ale i obrázky sousední viz Obr. 35. Toho lze využít k dalšímu zpřesnění určení polohy. Pro další zpracování vezmeme nejvyšší nalezený koeficient a koeficienty předchozího a následujícího snímku. Těmito 3 koeficienty proložíme aproximační parabolu

$$y = ax^2 + bx + c \qquad (5.9)$$

a vyhledáme její vrchol. Souřadnici vrcholu budeme považovat za hledaný bod.
Předpokládáme ekvidistantní snímky s roztečí $\Delta x = h$



Obr. 35 Aproximační parabola

Pro uvedené 3 body platí:

$$\begin{aligned} a_1^2 + bx_1 + c &= k_1 \\ ax_2^2 + bx_2 + c &= k_2 \\ ax_3^2 + bx_3 + c &= k_3 \end{aligned} \qquad (5.10)$$

Pro zjednodušení provedeme transformaci souřadnic posunutím do bodu $x_1 = 0$, takže bude

$$x_2 = h \text{ a } x_3 = 2h \qquad (5.11)$$

Rovnice se zjednoduší na:

$$\begin{aligned} c &= k_1 \\ ah^2 + bh + c &= k_2 \\ 4ah^2 + bh + c &= k_3 \end{aligned} \qquad (5.12)$$

Řešením této soustavy dostaneme koeficienty aproximační paraboly:

$$\begin{aligned} c &= k_1 \\ b &= \frac{4k_2 - 3k_1 - k_3}{2h} \\ a &= \frac{k_1 - 2k_2 + k_3}{2h^2} \end{aligned} \qquad (5.13)$$

U této aproximační paraboly $y = ax^2 + bx + c$ nalezneme extrém:

$$y' = 2ax + b \overset{!}{=} 0 \qquad (5.14)$$

$$x = -\frac{b}{2a} \qquad (5.15)$$

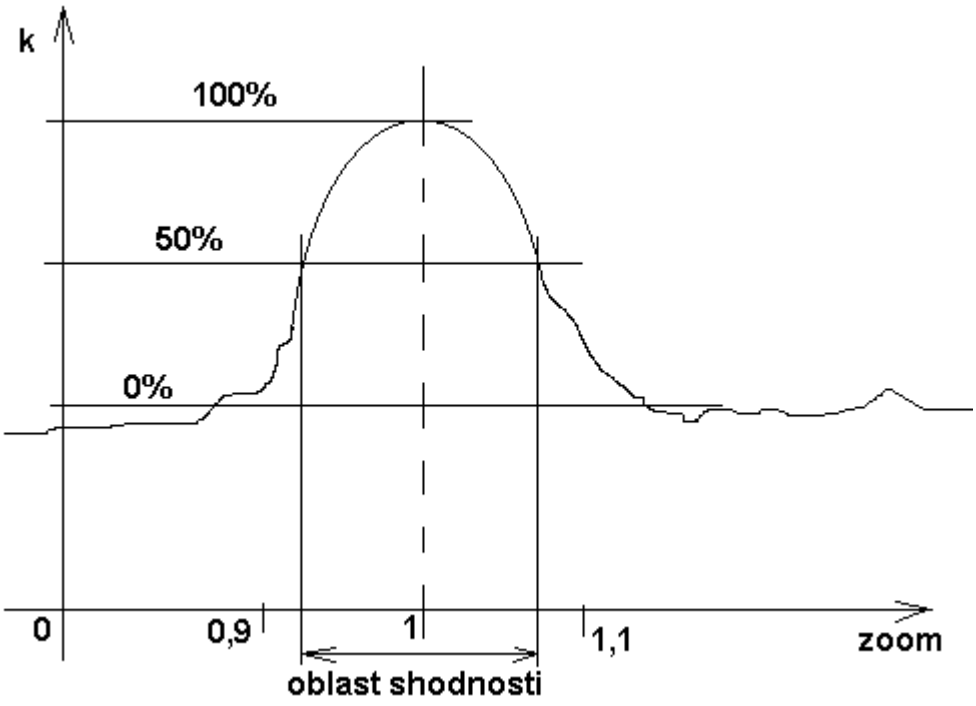
Toto je poloha vrcholu vzhledem k posunutým souřadnicím, skutečná poloha tedy bude:

$$x_v = x_1 + x \qquad (5.16)$$

Pro případ a=0 (body leží na jedné přímce) prostě musíme jako souřadnici x považovat souřadnici snímku s nejvyšším korelačním koeficientem. Sice se jedná o velice vyjímečné případy (korelační koeficienty mají hodnotu > 10^3), ale vytvářený algoritmus musí zabránit dělení nulou.

5.5 Šířka a výška korelačních koeficientů

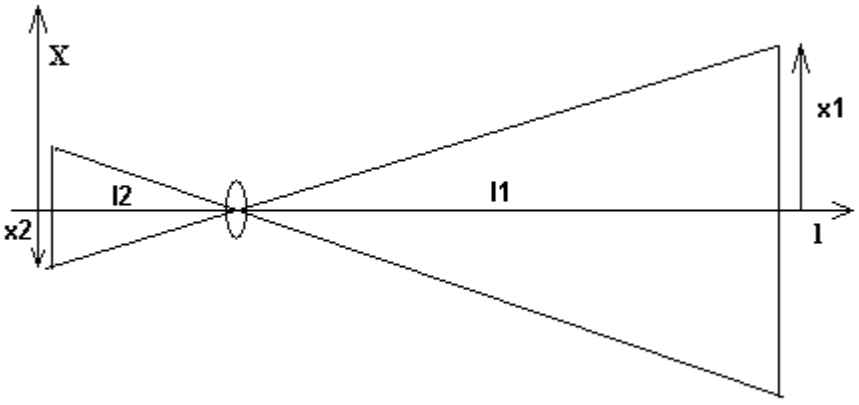
Teoretické hodnoty, odvozené z rozlišovací schopnosti kamery vyžadují extrémně vysokou frekvenci snímků. Pro použití nižší frekvence je nutno najít vhodný kompromis. Tím je využití šířky korelačního koeficientu zjištěného měřením reálných situací. Pro tento účel byl vytvořen program Fotky.exe, který se později stal základem pro knihovnu algoritmů řešících tuto úlohu. Byly v něm testovány vybrané záběry z kamery, pořízené z vozidla jedoucího po silnici, po železnici a ve volném terénu. Program s každým snímkem provede zvětšení resp. zmenšení změnou měřítka, čímž simuluje dopředný, resp. zpětný posuv vozidla s kamerou. Vyhodnotí se korelace původního snímku se snímkem digitálně zvětšeným, resp. zmenšeným. Nejvyšší hodnoty korelační funkce se dosáhne samozřejmě při měřítku 1:1 (obdoba autokorelační funkci v bodě nula) viz Obr. 36. Ve většině případů korelační funkce od svého vrcholu neklesá na obě strany až k nule, ale k nějaké nenulové hodnotě. Od tohoto místa se korelační koeficient mění již jen velmi málo.
Diskutabilní je otázka zavedení prahu, kdy ještě hodnotu korelace považujeme za příznak shodnosti obrázků a kdy již nikoliv. V této práci byla prostě stanovena jako hodnota, ve které je převýšení vyšší, než 50% maxima. Velikost převýšení nám vypovídá o tom, zda obraz obsahuje dostatek prvků k identifikaci. Pokud lokální maximum není vyšší, než 20% okolní hodnoty funkce, máme za to, že průběh lokální maximum neobsahuje. Toto je důležitá informace, která podmiňuje použití koeficientu v dalších metodách zpracování.



Obr. 36 Definice šířky korelačního koeficientu

5.6 Určení frekvence snímání

Pro praktické použití je zapotřebí, aby několik po sobě následujících snímků poskytlo nadprůměrnou korelaci. To je samozřejmě závislé na více faktorech, zejména na velikosti předmětů v okolí dráhy. Pro teoretické výpočty lze použít různé velikosti, nejužitečnější se však jeví snímání reálných scén, jejich vyhodnocení a zprůměrňování. Předpokládáme kameru, která má zorný úhel β a poskytuje obraz, který má na jednom řádku n obrazových bodů. Dále předpokládáme, že je vybavena ideálním objektivem, zprostředkujícím následující zobrazení:



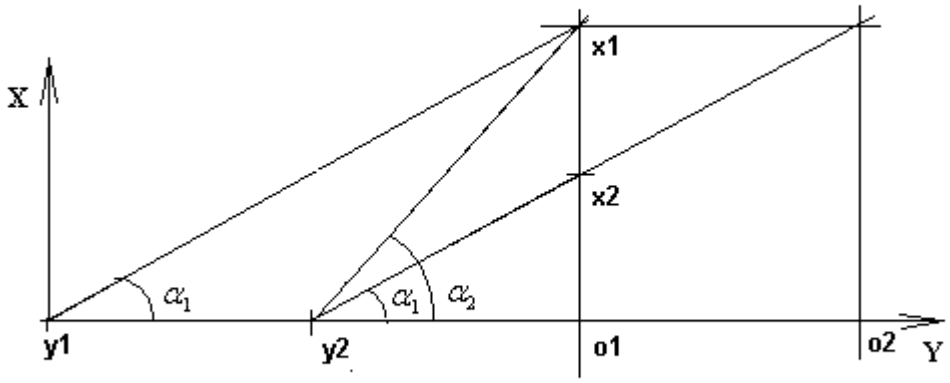
Obr. 37 Zobrazení objektivem

$$\frac{x_1}{l_1} = -\frac{x_2}{l_2} \qquad (5.17)$$

neboli:

$$\frac{x_1}{x_2} = -\frac{l_1}{l_2} \qquad (5.18)$$

Znaménko mínus jen zahrnuje fakt, že na obrazovém čipu snímací kamery vzniká převrácený obraz. Vlivem dalšího zpracování obrazu se toto otočení eliminuje.
Při posunutí kamery vpřed (v rozsahu její hloubky ostrosti) se nám obraz zvětší; resp. při posunutí kamery zpět se obraz zmenší podle Obr. 38:

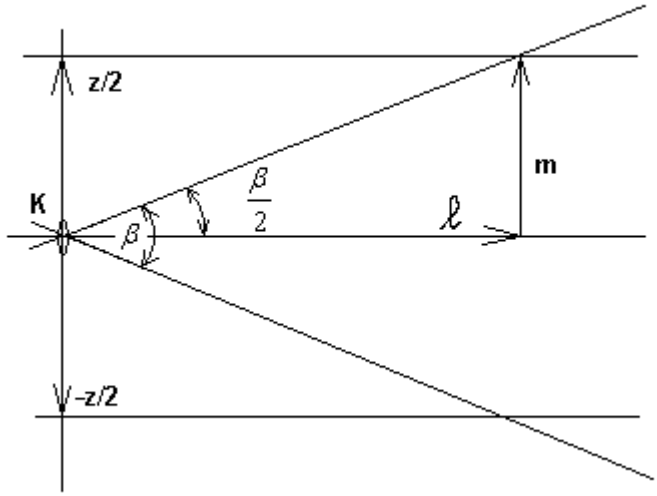


Obr. 38 Zobrazení při posuvu kamery

V bodě y_1 je předmět x_1 viditelný pod úhlem α_1, při posunutí kamery do bodu y_2 je viditelný pod úhlem α_2. Pod původním úhlem α_1 se zobrazí oblast bodu x_2. Pokud se bude jednat o body na okraji zobrazovaného pole, které nás zajímají nejvíc, budeme předpokládat, že body x_1 a x_2 se nacházejí na okraji zorného pole a každý má svou informační hodnotu, kterou je nutno odlišit alespoň jedním obrazovým bitem.
Pro zjednodušení si dovolíme určitou aproximaci, a sice, že všechny obrazové bity snímače zachycují stejný prostorový úhel od objektivu. To sice není nikdy splněno, ale u běžných kamer se snímacím úhlem do 50^o chyba nepřekročí poměr 1:1,5. Velikost je silně závislá na provedení objektivu, proto přesnější výpočty bez znalostí hodnot těchto konkrétních závislostí postrádají smysl.
Potom u kamery se snímacím úhlem β a počtem n pixelů na jeden rozměr obrázku bude rozdíl

$$(\alpha_1 - \alpha_2) = \frac{\beta}{n} \qquad (5.19)$$

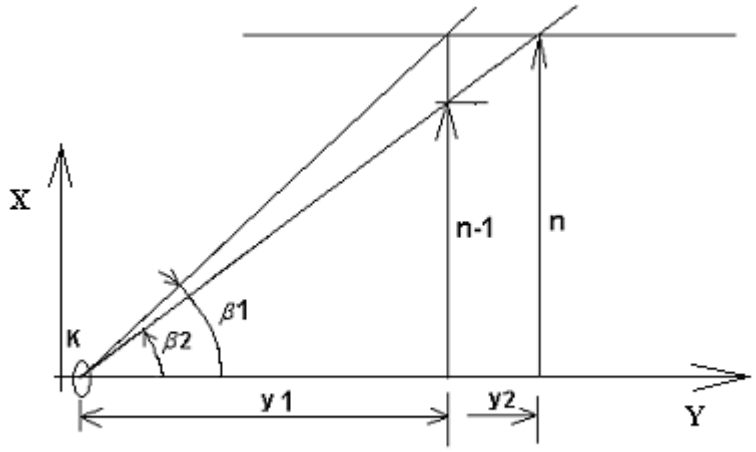
Pro určení konkrétní rychlosti nyní potřebujeme znát vzdálenost konkrétních sledovaných předmětů. Jak bylo zmíněno dříve, situace na dopravní cestě (a nad ní – mraky) nepřispívá k určení polohy vozidla, proto neuvažujeme tyto blízké objekty. Nejbližší objekty se tedy budou nacházet za okrajem dopravní cesty – stromy, dopravní značky, stavby. Šířku dopravní cesty můžeme předpokládat, nejbližší objekty tedy budou ty, které jsou na okraji cesty šířky 2m a právě opouštějí zorné pole kamery dle Obr. 39:



Obr. 39 Zobrazení okrajů dopravní cesty

$$\frac{z}{2} = l.tg\frac{\beta}{2} \qquad (5.20)$$

Při posunutí kamery (zpět) tak, aby se na obraze okraj posunul o 1 pixel ke středu se jedna strana cesty široké z zobrazuje na n/2 pixelech, jak je na Obr. 40:



Obr. 40 Minimální posuv kamery na dopravní cestě

$$\frac{z}{2} = y.tg\frac{\beta}{2} \qquad (5.21)$$

Sousední obrazový pixel zobrazuje předmět pod úhlem menším, zobrazí tedy předměty nacházející se na okraji dopravní cesty ve větší vzdálenosti. Šíře cesty z odpovídá n pixelům, jednomu pixelu tedy odpovídá šířka

$$\Delta z = \frac{z}{n} \qquad (5.22)$$

Úhel 2. pixelu od okraje:

$$\frac{\beta_1}{\beta_2} = \frac{n}{n-1} \tag{5.23}$$

$$\beta_2 = \beta_1 \frac{n-1}{n} \tag{5.24}$$

$$y_1 = \frac{z}{tg\beta_1} \tag{5.25}$$

$$y_2 = \frac{z}{tg\left(\beta_1 \frac{n-1}{n}\right)} \tag{5.26}$$

Příklad:

Z=5m, n=100 pixelů, $\beta=60^o$

Dostaneme:

y_1=8,660

y_2=8,766

Rozdíl činí 10,6 cm. Při rychlosti 36km/hod = 10m/sec je potřebná frekvence snímků cca 100Hz. Tato vysoká frekvence při poměrně nízké rychlosti vozidla odpovídá výchozím předpokladům, kde požadujeme nový snímek v okamžiku, kdy se obraz v okrajové oblasti posune o 1 pixel. Tento předpoklad je nutno oslabit. Při posunutí obrazu o 1 pixel máme stále použitelný obraz. Zobrazované objekty jsou ve většině případů zobrazeny větší, než právě jeden pixel a tudíž poskytnou použitelnou korelaci s více snímky. Uvedený výsledek lze tedy spíše považovat za citlivost algoritmu na 1 pixel a za hranici, kde další zvyšování obrazové frekvence již nepřináší ani teoretický zisk.

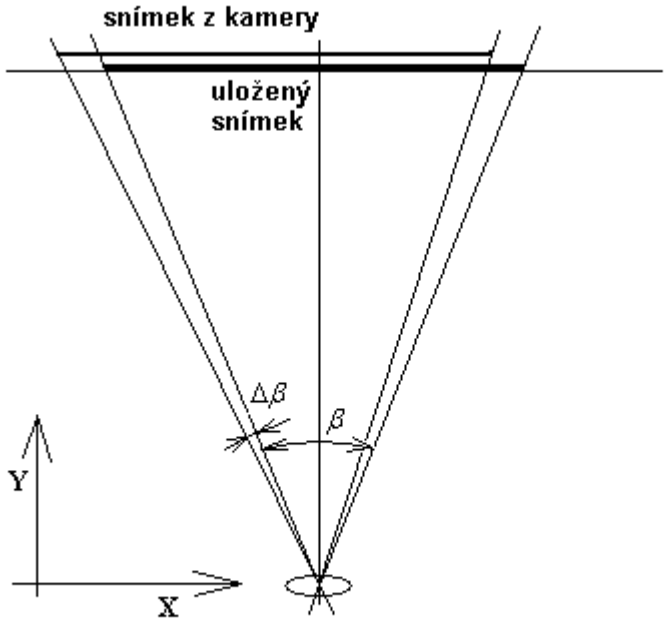
Druhým extrémem je situace, kdy obrazovaný objekt zabírá plochu celého dílčího obrázku. Dojdeme sice k extrémně nižší frekvenci snímků, ovšem v takovém případě nemůžeme očekávat nějaké výsledky získané pomocí korelací obrazů, neboť bude každý z jiného místa bez vzájemného překrytí.

5.7 Detekce stranové odchylky

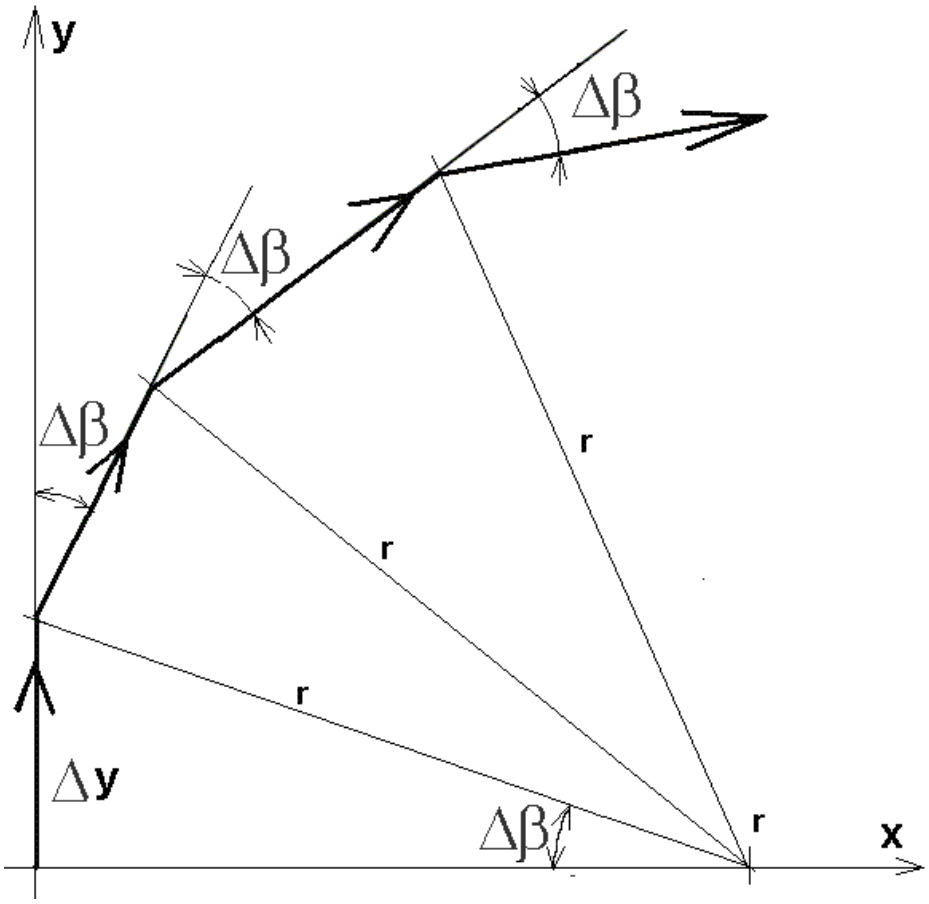
Vychýlení kamery se projeví posuvem obrazu doleva resp. doprava viz Obr. 41 . Vzniklým nesouladem obrázku z kamery a jeho vzorem v počítači dojde ke snížení korelačního koeficientu. Detekce je jednoduchá – prostě obraz posuneme vlevo, resp. Vpravo a provedeme další korelaci. Porovnáním korelačních koeficientů můžeme vyhodnotit, zda ke stranovému posuvu skutečně došlo. Pokud použijeme tuto metodu, musíme místo jedné korelace provádět ještě minimálně dvě další. Zde se evidentně projevuje požadavek paralelního zpracování obrazů.

Předpokládáme stranovou odchylku uloženého obrazu od obrazu z kamery $\Delta\beta$

Dále předpokládáme, že se nejedná o nahodilou chybu, ale je způsobena průjezdem vozidla zatáčkou. Charakteristické je, že u několika dalších snímků se tento boční posuv bude opakovat.



Obr. 41 Stranové vychýlení kamery



Obr. 42 Nájezd vozidla s kamerou do zatáčky

$$\Delta y = r.\sin\Delta\beta \qquad (5.27)$$

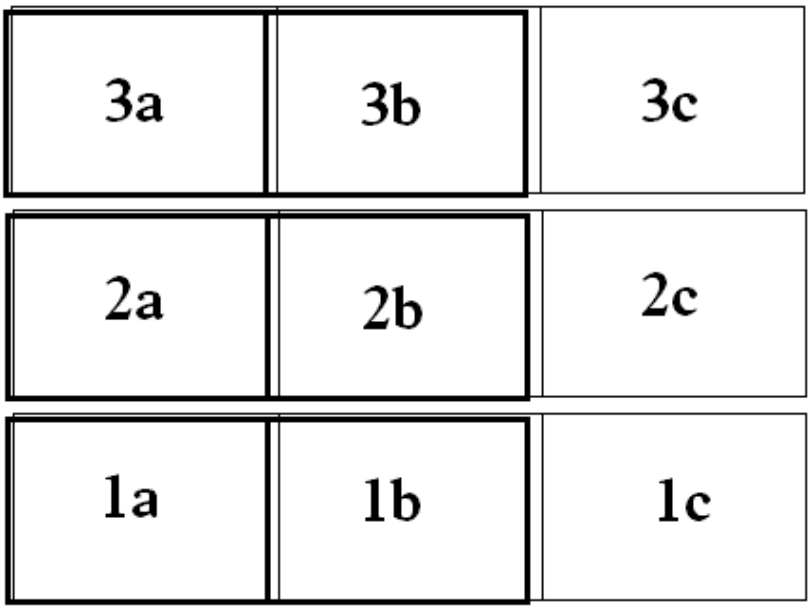
$$r = \frac{\Delta y}{\sin\Delta\beta} \qquad (5.28)$$

Z toho plyne, že určení stranové odchylky má klíčový význam v případě, že se má podle obrazu snímaného kamerou provádět navigace vozidla. Samotné zjištění, že dojde ke stranové odchylce je nedostačující informace, teprve zjištění poloměru otáčení umožní řízení směru jízdy. Další poznatek je, že vozidlo bude reagovat se zpožděním. Nejprve musí dojít k rozdílu v korelaci, ze kterého se stranová odchylka vyhodnotí a potom teprve vozidlo může reagovat, tedy pokračuje v přímé jízdě alespoň do místa dalšího snímku.
Při ukončení zatáčky je situace obdobná – vozidlo zatáčí ještě v době, kdy má mít nastaveno řízení na přímý směr. V případě zařazení případných filtrací pomocí více snímků mohou být prodlevy ještě delší. Toto vychýlení z původní dráhy může být často neakceptovatelné a je nutno jej korigovat. Možné korekce jsou při vytváření obrazových vzorů. První možnost je vychýlení snímací kamery již při vytváření prvotního záznamu v závislosti na řízení, tedy dříve, než dojde k zatáčení vozidla. Druhá možnost je úprava pořízeného prvotního záznamu ještě před jeho použitím.

Tyto prodlevy musí korigovat algoritmy řízení vozidla, což ale již nepatří do náplně této práce.

Stranový posuv obrazu, ať způsobený nepřesností vedení vozidla nebo jeho nájezdem do zatáčky, je rušivý element. Při první (vzorové) jízdě se ukládá pouze posloupnost snímků podle pohybu kamery vpřed. Jiná situace by nastala, kdyby se snímání provádělo více kamerami, což však není tento případ. Proto se detekce stranového pohybu provádí korelací se vzorem digitálně posunutým vlevo, resp. vpravo. Pokud je korelační koeficient s takto posunutým obrazem vyšší, než korel. koef. s obrazem neposunutým, je to příznak stranového vychýlení kamery. Následný algoritmus má několik možností:

- Odchylku ignorovat či tuto detekci vůbec neprovádět. Přichází v úvahu u kolejových vozidel.
- Odchylku pouze indikovat
- Provést další korelace s obrazy posunutými o jinou hodnotu a pomocí těchto dalších korelací vyhodnotit změnu směru jízdy.



Obr. 43 Posunutí obrazu pro detekci stranové odchylky

Skutečné posunutí obrazu se provede jen s 6 částmi z devíti – prostřední sloupec a příslušný krajní Obr. 43. U druhého okraje by došlo ke komplikacím vzhledem k chybějící části, proto se jednoduše tato část vynechá. Ukazuje se, že při detekci stranového posuvu tímto způsobem nevznikají problémy. Před vytvořením vzoru stranového posuvu je nutno obraz rekonstruovat, neboli doplnit zpět střední hodnoty barevných složek a následně u vytvořených posunutých částí je opět odstranit. Je to vlastně jediný případ, kdy se uložené střední hodnoty barevných složek použijí k něčemu jinému, než pro demonstraci uloženého obrázku v některém z oken aplikace.

5.8 Režimy jízdy při navigaci optickou kamerou

Při jízdě vozidla, které používá pro určení své polohy optickou kameru, může dojít ke ztrátě orientace vlivem různých okolností – výhled kamery je zastíněn, o okolí se nenacházejí dostatečně výrazné markanty atd. Proto můžeme sledování rozdělit do těchto tří kategorií:

a) **Souvislé sledování**. V tomto režimu je nepřetržitě k disposici údaj o posici, získaný korelací obrazů z kamery. Podmínkou je, že:
   - Snímky z kamery přicházejí s dostatečnou frekvencí
   - Korelační funkce mají dostatečně výrazná maxima, aby bylo možno jednoznačně určit souřadnice. V tomto režimu lze systém použít k řízení vozidla.

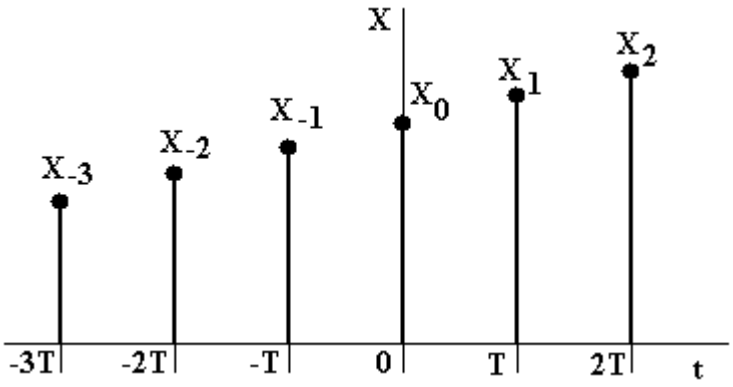
b) **Sledování s přerušením**. Vyznačuje se tím, že některé úseky dopravní cesty nemají dostatečné množství osobitých obrazových prvků, takže korelační funkce nemá výrazná maxima. V oblastech mezi úseky s použitelnou korelací musíme polohu určovat jinými způsoby, např. aproximací podle času a rychlosti z předchozího úseku cesty. V úsecích se ztrátou korelačních informace je nutno vozidlo řídit jiným způsobem, např. některým ze systémů INS.

c) **Vyhledávání orientačních bodů**. Některé části cesty mohou být na objekty vhodné k identifikaci velmi chudé, např. pouště, sněhové pláně či jízda v potrubí. Použití aproximace podle uplynulého času může být zavádějící, pokud nemáme spojitý údaj o rychlosti a směru jízdy. V tomto režimu pouze očekáváme, kdy se objeví obraz, který korelační koeficient poskytne vyšší než obraz okolí s některým z obrazů v dříve uloženém souboru. V tomto režimu vozidlo nemůže být řízeno pomocí tohoto systému, ale jiným způsobem (např. jízda po kolejích). Lze jej však použít jako duplicitní systém ve spolupráci s jiným navigačním systémem.

5.9 Filtrace a zpracování získaných souřadnic

5.9.1 Zpracování souřadnic

Předpokládáme, že máme z 1. průjezdu k dispozici sled snímků a jejich souřadnice. Vybereme si pro příklad souřadnici s označením x. Po vyjetí vozidla při opakované jízdě za dobu T získáme z kamery 1.snímek. Provedeme operaci porovnání (např. korelaci) tohoto snímku se snímky uloženými z 1.průjezdu. Při porovnání se snímkem ze stejného místa očekáváme podle kap. 5.1 vyšší korelační koeficient, než se snímky ostatními. Potom můžeme souřadnice zapsané o snímku z 1. průjezdu považovat za okamžitou souřadnici vozidla při 2. průjezdu. Za další dobu T pořídíme další snímek a porovnání zopakujeme a očekáváme získání další aktuální pozice viz Obr. 44



Obr. 44 Okamžité souřadnice vozidla

Kromě toho lze určit i okamžitou rychlost vozidla pomocí rozdílu souřadnic a doby T:

$$v = \frac{x_i - x_{i-1}}{T} \tag{5.29}$$

Kde x_i je souřadnice vozidla zapsaná u i-tého snímku. Z rozdílu rychlostí můžeme zkusit vypočítat i zrychlení, ale časový interval T je příliš malý a chyby souřadnic x_i velké, že tato veličina nebude použitelná.

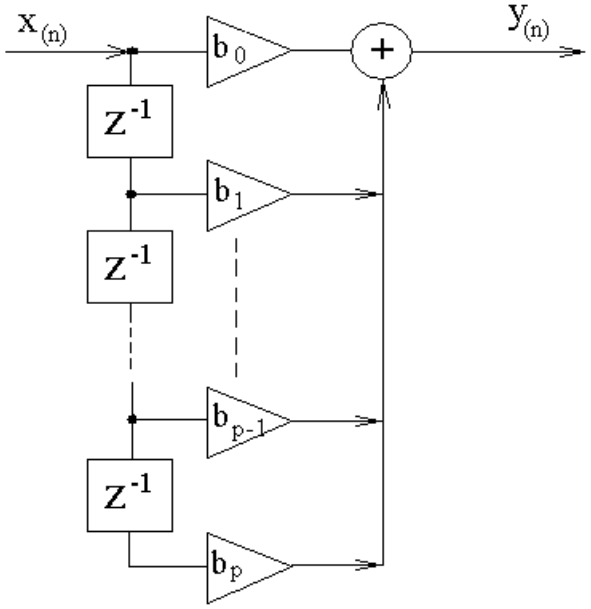
Tento teoretický princip je sice formálně správný, ale v praxi téměř nepoužitelný.

- Vzhledem ke konečnému výkonu technického vybavení se nám za dobu T nejspíš nepodaří provést korelace se všemi snímky z 1. průjezdu. Jednoduchou úvahou dojdeme k tomu, že by mělo stačit zkoumat snímky jen z okolí předpokládané nové polohy vozidla.
- Nový snímek může být nepoužitelný z některých důvodů uvedených v kap 5.3
- V uložené posloupnosti se mohou nacházet některé snímky, které jsou nepoužitelné z téhož důvodu.

Proto v praxi musíme použít t.zv. sledovací okno, které nám určí, které uložené snímky použijeme pro korelaci se snímkem novým, případně jakou váhu přiřadíme jednotlivým získaným korelačním koeficientům a jak s jejich pomocí získáme skutečnou souřadnici. Tím zároveň vzniká požadavek predikce příští polohy, podle které budeme muset nastavit souřadnice okna.

5.9.2 Určení okamžité polohy pomocí nalezených maxim korelací

Předpokládáme, že pomocí korelací byla určena posloupnost n souřadnic $\{x_{1-n}, x_{2-n},\ldots,x_{-1},x_0\}$ v časech $\{-nT, -(n-1)T,\ldots,-T,0\}$. Každá ze souřadnic je zatížena určitou chybou. Pro zpřesnění použijeme metodu MA (Noviny average model), zvanou též klouzavý průměr. Tento model má tvar na Obr. 45



Obr. 45 Model MA (moving average)

Potom pro souřadnici x_s platí

$$x_s = b_{p-1}\left[x_{1-p} + (1-p)vT\right] + b_{p-2}\left[x_{2-p} + (2-p)vT\right] + \ldots + b_{-1}\left(x_{-1} + vT\right) + b_0 x_0 =$$
$$= \sum_{i=1}^{p} b_i \left[x_{i-p} + (p-i)vT\right] \qquad (5.30)$$

Přičemž b_i jsou koeficienty okna a musí pro ně platit:

$$\sum_{i=1}^{p} b_i = 1 \qquad (5.31)$$

Tato problematika je podrobně rozpracována např. v radiolokaci.

5.9.3 Určení rychlosti vozidla

Předpokládáme, že vozidlo má konstantní rychlost. Tento předpoklad je dobře splněn, pokud si uvědomíme, kolik snímků se musí zpracovat intervalu, než vozidlo významněji změní rychlost. Určení rychlosti z jediného rozdílu souřadnic, jak bylo naznačeno v kap. 5.9.1 je nepřesné a pro predikci příští polohy a tedy i nastavení okna značně riskantní. Pokud tedy máme k dispozici posloupnost n souřadnic x_1 až x_n, můžeme rychlost přesněji určit z rozdílu souřadnic krajních hodnot

$$v = \frac{x_n - x_1}{(n-1)T} \qquad (5.32)$$

Pokud v posloupnosti nechybí žádný člen, např. z důvodu neúspěšné korelace, můžeme použít známou mnohem přesnější vyrovnávací metodu pro dráhu *l* ujetou v daném intervalu :

$$l = \frac{n-1}{(k-1)(n-k+1)} \sum_{i=0}^{n-k} \left(x_{k+i} - x_{i+1}\right) \qquad (5.33)$$

Kde k=n/2 nebo n/2+1; k celé.

Potom bude rychlost v:

$$v = \frac{l}{(n-1)T} \tag{5.34}$$

5.9.4 Predikce polohy

Predikce polohy je nezbytná pro práci se sledovacím oknem. Na predikovanou polohu nastavujeme střed sledovacího okna. Nejjednodušší případ je, když máme k dispozici výsledky (souřadnice) předchozích porovnání až do posledního okamžiku. Potom můžeme určit rychlost vozidla pomocí (5.33) a určit souřadnici následujícího snímku, pro který bude platit:

$$x_{s+1} = x_s + vT \tag{5.35}$$

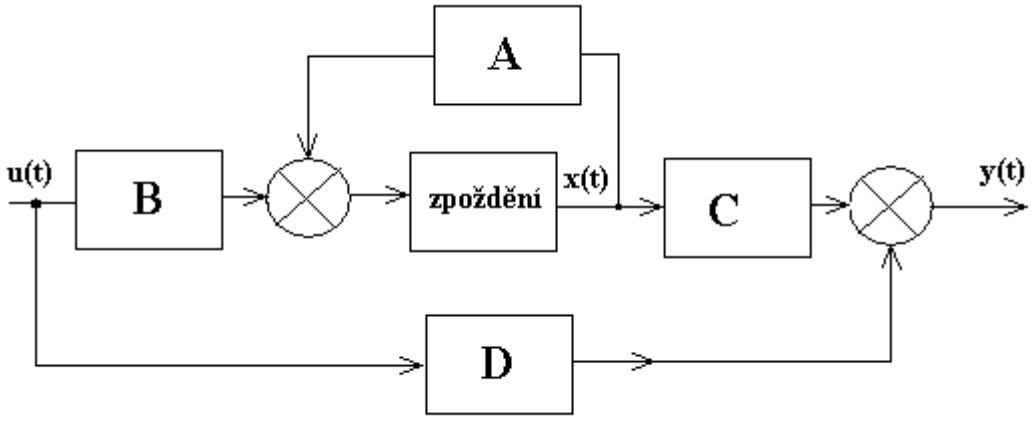
Pokud se však vozidlo pohybuje v místech, kde se nedaří určit souřadnice korelací snímků z kamery a snímků uložených, což může nastat např. zacloněním výhledu sousedním dopravním prostředkem, potom musíme provádět odhad souřadnic pro další snímky, viz Obr. 44. Pro ně potom platí:

$$x_{s+n} = x_s + nvT \tag{5.36}$$

Současně musíme sledovací okno rozšířit, neboť vozidlo může během ztráty sledování změnit rychlost. Algoritmus sledování musí být samozřejmě na takovouto situaci připraven a pokud se nepodaří určit správnou polohu po určitou dobu, musí toto signalizovat jako výpadek ze sledování.

5.9.5 Použití Kalmanova filtru pro predikci polohy

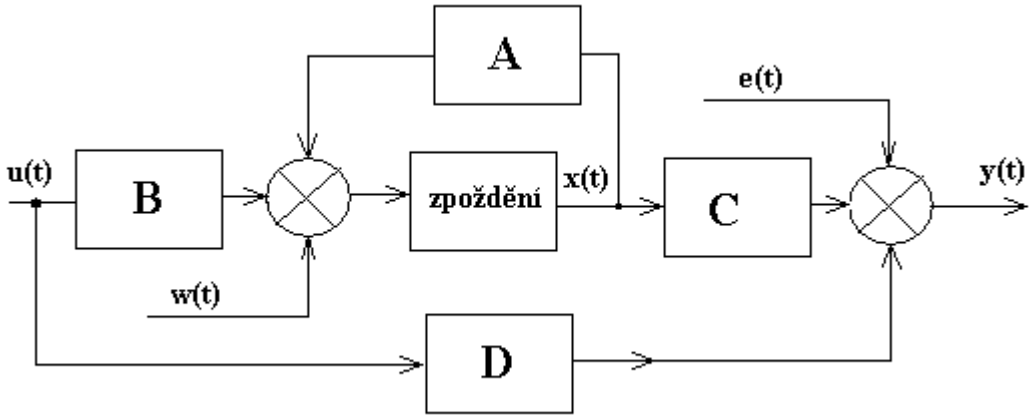
Použití Kalmanova filtru je dnes velice oblíbené a bez jeho uvedení by práce nebyla úplná. Základní teorii najdeme třeba v [36]. Dále uvedené vztahy podporuje knihovna programů OpenCV [14] i programový balík Matlab [30]. Filtr vychází ze známého stavového modelu na Obr. 46. popsaného rovnicemi (5.37)



Obr. 46 Stavový prostor

$$\vec{x}(t+T) = \vec{A}\vec{x}(t) + \vec{B}\vec{u}(t)$$
$$\vec{y}(t) = \vec{C}\vec{x}(t) + \vec{D}\vec{u}(t) \qquad .(5.37)$$

Pro účel filtrace doplněný o vstup rušivých signálů w(t) a e(t) na Obr. 47 popsaný rovnicemi (5.38):



Obr. 47 Stavový prostor s šumovými vstupy

$$\vec{x}(t+T) = \vec{A}\vec{x}(t) + \vec{B}\vec{u}(t) + \vec{w}(t)$$
$$\vec{y}(t) = \vec{C}\vec{x}(t) + \vec{D}\vec{u}(t) + \vec{e}(t) \qquad (5.38)$$

Protože systém neřídíme, ale jen pozorujeme, bude u=0. Potom se systém zjednoduší na:

$$\vec{x}(t+T) = \vec{A}\vec{x}(t) + \vec{w}(t)$$
$$\vec{y}(t) = \vec{C}\vec{x}(t) + \vec{e}(t) \qquad (5.39)$$

Pro pohyb objektu můžeme specifikovat polohu x(t), rychlost v(t) a zrychlení a(t), což můžeme popsat rovnicemi

$$v(t) = \dot{x}(t) = \frac{dx(t)}{dt} \qquad (5.40)$$

$$a(t) = \dot{v}(t) = \frac{dv(t)}{dt} \qquad (5.41)$$

Máme tedy 3 stavové proměnné:

$$\begin{bmatrix} x \\ v \\ a \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \qquad (5.42)$$

Konkrétně pro náš případ platí tento popis:

$$\begin{aligned} x(t+T) &= x(t) + v(t)T + \frac{1}{2}aT^2 \\ v(t+T) &= v(t) + a(t)T \end{aligned} \tag{5.43}$$

Přepíšeme do tvaru stavových rovnic:

$$\vec{x}(t+T) = \vec{A}(T)x(t) + \vec{w}(t) \tag{5.44}$$

$$\begin{bmatrix} x(t+T) \\ \dot{x}(t+T) \\ \ddot{x}(t+T) \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} + \begin{bmatrix} w_1(t) \\ w_2(t) \\ w_3(t) \end{bmatrix} \tag{5.45}$$

T je čas mezi jednotlivými snímky.

Porovnáním předchozích rovnic vidíme, že matice A má tvar:

$$\vec{A} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{5.46}$$

Ze stavových proměnných máme ovšem k dispozici pouze souřadnici x. Proto pro vztah (5.38) musí platit

$$\vec{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{5.47}$$

dosazením dostaneme

$$\vec{y}(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} * \vec{v}(t) \tag{5.48}$$

Funkce e(t) a w(t) pokládáme za náhodné procesy, jejichž kovarianční matice má tvar:

$$\varepsilon\left\{\begin{bmatrix}\vec{w}(t)\\ \vec{e}(t)\end{bmatrix}\begin{bmatrix}\vec{w}(t)\\ \vec{e}(t)\end{bmatrix}^T\right\}=\begin{bmatrix}\vec{Q} & 0\\ 0 & \vec{R}\end{bmatrix} \tag{5.49}$$

a z toho plyne:

$$Q(t)=E\left[w(t)w(t)^T\right] \tag{5.50}$$

$$R(t)=E\left[e(t)e(t)^T\right] \tag{5.51}$$

Při rozjezdu vozidla v čase t = 0 známe pouze výchozí pozici x_0. Stavový vektor bude mít tvar

$$\vec{x}(0)=\begin{bmatrix}x_0\\ 0\\ 0\end{bmatrix} \tag{5.52}$$

Kovarianční matice chyb bude mít tvar:

$$\vec{P}=\begin{bmatrix}r_{11} & 0 & 0\\ 0 & \lambda_1 & 0\\ 0 & 0 & \lambda_2\end{bmatrix} \tag{5.53}$$

Koeficient r_{11} je chyba souřadnice x. Při rozjezdu nastavíme koeficienty v této matici na velké hodnoty, což značí nevěrohodnost souřadnic. V dalších krocích se koeficienty automaticky upraví. Cílem je predikce posloupnosti odhadu stavů x(t) a odpovídající posloupnosti kovariančních matic chyb odhadu

$$\vec{P}(t)=\vec{E}\left\{\left(x(t)-\hat{x}(t)\right)\left(x(t)-\hat{x}(t)\right)^T\right\} \tag{5.54}$$

Minimalizací podle [36] obdržíme

$$\hat{x}(t \mid t) = \hat{x}(t \mid t-1) + K(t)\left(y(t) - \vec{C}\,\hat{x}(t \mid t-1) \right) \qquad (5.55)$$

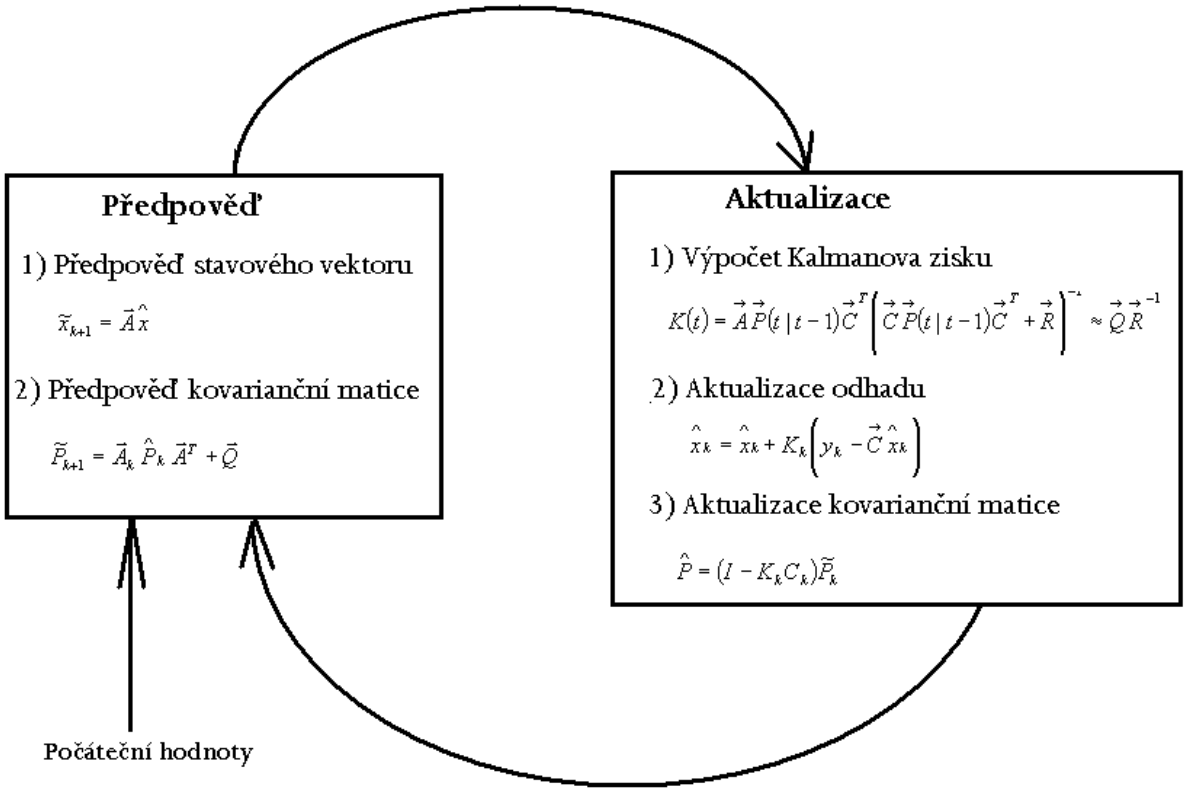
kde K(t) je Kalmanův zisk

$$K(t) = \vec{A}\,\vec{P}(t \mid t-1)\vec{C}^{T}\left(\vec{C}\,\vec{P}(t \mid t-1)\vec{C}^{T} + \vec{R} \right)^{-1} \approx \vec{Q}\,\vec{R}^{-1} \qquad (5.56)$$

Vývoj kovarianční matice je určen Riccatiho rovnicí

$$\vec{P}(t+1 \mid t) = \vec{A}\,\vec{P}(t \mid t-1)\vec{A}^{T} - \vec{K}(t)\left(\vec{C}^{T}\,\vec{P}(t \mid t-1)\vec{C} + \vec{R} \right)\vec{K}(t)^{T} + \vec{Q} \qquad (5.57)$$

Na Obr. 48 je diagram rekurze rozdělený na krok aktualizační a předpovědní (predikční)



Obr. 48 Diagram rekurze

Kalmanův filtr dobře pracuje i v případě, že nejsou k dispozici nová data pro aktualizaci. V tom případě prostě provádí předpovědi podle dat dřívějších. Samozřejmě, že přitom může docházet ke kumulaci chyb, případně vozidlo změní některý parametr jízdy.

6 POPIS VYVINUTÉHO PROGRAMOVÉHO VYBAVENÍ

6.1 Popis knihovny mycv.h

Programy byly sestavovány a odladěny v prostředí překladače C++ Builder od firmy Borland [4], [18]. Většina algoritmů odvozených v této práci je zahrnuta v knihovně mycv.h. Pro další použití této knihovny jsou dále popsány vlastnosti jednotlivých tříd, které knihovna obsahuje. Předpokládá se vytváření lineárního seznamu snímků při prvním průjezdu vozidla. Při opakovaných průjezdech se využijí vestavěné metody pro korelace snímků. Věrohodnostní koeficienty se nastavují podle některého ze zvolených režimů: při režimu vypnuto se automaticky nastaví na 1/9 (jsou tedy všechny stejné), vozidlo silniční nastaví koeficienty středního sloupce na nulu a ostatní na 1/6, železniční nastaví koeficienty polí 1b a 3b na 0,05 a ostatní na 0,9/7.

Primární zdroj souřadnic není nijak řešen, neboť z charakteru úlohy plyne, že např. signál GPS nemusí být k disposici, pokud se bude vozidlo provozovat v nějakém uzavřeném prostoru, jako jsou sklady, podzemní tunely, jeskyně či výhledově i povrch jiných planet. Proto se jako souřadnice ukládají prostě čísla snímků a při opakované jízdě se za souřadnici pokládá právě místo v této řadě. Použití jiného souřadnicového systému pro vstupní údaje je elementární záležitost.

Při první jízdě se vytvoří lineární seznam, který je možno uložit na disk pro pozdější použití. Ukládají se celé objekty, tedy včetně souřadnic, středních hodnot a věrohodnostních koeficientů.

Struktura TBarvaCenter

Obsahuje jen atributy R,G,B a je základním prvkem dalších objektů.

Třída GrafKorelacniFunkce

Je určena pro kreslení grafu hodnot korelačních funkcí.

Třída bitmapobrazek

Je základní třída pro další objekty. Obsahuje veřejně dostupné atributy a metody:

Atributy:

int vyska,sirka rozměry uloženého obrazu

TBarvaCenter **obrcentr vlastní bitová mapa uloženého obrazu

float Rs,Gs,Bs; // prumery barev

TJPEGImage *jp ukazatel na pole, ve kterém se vykreslí ukládaný obraz

Metody:

Konstruktory:

bitmapobrazek(bitmapobrazek *obraz,float m) - kopirovaci konstruktor - lupa

bitmapobrazek(bitmapobrazek *obraz,int odkudx,int kamx,int odkudy,int kamy,int novasirka) - kopirovaci konstruktor - lupa z vyrezu

bitmapobrazek(int vyska, int sirka) - konstruktor prázdného objektu

bitmapobrazek(bitmapobrazek &obraz) - kopírovací konstruktor

bitmapobrazek(bitmapobrazek *obraz,int,int) - kopírovací konstruktor s převodem na novou šířku

bitmapobrazek(char*nazev,TImage *obraz) - konstruktor ze souboru JPG do bitmapobrazek

bitmapobrazek& operator=(bitmapobrazek vstup) - operátor =

malujdo(TImage *obraz) – vykreslení obrazu do pole typu TImage

korelace(bitmapobrazek *B) – vytvoří korelační koeficienty jednotlivých barevných složek s obrazem B

Třída devítka

Slouží k rozdělení vstupního obrazu na 9 dílčích a jejich uložení

Atributy:

bitmapobrazek **devitina - pole devíti obrázků

Metody:

Devitka(bitmapobrazek *vstup); // vytvori 9 bitmapobrazku z bitmapobrazku

Devitka(char*nazev,TImage *obraz); // vytvori 9 bitmapobrazku ze souboru

Devitka(int x, int y); // vytvoří objekt zadaného rozměru bez vložených dat

Devitka(Devitka &d); // kopirovaci konstruktor

Devitka& operator =(Devitka &d) - operator přiřazení

maluj_devitku(TImage *obraz[][3]) - zobrazí všech 9 dílčích obrázků do pole polí

Třída snimek

Atributy:

snimek *dalsi,*predchozi - ukazatelé na následující, resp. předchozí snímek potřebné k vytvoření seznamu snímků.

float x,y - souřadnice

float vaha[3][3] - váhy jednotlivých částí snímku

float vysledek_korelace - proměnná k volnému použití uživatelským programem

Devitka a - vestavěná třída obsahující 9 dílčích částí snímku

Metody:

snimek(bitmapobrazek *_a,float _x,float _y) - konstruktor pro rozdělení a uložení vstupního snímku včetně jeho souřadnic

snimek(int _sirka, int _vyska) - vytvoření prázdného snímku

snimek(snimek &s) - kopírovací konstruktor

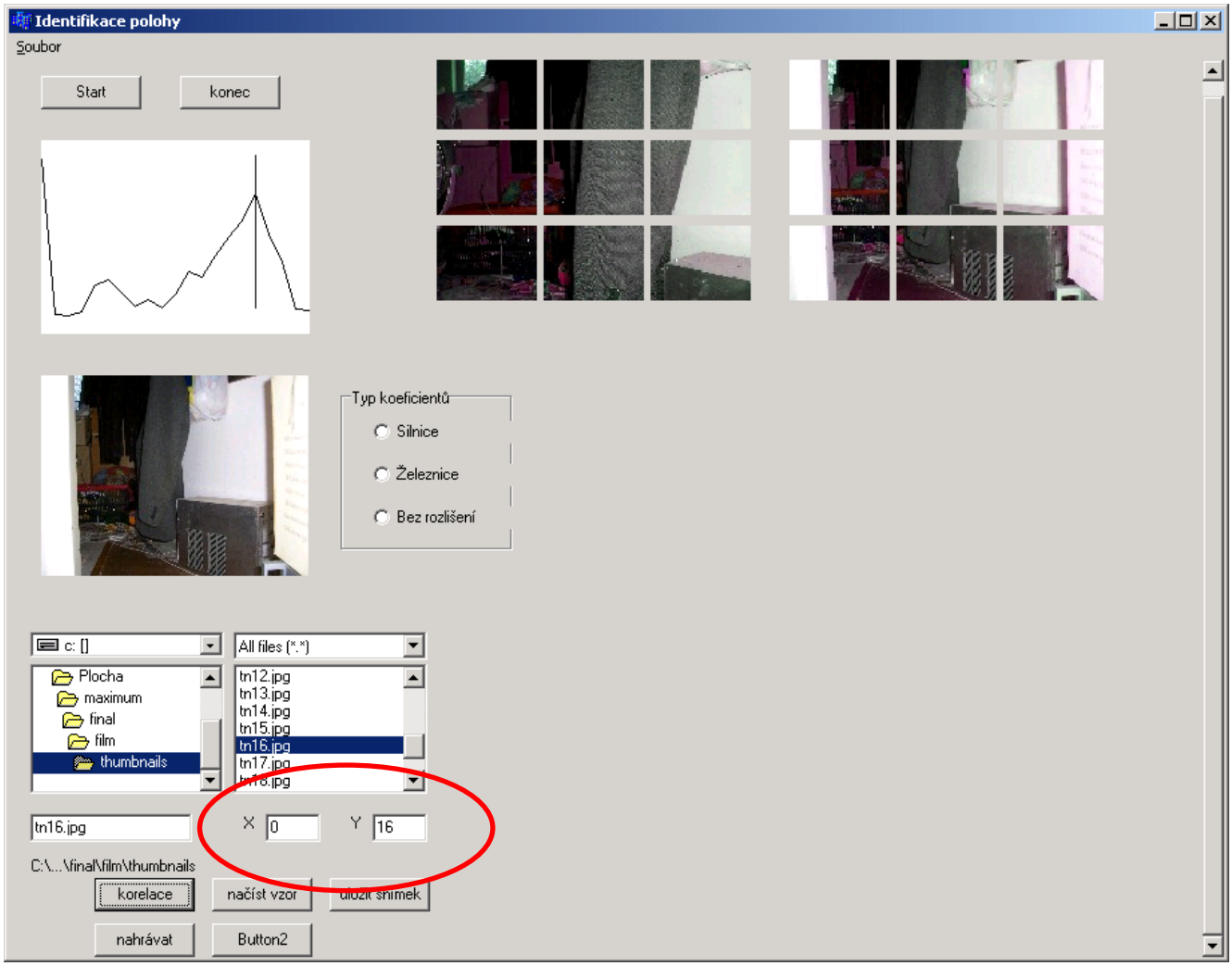
snimek& operator=(snimek &s) - přiřazovací operátor

float korelace(snimek *a) - korelace s jiným snímkem

6.2 Použití knihovny

Prostředky, uvedené v knihovně, byly použity pro sestavení jednoduchého programu, který na základě obrazu z kamery provede:

- Záznam obrázků z první jízdy, doplněný pořadovými čísly, které suplují souřadnice
- Při další jízdě identifikuje svou polohu porovnáním obrázku z kamery porovnáním s obrázky uloženými



Obr. 49 Identifikace polohy

Panel této aplikace zobrazuje i graf znázorňující velikosti korelačních koeficientů jednotlivých uložených snímků a označí vrchol této křivky. Tímto vrcholem a dvěma sousedními proloží parabolu, ze které určí polohu i mezi snímky metodou popsanou v kapitole 5.4. V prostředním obrázku horní řady se průběžně zobrazuje snímek z kamery vozidla. V pravém obrázku horní řady je obrázek, který byl mezi dříve uloženými nalezen jako nejpodobnější. Ostatní prvky panelu aplikace slouží k manipulaci se souborem obrázků.

Toto je jen jedna z možných aplikací vytvořená k demonstraci použití vytvořené knihovny.

7 ZÁVĚR

Zadaná úloha se ukázala mnohem náročnější, než se původně předpokládalo. Této problematice se systematicky patrně nikdo nevěnuje a proto se nepodařilo nalézt potřebnou podporu. Naděje, vkládané do knihovny programů openCV se nenaplnily. Z toho důvodu bylo nutno pro splnění původního cíle řešit řadu dalších problémů. To však dalo vzniknout novým algoritmům, které dosud nebyly řešeny.

Bylo požadováno, aby řešený systém dokázal na základě aktuálního obrazu z kamery a obrázků uložených v předchozí jízdě určit souřadnice místa, kde se právě vozidlo s kamerou nalézá. Odvodit příslušné algoritmy, pomocí nichž je možno získat souřadnice vozidla porovnáním obrazu z optické kamery a obrazy se souřadnicemi dříve uloženými.

Cíle disertace

- Najít vhodnou metodu pro určení posice vozidla opakovaně se pohybujícího po určité trase pomocí optické kamery
- Odvodit příslušné algoritmy, pomocí nichž je možno získat souřadnice vozidla porovnáním obrazu z optické kamery a obrazy se souřadnicemi dříve uloženými.
- Testovat odvozené algoritmy jako důkaz jejich správnosti
- Vytvořené algoritmy realizovat v jazyce C++ ve formě knihovny a zpřístupnit ji.

byly splněny.

Algoritmy, odvozené v této práci byly zprovozněny na malém modelu vozidla (realizovaného z dětské stavebnice) s webovou kamerou (Herkules de Luxe). Vzhledem k velkým nárokům na procesor zpracovávající bitové mapy se vozidlo může pohybovat jen pomalu, aby počítač dokázal včas zpracovat obrazy z kamery. Zvýšení rychlosti je však záležitostí už jen síly procesorového vybavení, buď řádovým zvýšením jeho rychlosti, nebo paralelním chodem více procesorů, což je možné, na což je v této práci na příslušných místech poukazováno.

Algoritmy byly naprogramovány v jazyce C++ vývojového prostředí C++ Builder. Nutno upozornit, že tento překladač pracuje velmi efektivně a ruční přepsání programů např. do jazyka Assembler nepřinese významnější zrychlení výsledných kódů.

Z uvedeného plyne, že zadání se podařilo beze zbytku splnit a jako vedlejší produkt vznikly dříve nepublikované teorie a algoritmy.

PŘEHLED PRACÍ AUTORA

[1] Šroll,J,:*Digitální zpracování rádiových signálů*, Radiožurnál 3/04 2005:

[2] Šroll,J,:*Hybridní pohony osobních vozidel*, příspěvek na konferenci Dopravní systémy 05 2006:

[3] Šroll, J., *Orientace vozidla pomocí vestavěné optické kamery*. Písemná zpráva ke státní doktorské zkoušce, obhájená 16.6.2006

[4] Šroll,J,:*Orientace vozidla pomocí vestavěné optické kamery*, příspěvek na konferenci Teorie dopravních systémů 2007

[5] Sroll, J.; Schejbal, V.; *Orientation Vehicles using embedded optical kamera* Radioelektronika, 2009. RADIOELEKTRONIKA '09. 19th International Konference 22-23 April 2009 Page(s):131 – 134

[6] ŠROLL, J., SCHEJBAL, V. *Zdravotní rizika radarů* Sdělovací technika 2/2009; ISSN 0036-9942 Page(s) 9-13

POUŽITÁ LITERATURA A JINÉ PRAMENY

[1] KEITH, A. Redmill and UMIT, Ozguner. *The Ohio State University Automated Highway System Demonstration Vehikle.* SAE Internation Congress and Exposition, February 1998, SAE Paper 980855.

[2] MURAKAMI, N.-INOUE, K.Otsuka. *Selective harvesting robot of cabbage*. Proceedings of international symposium of automation and robotics in bioproduction and processing, JSAM, Vol.2, 24-31,1995

[3] *Image Compression Using Discrete Cosine Transform.* Sriharsha Maganti Villanova University 2004

[4] VIRIUS, M. *C++ Builder 4.0 Podrobný průvodce.* Praha:Grada, 1999. 264 s. ISBN 80-7169-796-6.

[5] Verilog HDL: *Discrete Cosine Transform*. last revision 9th of January 2004 http://www.altera.com/support/examples/verilog/ver_dct.html

[6] KUHN, J. Conventional Analog Television - An Introduction internet. http://www.ee.washington.edu/conselec/CE/kuhn/ntsc

[7] VIKIPEDIA. *Inertial Navigation System* last revision 11 November 2007 http://en.wikipedia.org/wiki/Inertial_guidance

[8] SONKA,Milan. - Hlavac, Václav. *Image Processing, Analysis, and Machine Vision.* Roger Boyle, PWS Boston 1999, 2nd edition; ISBN 0-534-95393-X

[9] NAVAJO *HSV barevný prostor*. 1999. http://hsv-barevny-prostor.navajo.cz.

[10] LYNXMOTOTION. *4WD1 Robot Combo Kit for Autonomous Behavior*. http://www.lynxmotion.com/Product.aspx?productID=373.

[11] IBM CVUT Students research projects 2007 *Non-speech Control of BlueCar*. http://ibm-cvut.felk.cvut.cz/~nscbc07/project2/.

[12] SOURCEFORGE. *SourceForge.net: Open Computer Vision Library* 1999-2008. http://sourceforge.net/projects/opencvlibrary/.

[13] ADOLF, Florian. OpenCV Object Detection Howto. Last revision 2008-05-28 http://opencvlibrary.sourceforge.net/ObjectDetection.

[14] Intel Corporation. *Opensource Computer Vision Library*, 2001.

[15] PRATT W. *Digital image processing* (3ed., Wiley, 2001. 738 s. ISBN 0471374075.

[16] KEITH A. Redmill – OZGUNER, Umit . *Automated Highway System Demonstration Vehicle*, The Ohio State University.1998. SAE Internation Congress and Exposition, February 1998, SAE Paper 980855.

[17] MURAKAMI, N. – INOUE, K. – OTSUKA, K. *Selective harvesting robot of cabbage*. Proceedings of international symposium of automation and robotics in bioproduction and processing, JSAM, Vol.2, 24-31,1995

[18] MATOUŠEK, David. *C++ Builder.* 1.díl 3.vydání. Praha:Ben. 2002.688 s. ISBN 80-7300-064-4

[19] ZHIGANG Zhu. - GUANGGYOU, Xu, BO Yang. – DINJI, Shi. – XUEIN, Lin. *VISATRAM: a real-time vision system for automatic traffic monitoring*. Image and Vision Computing 18 (781 - 794)

[20] ESCALERA, A. - ARMINGOL, J. M. – MATA, M. *Traffic sign recognition and analysis for intelligent vehicles* Image and Vision Computing 21 (247-258)

[21] CURIO, C. – EDELBUNNER, J.- KALINKE, T. – TZOMAKAS, C. – SEELEN, W. *Walking Pedestrian Recognition*. IEEE transactions on Intelligent transportation Systems. September 2000.

[22] STILLER, C. – HIPP, J. – ROSSING, C. – EWALD, A. *Multisensor obstacle detection and tracking*. Image and Vision Computing 18 (389 – 396)

[23] KAMIJO, S. – MATSUSHITA, Y. – IKEUCHI, K. – SAKAUCHI, M. *Traffic Monitoring and Accident Detection at Intersections*. IEEE transactions on Intelligent transportation Systems. June 2000.

[24] HECKS, Karl. Bombing *1939-1945: the air offensive against land targets in World War Two*. Robert Hale Ltd., London. 1990. ISBN 0-7090-4020-2.

[25] Wallace, G.K. *The JPEG Still Picture Compression Standard*. Communications of the ACM, April, 1991.

[26] Assumption University of Thailand. *Solaris XIL 1.1 Imaging Library Programmer's Guide*, Intercampus Code: 0000214683. November 1993

[27] THOMPSON, C. - SHURE L. *The MATLAB Imaging Processing Toolbox Manual*, 1993, Mathworks, Inc.

[28] NELSON, Mark. *The Data Compression Book*, 1992, M and T Publishing, Inc., New York, NY

[29] Nevřiva P.: *Analýza signálů a soustav*, BEN Praha 2000, ISBN 80-7300-004-0

[30] Thompson C. and Shure L., *The MATLAB Imaging Processing Toolbox Manual*, 1993

[31] Rak R., Matyáš V., Říha Z., *Biometrie a identita člověka ve forenzních a komerčních aplikacích*, GRADA 2008, ISBN 978-80-247-2365-5

[32] Smith, Gregory H.; Hagerott, Edward C.; Scherr, Lawrence M.; Herkenhoff, Kenneth E.; Bell, James F. *Optical designs for the Mars '03 rover cameras* http://www.mwoa.org/SPIE_paper.pdf

[33] Keith A. Redmill and Umit Ozguner, *The Ohio State University Automated Highway System Demonstration Vehicle*, 1998 SAE Internation Congress and Exposition**,** February 1998, SAE Paper 980855.

[34] Murakami,N., K.Inoue, K.Otsuka. *Selective harvesting robot of cabbage*. Proceedings of international symposium of automation and robotics in bioproduction and processing, JSAM, Vol.2, 24-31,1995

[35] Havlena, V.; Štecha, J. *Moderní teorie řízení*. Praha: Vydavatelství ČVUT, 2000.

[36] Havlena, V. *Moderní teorie řízení – doplňkové skriptum.* Praha: Vydavatelství ČVUT, 1999. ISBN 80-01-02036-3

8 PŘÍLOHY

8.1 Knihovní soubor

```
#include <vcl.h>
#include <Classes.hpp>
#include <ComCtrls.hpp>
#include <Controls.hpp>
#include <Dialogs.hpp>
#include <ExtCtrls.hpp>
#include <FileCtrl.hpp>
#include <Forms.hpp>
#include <StdCtrls.hpp>
#include <CheckLst.hpp>
#include <Menus.hpp>

//---------------------------------------------------------------------------

#ifndef __mycv__
#define __mycv__
//---------------------------------------------------------------------------
#include <ExtDlgs.hpp>
#include <Graphics.hpp>
#include <jpeg.hpp>
#include <Menus.hpp>
#include <stdio.h>
#include <dblookup.hpp>



//---------------------------------------------------------------------------
typedef struct {
long int R,G,B;
}TBarvaCenter;

//---------------------------------------------------------------------------
class GrafKorelacniFunkce{
public:
int pocet;
TBarvaCenter *hodnota;
 GrafKorelacniFunkce(int _pocet): pocet(_pocet)    // konstruktor
 {hodnota=(TBarvaCenter*)malloc(pocet*sizeof(TBarvaCenter));
```

```
 }
 pridejhodnotu(int i,TBarvaCenter *dalsi)
 {  hodnota[i]=*dalsi;
 }
 ~ GrafKorelacniFunkce(){free(hodnota);}; // destruktor
 malujgraf(TImage *obraz,int vyska,int sirka,float rychlx,float rychly)
 { int i;
  for(i=0;i<pocet-1;i++)
  { obraz->Canvas->MoveTo(i*rychlx+10,vyska-hodnota[i].R/rychly);
    obraz->Canvas->Pen->Color=clRed;
    obraz->Canvas->LineTo(rychlx*(i+1)+10,vyska-hodnota[i+1].R/rychly);

    obraz->Canvas->MoveTo(i*rychlx+10,vyska-hodnota[i].G/rychly);
    obraz->Canvas->Pen->Color=clGreen;
    obraz->Canvas->LineTo(rychlx*(i+1)+10,vyska-hodnota[i+1].G/rychly);

    obraz->Canvas->MoveTo(i*rychlx+10,vyska-hodnota[i].B/rychly);
    obraz->Canvas->Pen->Color=clBlue;
    obraz->Canvas->LineTo(rychlx*(i+1)+10,vyska-hodnota[i+1].B/rychly);
  }
  }
};

//---------------------------------------------------------------------------


class bitmapobrazek{
private:
int i,j,k;
TColor barva;
TBarvaCenter bod;
public:
void alokujpamet(void);
  int vyska,sirka;
  int **obrsav;
  TBarvaCenter **obrcentr;
  float Rs,Gs,Bs;  // prumery barev
  TJPEGImage *jp;
  Graphics::TBitmap *Bitmap;

TBarvaCenter korelace(bitmapobrazek *B);
```

```
/* kopirovaci konstruktor - lupa */
bitmapobrazek(bitmapobrazek *obraz,float m);

/* kopirovaci konstruktor - lupa z vyrezu */
bitmapobrazek
(bitmapobrazek *obraz,int odkudx,int kamx,int odkudy,int kamy,int novasirka);

/* konstruktor prazdneho objektu */
bitmapobrazek(int vyska, int sirka);

/* kopirovaci konstruktor */
bitmapobrazek(bitmapobrazek &obraz);

/* kopirovaci konstruktor s prevodem na novou sirku*/
bitmapobrazek(bitmapobrazek *obraz,int,int);

/* konstruktor ze souboru JPG do bitmapobrazek */
bitmapobrazek(char*nazev,TImage *obraz);

bitmapobrazek& operator=(bitmapobrazek vstup);

void malujdo(TImage *obraz);
~bitmapobrazek();  // destruktor
TBarvaCenter kontrolacentru(void); // kontrola vycentrovani barev
};

//---------------------------------------------------------------------------

TBarvaCenter bitmapobrazek::korelace(bitmapobrazek *B)
{
TBarvaCenter vysledek;
 register int i,j,k;
 long r,g,b,plocha;
 plocha=(vyska*sirka);
 r=g=b=0;
 for(i=0;i<vyska;i++)
 for(j=0;j<sirka;j++)
 { k=B->vyska;
  r+=obrcentr[i][j].R*B->obrcentr[i][j].R;
  g+=obrcentr[i][j].G*B->obrcentr[i][j].G;
  b+=obrcentr[i][j].B*B->obrcentr[i][j].B;
 }
```

```
vysledek.R=r/plocha;
vysledek.G=g/plocha;
vysledek.B=b/plocha;
return vysledek;
}
//---------------------------------------------------------------------------

void bitmapobrazek::alokujpamet(void)
{ int i;
  obrsav=(int**)malloc(vyska*sizeof(int*));
  obrcentr=(TBarvaCenter**)malloc(vyska*sizeof(TBarvaCenter*));
  for(i=0;i<vyska;i++)
  {obrsav[i]=(int*)malloc(sirka*sizeof(int));
   obrcentr[i]=(TBarvaCenter*)malloc(sirka*sizeof(TBarvaCenter));
  }
}

//---------------------------------------------------------------------------
/* konstruktor prazdneho objektu */
bitmapobrazek::bitmapobrazek(int _vyska, int _sirka):vyska(_vyska), sirka(_sirka)
{ alokujpamet();
}

//---------------------------------------------------------------------------
/* kopirovaci konstruktor - lupa */
bitmapobrazek::bitmapobrazek(bitmapobrazek *obraz,float m){
  int x2,y2,plocha;
  TBarvaCenter bod,k;
sirka=obraz->sirka;
vyska=obraz->vyska;
alokujpamet();
x2=sirka/2;
y2=vyska/2;
    Rs=Gs=Bs=0;
for(i=0;i<vyska;i++)
for(j=0;j<sirka;j++)
{obrsav[i][j]=obraz->obrsav[(int)(y2+(i-y2)*m)][(int)(x2+(j-x2)*m)];
bod=obrcentr[i][j]=obraz->obrcentr[(int)(y2+(i-y2)*m)][(int)(x2+(j-x2)*m)];
Rs+=bod.R;
Bs+=bod.B;
Gs+=bod.G;
}  plocha=(vyska)*(sirka);
```

```
  Rs=Rs/plocha;
  Gs=Gs/plocha;
  Bs=Bs/plocha;
  for(i=0;i<vyska;i++)
  for(j=0;j<sirka;j++)
 {
 k=obrcentr[i][j];
 bod.G=k.G -(int)Gs;
 bod.B=k.B-(int)Bs;
 bod.R=k.R-(int)Rs;
 obrcentr[i][j]=bod;
 }
Rs+=obraz->Rs;
Gs+=obraz->Gs;
Bs+=obraz->Bs;
} // of  kopirovaci konstruktor - lupa */


//---------------------------------------------------------------------------
/* kopirovaci konstruktor - lupa z vyrezu */
bitmapobrazek::bitmapobrazek
(bitmapobrazek *obraz, int odkudx,int kamx,int odkudy,int kamy,int novasirka)
{ int plocha,i,j;
  float m;
  TBarvaCenter bod,k;
  m=novasirka/((float)(kamx-odkudx));
  sirka=novasirka;
  vyska=(kamy-odkudy)*m;
  alokujpamet();
  Rs=Gs=Bs=0;
for(i=0;i<vyska;i++)
for(j=0;j<sirka;j++)
 {obrsav[i][j]=obraz->obrsav[(int)(odkudy+i/m)][(int)(odkudx+j/m)];
  bod=obrcentr[i][j]=obraz->obrcentr[(int)(odkudy+i/m)][(int)(odkudx+j/m)];
  Rs+=bod.R;
  Bs+=bod.B;
  Gs+=bod.G;
 }
   plocha=(vyska)*(sirka);
   Rs=Rs/plocha;
   Gs=Gs/plocha;
   Bs=Bs/plocha;
```

```
  for(i=0;i<vyska;i++)
  for(j=0;j<sirka;j++)
 {
 k=obrcentr[i][j];
 bod.G=k.G -(int)Gs;
 bod.B=k.B-(int)Bs;
 bod.R=k.R-(int)Rs;
 obrcentr[i][j]=bod;
 }
Rs+=obraz->Rs;
Gs+=obraz->Gs;
Bs+=obraz->Bs;
}

//---------------------------------------------------------------------------
/* kopirovaci konstruktor */
bitmapobrazek::bitmapobrazek(bitmapobrazek &obraz){
sirka=obraz.sirka;
vyska=obraz.vyska;
Rs=obraz.Rs;
Gs=obraz.Gs;
Bs=obraz.Bs;
alokujpamet();
for(i=0;i<vyska;i++)
for(j=0;j<sirka;j++)
 {obrsav[i][j]=obraz.obrsav[i][j];
  obrcentr[i][j]=obraz.obrcentr[i][j];
 }
}

//---------------------------------------------------------------------------
/* kopirovaci konstruktor s prevodem na novou sirku*/
bitmapobrazek::bitmapobrazek(bitmapobrazek *obraz,int nsirka,int nvyska){
int osirka,ovyska;
osirka=obraz->sirka;
ovyska=obraz->vyska;
sirka=nsirka;
vyska=nvyska;
Rs=obraz->Rs;
Gs=obraz->Gs;
Bs=obraz->Bs;
alokujpamet();
```

```
for(i=0;i<vyska;i++)
for(j=0;j<sirka;j++)
 {obrsav[i][j]=obraz->obrsav[i*ovyska/nvyska][j*osirka/nsirka];
  obrcentr[i][j]=obraz->obrcentr[i*ovyska/nvyska][j*osirka/nsirka];
 }
}

//---------------------------------------------------------------------------
/* konstruktor nacitajici obrazek ze souboru */
bitmapobrazek::bitmapobrazek(char*nazev,TImage *obraz)
{
 Bitmap = new Graphics::TBitmap();
 TJPEGImage *jp = new TJPEGImage();
 long lRs,lGs,lBs;
 obraz->Picture->LoadFromFile(nazev);
 Bitmap->Assign(obraz->Picture->Graphic);
 vyska=Bitmap->Height;
 sirka =Bitmap->Width;
 obraz->Width=sirka;
 obraz->Height=vyska;
 alokujpamet();
 lRs=lGs=lBs=0;
 for(i=0;i<vyska;i++)
 for(j=0;j<sirka;j++)
 { barva=Bitmap->Canvas->Pixels[j][i];
   obrsav[i][j]=barva;
   lBs+=((barva&0xff0000)>>16);
   lGs+=((barva&0xff00)>>8);
   lRs+=(barva&0xff);
 }
   Rs=lRs/((float)((vyska)*(sirka)));
   Gs=lGs/((float)((vyska)*(sirka)));
   Bs=lBs/((float)((vyska)*(sirka)));
   for(i=0;i<vyska;i++)
   for(j=0;j<sirka;j++)
 {k=obrsav[i][j];
  bod.B=((k&0xff0000)>>16)-Bs;
  bod.G=((k&0xff00)>>8)-Gs;
  bod.R=(k&0xff)-Rs;
  obrcentr[i][j]=bod;
 }
} // of konstruktor   bitmapobrazek(char*nazev,TImage *obraz)
```

```
//---------------------------------------------------------------------------
void bitmapobrazek::malujdo(TImage *obraz){
 obraz->Width=sirka;
 obraz->Height=vyska;
  for(i=0;i<vyska;i++)
  for(j=0;j<sirka;j++)
 {bod=obrcentr[i][j];
  k=((int)(bod.B+Bs)<<8)+((int)(bod.G+Gs)<<16)+(int)(bod.R+Rs);
  obraz->Canvas->Pixels[j][i]=(TColor)k;
 }
} //of malujdo(TImage obraz)


//---------------------------------------------------------------------------
   TBarvaCenter bitmapobrazek::kontrolacentru(void)
  {  TBarvaCenter bod,vysl;
    int i,j;
    long R,G,B;
    vysl.R=vysl.G=vysl.B=0;
    for(i=0;i<vyska;i++)
    for(j=0;j<sirka;j++)
    {bod=obrcentr[i][j];
     vysl.R+=bod.R;
     vysl.G+=bod.G;
     vysl.B+=bod.B;
     R+=bod.R;
     G+=bod.G;
     B+=bod.B;
    }
 return vysl;
 }
//---------------------------------------------------------------------------
 bitmapobrazek& bitmapobrazek::operator =(bitmapobrazek vstup){
  int i,j;
  vyska=vstup.vyska;
  sirka=vstup.sirka;
  Rs=vstup.Rs;
  Gs=vstup.Gs;
  Bs=vstup.Bs;
  for(i=0;i<vyska;i++)
```

```
  for(j=0;j<sirka;j++)
  obrcentr[i][j]=vstup.obrcentr[i][j];
  return *this;
}


//---------------------------------------------------------------------------
bitmapobrazek::~bitmapobrazek()   // destruktor
{
  int i;
   for(i=0;i<vyska;i++)
  {free(obrsav[i]);
   free(obrcentr[i]);
  }
   free(obrsav);
   free(obrcentr);
   obrsav=0;
   obrcentr=0;
}

//---------------------------------------------------------------------------
class Devitka{
int i,j,k,l,sirka,vyska;
void rozdelobraz(bitmapobrazek *vstup); // vytvori 9 bitmapobrazku z bitmapobrazku
bitmapobrazek** namapuj(int sirka, int vyska);
public:
bitmapobrazek **devitina;
Devitka(bitmapobrazek *vstup); // vytvori 9 bitmapobrazku z bitmapobrazku
//bitmapobrazek::bitmapobrazek(char*nazev,TImage *obraz)
Devitka(char*nazev,TImage *obraz); // vytvori 9 bitmapobrazku ze souboru
Devitka(int x, int y); // prazdna devitka
Devitka(Devitka &d); // kopirovaci konstruktor
// bitmapobrazek& bitmapobrazek::operator =(bitmapobrazek vstup)

Devitka& operator =(Devitka &d);
void maluj_devitku(TImage *obraz[][3]);
~Devitka(); // destruktor
};

//---------------------------------------------------------------------------

Devitka&  Devitka::operator =(Devitka &d)
```

```
{
sirka=d.sirka;
vyska=d.vyska;
 for(i=0;i<3;i++)
 for(j=0;j<3;j++)
 devitina[i][j]=d.devitina[i][j];
}

//---------------------------------------------------------------------------
bitmapobrazek** Devitka::namapuj(int sirka, int vyska)
{
bitmapobrazek **devitina;
devitina=(bitmapobrazek**)malloc(3*sizeof(bitmapobrazek*));
for(i=0;i<3;i++)
 {devitina[i]=(bitmapobrazek*)malloc(3*sizeof(bitmapobrazek));
  {for(j=0;j<3;j++)
   {
 devitina[i][j].vyska=vyska;
 devitina[i][j].sirka=sirka;
 devitina[i][j].alokujpamet();
   }
  }
 }
 return devitina;
}

//---------------------------------------------------------------------------

Devitka::Devitka(Devitka &d) // kopirovaci konstruktor
{
sirka=d.sirka;
vyska=d.vyska;

 devitina=namapuj(sirka,vyska);
 for(i=0;i<3;i++)
 for(j=0;j<3;j++)
 devitina[i][j]=d.devitina[i][j];
}
//---------------------------------------------------------------------------
void Devitka::maluj_devitku(TImage *obraz[][3])
{
for(int j=0;j<3;j++)
```

```
for(int k=0;k<3;k++)
{
devitina[j][k].malujdo((obraz[j][k]));
}
}
//---------------------------------------------------------------------------
void Devitka::rozdelobraz(bitmapobrazek *vstup)
{
int i,j,k,T=75;
vyska=T*0.75;
sirka=T;

devitina=namapuj(sirka,vyska);

int sirkaold=vstup->sirka/3, vyskaold=vstup->vyska/3;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
 {
 int x1,x2,y1,y2,ns;
 x1=sirkaold*j;
 x2=sirkaold*(j+1);
 y1=vyskaold*i;
 y2=vyskaold*(i+1);
 devitina[i][j]=bitmapobrazek(vstup,x1,x2,y1,y2,T);
 }
}
//---------------------------------------------------------------------------
Devitka::Devitka(bitmapobrazek *vstup)
{rozdelobraz(vstup);
}

//---------------------------------------------------------------------------
Devitka::Devitka(int _vyska, int _sirka):vyska(_vyska),sirka(_sirka)
{
 devitina=namapuj(sirka,vyska);
}

//---------------------------------------------------------------------------
Devitka::Devitka(char*nazev,TImage *obraz) // vytvori 9 bitmapobrazku ze souboru
{bitmapobrazek vstup(nazev,obraz);
 rozdelobraz(&vstup);
};
```

```
//---------------------------------------------------------------------------
Devitka::~Devitka(){
for(i=0;i<3;i++)
 free(devitina[i]);
 free(devitina); devitina=0;
}

//---------------------------------------------------------------------------

class snimek{
int i,j;
public:
snimek *dalsi,*predchozi;
float x,y;
float vaha[3][3];
float vysledek_korelace;
Devitka a;
snimek(bitmapobrazek *_a,float _x,float _y);
snimek(int _sirka, int _vyska);
snimek(snimek &s);

snimek& operator=(snimek &s);

float korelace(snimek *a);

};
//---------------------------------------------------------------------------
snimek::snimek(bitmapobrazek *_a,float _x,float _y):a(_a),x(_x),y(_y){
for(i=0;i<3;i++)
for(j=0;j<3;j++)
vaha[i][j]=1.0/9.0;
dalsi=NULL;
};

//---------------------------------------------------------------------------
float snimek::korelace(snimek *b)
{int i,j,k;
TBarvaCenter v1;
//float vysledek;
 vysledek_korelace=0;
for(j=0;j<3;j++)
```

```
for(k=0;k<3;k++)
{v1=a.devitina[j][k].korelace(&(b->a.devitina[j][k]));
 vysledek_korelace+=vaha[j][k]*sqrt(v1.R*v1.R+v1.G*v1.G+v1.B*v1.B);
}
return vysledek_korelace;
};

//---------------------------------------------------------------------------
snimek::snimek(int _sirka, int _vyska):a(_sirka,_vyska){
predchozi=dalsi=NULL;
}

//---------------------------------------------------------------------------
snimek& snimek::operator=(snimek &s)
{
a=s.a;
dalsi=s.dalsi;
predchozi=s.predchozi;
x=s.x;y=s.y;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
vaha[i][j]=s.vaha[i][j];
vysledek_korelace=s.vysledek_korelace;
}
//---------------------------------------------------------------------------
snimek::snimek(snimek &s):a(s.a)
{
//a=s.a;
dalsi=s.dalsi;
predchozi=s.predchozi;
x=s.x;y=s.y;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
vaha[i][j]=s.vaha[i][j];
vysledek_korelace=s.vysledek_korelace;
}
//---------------------------------------------------------------------------



//---------------------------------------------------------------------------
class Tdeleniobrazu : public TForm
```

```
{
__published:       // IDE-managed Components
        TButton *Button1;
        TButton *konec;
        TImage *obraz;
        TImage *dil1;
        TImage *dil2;
        TImage *dil3;
        TImage *dil4;
        TImage *dil5;
        TImage *dil6;
        TImage *dil7;
        TImage *dil8;
        TImage *dil9;
        TImage *k1;
        TImage *k4;
        TImage *k2;
        TImage *k5;
        TImage *k3;
        TImage *k6;
        TImage *k9;
        TImage *k8;
        TImage *k7;
        TLabel *l1;
        TLabel *l2;
        TLabel *l3;
        TLabel *l4;
        TImage *porovn;
        TLabel *l5;
        TImage *n1;
        TImage *n2;
        TImage *n3;
        TImage *n4;
        TImage *n5;
        TImage *n6;
        TImage *n7;
        TImage *n8;
        TImage *n9;
        TLabel *l6;
        TImage *smazat;
        TImage *smazat2;
        TFileListBox *FileListBox1;
```

```
        TFilterComboBox *FilterComboBox1;
        TDirectoryListBox *DirectoryListBox1;
        TEdit *Edit1;
        TLabel *cesta;
        TDriveComboBox *DriveComboBox1;
        TButton *pridej_obrazek;
        TEdit *sourx;
        TEdit *soury;
        TLabel *Label2;
        TLabel *Label3;
        TMainMenu *MainMenu1;
        TMenuItem *Soubor1;
        TMenuItem *Nacti;
        TMenuItem *Ulozit;
        TButton *nactivzor;
        TButton *korel9;
        TButton *Button2;
        TButton *nahrej;
        TTimer *Timer1;
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall konecClick(TObject *Sender);
        void __fastcall novyadr(TObject *Sender);
        void __fastcall pridej_obrazekClick(TObject *Sender);
        void __fastcall ukazobrazek(TObject *Sender);
        void __fastcall UlozitClick(TObject *Sender);
        void __fastcall NactiClick(TObject *Sender);
        void __fastcall nactivzorClick(TObject *Sender);
        void __fastcall korel9Click(TObject *Sender);
        void __fastcall nahrejClick(TObject *Sender);
        void __fastcall Timer1Timer(TObject *Sender);
private:    // User declarations
        FILE *vstup;

public:             // User declarations
        __fastcall Tdeleniobrazu(TComponent* Owner);
        __fastcall ~Tdeleniobrazu();
        TBitmapImage *bmi;
        bool bezi_nahravani;

        void osygrafu(TImage*,int,int,int,int);
        void dosouboru(TImage *obr, char* jmeno);
```

```
        void viditelnost_nadpisu(bool);

          TImage *korely[3][3];
          TImage *vzory[3][3];
          TImage *dily[3][3];
          snimek *vzor_z_cesty;

    };


    //---------------------------------------------------------------------------
    extern PACKAGE Tdeleniobrazu *deleniobrazu;
    //---------------------------------------------------------------------------
#endif
```

8.2 Vyhodnocení snímků

8.2.1 Vyhodnocení snímků ze silnice

Snímky byly pořízeny kamerou Kodak CX7525. Snímky, které vykazovaly zjevné nedostatky (rozmazané, zacloněné jedoucím vozidlem atd.) byly ze souboru vyřazeny. Tabulka patří ke kapitole 5.3. Koeficient 1 znamená, že příslušná devítina obrázku poskytla věrohodný průběh korelačního koeficientu, 0 znamená, že průběh korelačního koeficientu nevykazoval maximum v místě, kde by se ono maximum mělo vyskytovat.

Tab. 3 Vyhodnocení snímků ze silnice (na přiloženém disku)

8.2.2 Železnice

Snímky byly pořízeny kamerou Kodak CX7525. Protože nebylo možno umístit kameru na lokomotivu, byly snímky pořízeny ve stoje na železniční trati a další oknem posledního vagonu z jedoucího vlaku. Snímky, které vykazovaly zjevné nedostatky (rozmazané, zacloněné jedoucím vozidlem atd.) byly ze souboru vyřazeny.

Tab. 4 Vyhodnocení snímků železnice (na přiloženém disku)

8.2.3 Volná krajina

Při jízdě volnou krajinou, obecně prostorem, kde se nevyskytuje jiná další doprava, není třeba řešit problémy s částmi obrazu zatíženými dopravní situací. Proto není třeba nastavovat koeficienty věrohodnosti na navzájem různé hodnoty, postačí jejich nastaveni na 1/9 tak, aby

$$\sum_{i=1}^{9} k_i = 1 \qquad (8.1)$$

kde k_i jsou váhy (či koeficienty věrohodnosti) jednotlivých částí obrazu. Proto tento případ nebyl dále zkoumán.