Využití knihovny OpenCV pro zpracování obrazu v mobilních robotických systémech

The Use of OpenCV Library for Image Processing in Mobile Robotic Systems

Ondřej Lapáček

Bakalářská práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej LAPÁČEK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Využití knihovny OpenCV pro zpracování obrazu v mobilních robotických systémech.**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na téma zpracování obrazu v mobilních robotických systémech.**
2. **Analyzujte možnosti využití knihovny OpenCV pro zpracování obrazu v mobilních robotických zařízeních.**
3. **Vypracujte návrh ukázkové aplikace knihovny OpenCV.**
4. **Demonstrujte řešení s využitím jakyka C resp. C++.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **FORSYTH, David. – PONCE, Jean. Computer Vision: A Modern Approach. Prentice Hall, 2003. 693 s. ISBN 0130851981.**
2. **SNYDER, WESLEY, E. QI, Hairong. Machine Vision. Cambridge University Press, 2004. 434 s. ISBN 052183046X.**
3. **DUDEK, Gregory. RICHARD, Michael. JENKIN, MacLean. JENKIN, Michael. Computational Principles of Mobile Robotic. Cambridge University Press, 2000. 280 s. ISBN 0521568765.**

Vedoucí bakalářské práce: **Ing. Erik Král**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Vzrůstající výpočetní kapacita počítačů umožňuje rychlejší a lepší zpracovávání reálného obrazu. Tohoto faktu využívají i knihovny počítačového vidění, mezi které patří i knihovna OpenCV. Cílem práce je seznámit čtenáře s možností využití zpracování obrazu v mobilních robotických systémech a s tím spojenou problematikou. V práci jsou uvedeny nejpoužívanější knihovny počítačového vidění a jejich srovnání. Pro knihovnu OpenCV, která umožňuje přenositelnost mezi různými zařízeními a operačními systémy, je také zkonstruován ukázkový program v programovacím jazyku C++.

Klíčová slova: Knihovna počítačového vidění, OpenCV, mobilní robotické systémy.

ABSTRACT

Increasing computational capacity of computers allows for faster and better processing of real images. This fact is used by the computer vision libraries, including library OpenCV. The aim of the project is to acquaint the reader with the possibility of the use of image processing in mobile robotic systems and the associated problems. There are listed the most common computer vision libraries and their comparison. The OpenCV library allows for portability among different devices and operating systems. There is also designed sample program in the programming language C++.

Keywords: The Computer Vision Library, OpenCV, Mobile Robotic Systems.

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. et Ing. Eriku Královi za odbornou pomoc při zpracovávání a řešení problémů spojných s psaním teorie počítačového vidění a s konstrukcí ukázkového programu.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému, dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci, nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….…….

podpis diplomanta

OBSAH

ÚVOD

Vzrůstající výpočetní kapacita počítačů umožňuje rychlejší a lepší zpracovávání reálného obrazu. Tohoto faktu využívají i knihovny počítačového vidění, mezi které patří i knihovna OpenCV, jejímž autorem je společnost Intel. Dále se tato knihovna rozvijí pod licencí BSD. Jako jeden z příkladů úspěšné implementace knihovny OpenCV lze uvést závody autonomních vozidel DARPA v roce 2005, které vyhrálo autonomní vozidlo Stanly, v němž byla tato knihovna použita ke zpracování reálného obrazu.

Cílem práce je seznámit čtenáře s možností využití zpracování obrazu v mobilních robotických systémech a s tím spojenou problematikou. Jedna z knihoven počítačového vidění je knihovna OpenCV, na kterou je tato práce zaměřena.

V teoretické části jsou popsány vybrané knihovny počítačového vidění a je zde i jejich srovnání. Dále v této části jsou popsány teoretické znalosti pro konstrukci ukázkového programu, který je popsán v části praktické.

Praktická část se věnuje popisu konstrukce ukázkového programu. Program se dá rozdělit na dva jednodušší, první z nich je detekování laserových stop. Pro tuto detekci se využívá kombinace dvou metod. První z nich je světlost pixelu a druhá je červenost pixelu. Druhá část se zabývá transformací bodu nalezeného pomocí laserových stop z jednoho obrazu do druhého.

I. TEORETICKÁ ČÁST

1 ZPRACOVÁNÍ OBRAZU V MOBILNÍCH ROBOTICKÝCH SYSTÉMECH

Počítačové vidění můžeme chápat jako transformaci dat ze statického nebo dynamického obrazu, pořízeného například web kamerou, k zachycení stavu, nebo pro novou prezentaci. Všechny tyto transformace se provádějí s určitým cílem. Vstupní data mohou obsahovat i další informace. Novou prezentací je například změna barevného obrazu na černobílý. Díky tomu, že lidé jsou vizuálně založení, je snadné se zmýlit v tom, že počítačové vidění je jednoduchá úloha. Lidská intuice může být velmi matoucí. Lidský mozek dělí vidění na několik různých kanálů, z nichž informace posléze posílá do mozku. Mozek má pozorovací systém, který identifikuje na základě důležitosti úlohy, důležité části obrazu prozkoumá a zároveň potlačuje zpracování dalších oblastí. Je zde velká zpětná vazba na vizuální podněty, kterým se doposud nepodařilo porozumět. Existují rozsáhlé asociativní vstupy pocházející od svalových kontrolních senzorů a od všech ostatních smyslů, které dovolují mozku čerpat křížové asociace vytvořené za léta života. Smyčky zpětných vazeb v mozku se vracejí zpět ke všem stupňům zpracování, včetně samotných hardwarových senzorů (očí), které mechanicky kontrolují světlo cestou paprsku a ladí příjem na povrchu sítnice.

V systému strojového vidění však počítač přijímá síť čísel z kamery nebo disku.. Z větší části tu není vestavěné obrazové rozpoznání, žádná automatická kontrola ohniska a clony, žádné křížové asociace vzniklé lety zkušeností. Většinou jsou vizuální systémy stále pohádkově naivní.

Ve skutečnosti problém, jak jsme jej ze široka vytýčili, je více než obtížný: je formálně nemožné jej vyřešit. Je-li dáno dvourozměrné vidění (2D) trojdimenzionálního světa (3D), neexistuje žádný způsob rekonstrukce 3D signálu. Takto formálně špatně stanovený problém nemá jedinečné nebo definitivní řešení. Ten sám 2D obraz může přestavovat nějakou nekonečnou kombinaci 3D scény, i kdyby byla data dokonalá. Avšak, jak bylo již zmíněno, data jsou porušena šumem a deformacemi. Takové poškození pochází ze změn ve světě (počasí, světlo, odraz, pohyb), nedokonalostí čočky a mechanických částí, rozmazání vzniklé pohybem (konečný integrovaný čas na senzoru), elektrickým šumem v senzoru, nebo v jiných elektronických částech, nebo vzniká kompresí po pořízení snímku. Je třeba zjistit, zda lze dosáhnout nějakého pokroku.

V návrhu praktického systému může být často použito dodatečné souvislosti za účelem překonání omezení, která jsou nám dána vizuálními senzory. Připomeňme si příklad pohyblivého robota, který musí najít a sebrat sešívačky v budově. Robot může využít fakt, že stůl je objekt nalézající se uvnitř stavby a že sešívačky se nacházejí většinou na pracovních stolech. To poskytuje implicitní údaj: sešívačky se musí vejít na pracovní stůl. Také to pomáhá rychle eliminovat sešívačky na nesmyslných místech (tj. na stropě, nebo na okně). Robot může bezpečně ignorovat reklamní aerostat tvářící se jako sešívačka, protože aerostat nemá pozadí představující dřevo pracovního stolu. Naproti tomu s takovými úkoly, jako je znovuzískání obrazu, všechny obrazy sešívačky v databázi mohou být skutečnými sešívačkami a tak rozměrné velikosti a jiné neobvyklé konfigurace mohou být implicitně znemožněny domněnkami těch, kteří udělali fotografie. To znamená, že fotograf pravděpodobně pořizoval snímky jen reálných, normálně velikých sešívaček. Lidé se také snaží vystředit objekty, když pořizují snímky, a dát jim správnou orientaci. Tím je dáno, že ve fotografiích pořízených lidmi existuje mnoho nezáměrných implicitních informací.

V té souvislosti může být tedy informace modelována explicitně technikami strojního učení. Skryté proměnné, jako je velikost, orientace atd., mohou pak být dány do vztahu s hodnotami v označené tréninkové sestavě. Alternativně se lze pokusit měřit skryté zkreslené proměnné za použití doplňkových senzorů. Užití laserového hledáčku k měření hloubky nám dovoluje přesně měřit velikost objektu.

Další problém spojený s počítačovým viděním je šum. Obvykle se setkáváme s šumem při použití statistických metod. Například může být nemožné detekovat hranu v obrazu pouze srovnáním bodu v jejím bezprostředním sousedství. Ale když se podíváme na statistiku v oblasti, detekce hran se stane jednodušší. Skutečná hrana se může objevit jako řada takových bezprostředně sousedících odezev v oblasti, jejichž každá orientace je konzistentní se sousedy. Je také možné kompenzovat šum získáním statistických údajů přes čas. Také jiné techniky se snaží odstranit šum nebo distorze vytvořením explicitních modelů naučených přímo z dostupných dat. Například vzhledem k tomu, že čočkové distorze jsou dobře známé, je třeba zjistit parametry pro jednoduchý polynomiální model za účelem popisu takových distorzí (a tak je téměř úplně opravit).

Akce, nebo rozhodnutí, o které se počítačové vidění snaží, založené na datech z kamery jsou vytvořeny v kontextu se zvláštním účelem nebo cílem. Můžeme chtít odstranit šum

nebo poškození obrázku tak, že náš bezpečnostní systém vydá varovný signál, jestliže se někdo pokusí přelézt plot, nebo protože potřebujeme monitorovat systém, který počítá, kolik lidí přechází přes zábavní park. Vizuální software pro roboty, kteří cestují budovami, použije různé strategie, ve srovnání s vizuálním softwarem pro stacionární bezpečnostní kamery, protože dva systémy mají významně různé souvislosti a cíle. Jako obecné pravidlo platí: čím více jsou omezeny souvislosti počítačového vidění, tím více můžeme reagovat na ona omezení zjednodušením problému a tím více odpovídající může být naše konečné řešení.

OpenCV je zaměřeno na poskytnutí základních nástrojů, potřebných k řešení problémů počítačového vidění. V některých případech jsou vysokoúrovňové funkce v knihovně dostatečné k řešení komplexnějších problémů v počítačovém vidění. I v případě, že tomu tak není, základní komponenty v knihovně úplně dostačují k vytvoření vlastního kompletního řešení téměř všech problémů počítačového vidění. V dalším případě je tu několik pokusných metod k užití knihoven: všechny začínají řešením problému za užití všech dostupných knihovních komponent. Obvyklé je, že v případě vyvinutí prvního řešení se vyjasní, kde má řešení nedostatky, a pak lze opravit tyto nedostatky za použití vlastního kódu a chytrosti (lépe známo jako „vyřeš svůj současný problém, ne ten, který si představuješ“). Lze pak použít vlastní řešení jako měřítko hodnoty zlepšení, kterého bylo dosaženo. Na základě toho mohou být všechny nedostatky, které zůstanou, vyřešeny v kontextu širšího systému, ve kterém je problém zakotven. [1]

1.1 OpenCV

1.1.1 Úvod

Knihovna počítačového vidění byla původně vyvinutá společností Intel. Nyní je tato knihovna volně šiřitelná pro vývoj a komerční použití pod licencí BSD. Knihovna je napsána v C a C++ a lze ji provozovat na mnoha operačních systémech, jako je Windows, Linux a další.

1.1.2 Historie

Projekt byl oficiálně zahájen v roce 1999 firmou Intel, která spolupracovala na vývoji s ruským expertním týmem zaměřeným na zvýšení výkonu knihoven. Projekt byl původně

zpracováván jako výzkum aplikací, které silně zatěžují procesor. Na začátku si projekt stanovil několik výchozích úkolů:

- Pokročilý vývoj vidění, které by poskytovalo nejen otevřený, ale i optimalizovaný kód pro základní infrastrukturu vidění.
- Rozšiřování znalosti vidění tak, aby s využitím společné infrastruktury mohli vývojáři vytvářet kód, který by byl snadno čitelný a přenositelný.
- Pokročilé komerční aplikace v oblasti vidění, využívající přenosný, volně šiřitelný kód s licencí, která nevyžaduje otevření komerční aplikace nebo její volné šíření.

Dostupnost aplikací počítačového vidění by zvýšilo poptávku po rychlejších procesorech. Snad proto vznikl tento otevřený a svobodný kód od hardwarového prodejce místo od softwarové společnosti. Nyní OpenCV vylepšuje mnoho uživatelů i mimo samotný Intel. V průběhu vývoje byly doby, kdy se na vývoji nikdo z firmy Intel nepodílel, avšak s vypuštěním několika jádrových procesorů a spoustou nových aplikací začaly hodnoty OpenCV zase růst. Nyní je OpenCV oblastí vývoje v několika institucích, a tak lze očekávat mnoho aktualizací v oblasti multikamerové kalibrace, vnímání hloubky a také kombinace počítačového vidění a měření vzdálenosti pomocí laseru.

1.1.3 Přenositelnost

OpenCV bylo navrženo pro přenositelnost a původně pro sestavení pomocí Borland C++, MSVC++ a intelovských kompilátorů. To znamená, že kód napsaný v C a C++ je poměrně standardní, aby přenositelnost mezi platformami byla jednodušší. Obrázek 1 ukazuje platformy, na nichž je vyzkoušen běh knihovny OpenCV. Nejvyspělejší je podpora pro 32-bitovou architekturu Intel (IA32) pro Windows, následně pro Linux na stejné architektuře. Přenositelnost na architekturu Mac OS X se stala prioritou poté, co Apple začal používat procesory Intel. (Použití knihovny v OS X není tak vyspělé jako ve Windows nebo Linuxu, ale to se mění rychle). Tento vývoj je následován podporou 64-bitového rozšíření pamětí (EM64T) a 64-bitovou architekturou Intel (IA64). Nejméně vyspělá je přenositelnost na hardware Sun a jiné operační systémy.

Pokud se architektura nebo OS neobjevuje na Obrázku 1, neznamená to, že zde není přenositelnost. OpenCV byla portována na téměř každý komerční systém i na PowerPC Macy – robotické psy. OpenCV běží dobře i na linii procesorů AMD a také další

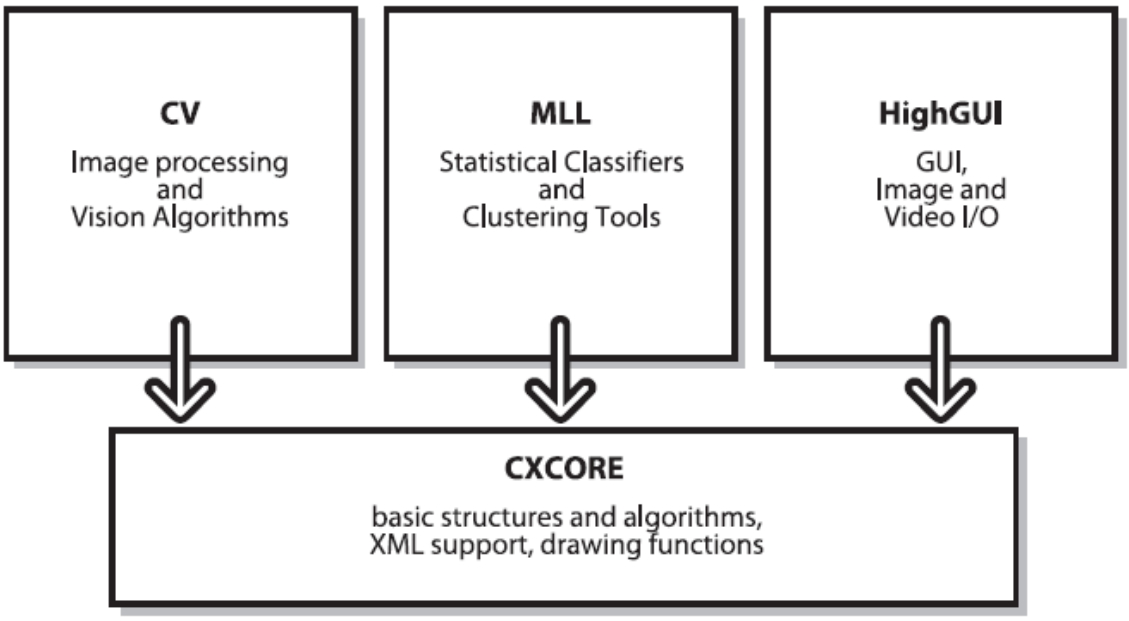
optimalizace dostupná v IPP bude využívat multimediální rozšíření (MMX) v AMD procesorech, které jsou spojené s touto technologií.

	IA32	EM64T	IA64	Other (PPC, Sparc)
Windows	✓ (w. IPP; MSVC6, .NET2005+OMP, ICC, GCC, BCC)	✓ (w. IPP; MSVC6+PSDK.NET2005+OMP, PSDK)	± (w. IPP; PSDK, some tests fail)	N/A
Linux	✓ (w. IPP; GCC, BCC)	✓ (w. IPP; GCC, BCC)	✓ (GCC, ICC)	✗
MacOSX	✓ (w. IPP, GCC, native APIs)	? (not tested)	N/A	✓ (iMac G5, GCC, native APIs)
Others (BSD, Solaris…)	✗	✗	✗	Reported to build on UltraSparc Solaris

Obrázek 1. Platformy, na nichž je vyzkoušen běh knihovny OpenCV.

1.1.4 Struktura a obsah

OpenCV je strukturovaná do pěti hlavních komponent, z nichž čtyři jsou uvedeny v Obrázku 2. Složka CV obsahuje základní zpracování obrazu a vyšší úroveň algoritmů počítačového vidění. ML je knihovna strojového učení, která obsahuje mnoho statistického třídění a seskupovacích nástrojů. HighGUI obsahuje I/O rutiny a funkce pro ukládání a načítání videa a obrázků. CXCore obsahuje základní datové struktury a jejich obsah. Obrázek 2 nezahrnuje CvAux, tato komponenta obsahuje obě zaniklé oblasti (vestavěné HMM rozpoznávání obličeje) a experimentální algoritmy (pozadí / popředí segmentace). CvAux dosud není příliš dobře zdokumentováno.



Obrázek 2. Struktura knihovny OpenCV.

1.1.5 Budoucnost

Množství aplikací počítačového vidění rychle roste, od kontroly výrobků přes indexování videa na webu, medicínské aplikace, dokonce i pro lokální navigování na Marsu. Knihovna OpenCV také roste, aby se přizpůsobila tomuto vývoji.

OpenCV byla dlouhou dobu podporována firmou Intel a nyní také dostává podporu od Willow Garage, výzkumného institutu robotiky a technologického inkubátoru.

Jednou z nových klíčových oblastí rozvoje OpenCV je robotické vnímání, které se zaměřuje na 3D vnímání, stejné jako 2D a 3D poznávání objektů, neboť kombinace různých datových typů vylepší vlastnosti detekce objektů, segmentace a rozpoznávání. Robotické vnímání je do značné míry závislé na 3D snímání, a proto je vyvíjeno úsilí s cílem vylepšení kamerové kalibrace, opravy a komunikace s více kamerami a také kombinace kamera + laserové měření vzdálenosti.

1.2 Generic Image Library

1.2.1 Úvod

Obrázky jsou základními prvky každého projektu, který zahrnuje grafiku, zpracování obrazu a videa a přitom proměnlivost dat reprezentovaných každým pixelem (barevný prostor, bitová hloubka, kanál, zarovnání). Je těžké psát kód související s obrázky, který je zároveň efektivní a obecný. Generic Image Library (GIL) je napsána jako obecná knihovna v C++, která umožňuje psaní obecných zobrazovacích algoritmů se srovnatelným výkonem pro konkrétní typ obrázku. Knihovna byla konstruována s pěti následujícími cíly:

- Obecnost: Abstraktní obrazové vyjádření pro obrázky. Umožňuje psát kód pouze jednou a mít ho pro práci na jakémkoliv jiném typu obrázku.
- Výkon: Rychlost byla pomocná při návrhu knihovny. Obecné algoritmy v knihovně poskytují srovnatelnou rychlost s ručně psanými algoritmy pro konkrétní typ obrázku.
- Flexibilita: Rozšíření vyplněných parametrů zrychluje kód, ale přísně omezuje jeho flexibilitu. Knihovna umožňuje, aby jakýkoliv parametr obrázku byl za běhu upřesněn (pro malé náklady na výkonnost, srovnatelné s virtuální volací režií).
- Rozšiřitelnost: GIL umožňuje vytvoření virtuálních součástí – kanály, barevný prostor, pixely, čítače pixelů, lokátory, zobrazení obrázků a algoritmů, které mohou být nahrazeny.
- Kompatibilita: Knihovna je navržena jako STL a oživení doplňků. Obecné STL algoritmy mohou být použity pro manipulaci s pixely, jsou především optimalizované. Knihovna funguje nativně pro stávající data charakterizující pixel.

1.3 VXL (the Vision-*something*-Libraries)

1.3.1 Úvod

VXL je soubor knihoven počítačového vidění napsaných v C++ pro použití ve výzkumu a užití. Byla vytvořena z TargetJr a IUE s cílem vytvořit lehký, rychlý a konzistentní

systém. VXL je napsána v ANSI / ISO C++ a je určena pro přenos mezi mnoha platformami. Knihovny jádra VXL jsou:

- Vnl (numerické): Numerické kontejnery a algoritmy, např. matice, vektory, dekompozice.
- Vil (obrázky): Načítání, ukládání a manipulaci s obrázky mnoha běžných formátů, včetně velmi velkých obrázků.
- Vgl (geometrie): Geometrie pro body, křivky a další základní objekty v 1, 2 nebo 3 rozměrech.
- VSL (streamování I / O), vbl (základní šablony), Vul (utility): Různé na platformě nezávislé funkce.

Stejně jako hlavní knihovny existují knihovny zahrnující numerické algoritmy pro zpracování obrazu, koordinační systémy, kamerové geometrie, stereo, video manipulace, rekonstrukci struktur z pohybu, GUI design, klasifikace, robustní odhady, funkce sledování, topologie, manipulace se strukturami, 3D zobrazení atd.

Každá knihovna jádra je samostatná a všechny mohou být použity bez ohledu na ostatní knihovny jádra. Podobně jsou na tom ostatní knihovny, které nejsou závislé více, než je potřeba, takže lze projekt zkompilovat a odkazovat pouze na knihovny, které jsou opravdu třeba.

VXL je vyvíjena a používána v mezinárodním týmu v akademické obci a průmyslu, včetně některých předních světových odborníků počítačového vidění.

1.4 NokiaCV

1.4.1 Úvod

NokiaCV je postavena na mobilním operačním systému Symbian, rozšiřuje zobrazovací schopnosti a standardizuje obraz uvnitř operačního systému. Knihovna poskytuje standardní grafické operace, stejně jako soubor lineární algebry, který je zapotřebí pro mnoho pokročilých aplikací zpracovávajících obraz. Tyto vlastnosti slouží jako základní stavební kameny pro pokročilejší knihovny. Některá rozšíření knihovny, která mohou být

vytvořena výzkumnými týmy firmy Nokia, budou v budoucnu vypuštěna, ale kdokoliv jiný může knihovnu rozšířit.

1.4.2 Obsah knihovny

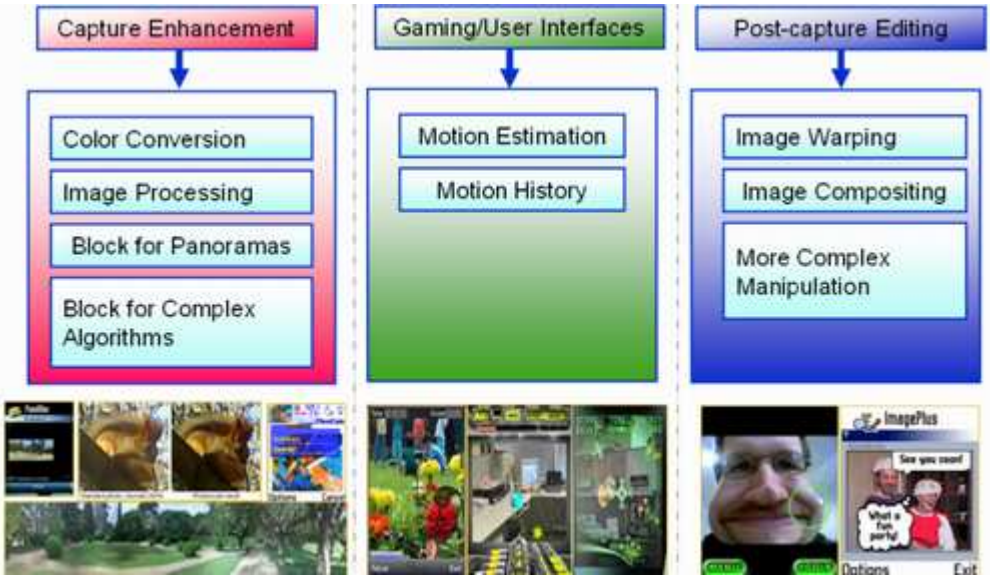
- Knihovna počítačového vidění Nokia
    - Obsahuje základní funkce pro počítačové vidění a zpracování obrazu
    - Může být použita na OS Symbian S60
- Knihovna počítačového vidění Nokia je vázaná na programovací jazyk Python
    - Jazyk python je vázaný na všechny funkce původní knihovny
    - Může být použita na PyS60
- Knihovna pro základní odhad pohybu fotoaparátu (stabilizace obrazu)
    - Obsahuje algoritmy pro odhad pohybu fotoaparátu
    - Může být použita na OS Symbian S60
- Knihovna pro základní odhad pohybu fotoaparátu (stabilizace obrazu) je vázána na programovací jazyk Python
    - Jazyk python je vázaný na algoritmy pro odhad pohybu fotoaparátu
    - Může být použita na PyS60
- Knihovna pro základní funkce fotoaparátu je vázaná na programovací jazyk Python
    - Jazyk python je vázaný na mnoho funkcí fotoaparátu z S60
    - Může být použita na PyS60

1.4.3 Příklady použití knihovny

Zdokonalené zachycování obrazu: pomocí barevné konverze a image wrapper můžeme přímo pracovat a měnit jednotlivé body obrazu. Vývojáři mohou vytvořit vysoce kvalitní algoritmy pro panoramata a také mohou implementovat složité algoritmy bez nutnosti vytvoření a odladění platformy pro vývoj.

Pro uživatelské rozhraní, popřípadě rozhraní pro hry, poskytuje knihovna díky maximálnímu odhadu pohybu softwarové komponenty, které jsou připraveny pro hry a aplikace.

Zpracování již zachyceného obrazu pomocí operace image warping může být použito do mnoha zábavných aplikací. Pomocí image compositing můžeme vkládat objekty do již zachycených obrazů.



Obrázek 3. Příklady použití knihovny.

1.5 Shrnutí

Přehled a shrnutí pozitivních a negativních vlastností knihoven počítačového vidění.

1.5.1 OpenCV, Open Computer Vision

Výhody:

- velmi jednoduchá
- spolupracuje s programovacími jazyky C a C++
- velmi široká škála algoritmů
- komplexní algoritmy: detekce obličejů…
- populární

Nevýhody:

- nutnost použít IplImage z OpenCV
- Byte barev v IplImage je seřazen jako BGR namísto obvyklého RGB

1.5.2 GIL, Generic Image Library

Výhody:

- založeno jen na hlavičkových souborech
- nabízí wrapper pro mnoho formátů obrázků
- algoritmus napsaný pro jeden formát obrázků bude fungovat u většiny ostatních formátů

Nevýhody:

- nepřichází s více algoritmy pro zpracování obrazu

1.5.3 VXL, Vision X Library

Výhody:

- dobře otestovaná technologie
- jednoduchý konstrukční proces pomocí cmake
- využívají moderní programovací techniky: Třídy, šablony a STL
- má mnoho funkcí
- jednoduché začít s touto knihovnou

Nevýhody:

- není běžné použití STL
- třída struktura je v některých místech složitější

2 BAREVNÉ MODELY

Barevné modely se snaží napodobit co nejvěrněji barvy v přírodě, které jsou způsobeny směsí světla různých vlnových délek. Popisují základní barvy a samotný model, který slouží pro míchání základních barev do výsledné barvy. Nejčastěji se používají ty barevné modely, u kterých není příliš složitý model a které mají věrné podání barvy. Mezi tyto zástupce patří barevné modely: **RGB** (Red - červená, Green - zelená, Blue - modrá), do něj se ukládá většina fotografií, **CMYK** (Cyan - azurová, Magenta - purpurová, Yellow - žlutá, blacK - černá), určený zejména pro tisk jako zástupce subtraktivního míchání barev, a **HSV** (Hue - barva, Saturation - sytost, Value – hodnota jasu), označovaný také jako model HSB.

2.1 Aditivní míchání barev

Barevné modely pracují s jednotlivými složkami barev, které se sčítají a vytvářejí světlo vyšší intenzity. Smícháním dvou základních barev vznikne třetí, základní barva, která je barvou komplementární (doplňkovou).

2.1.1 Barevný model RGB

Jedná se o jeden z nejznámějších zástupců aditivního míchání barev, používaný například v monitorech. Skládá se ze tří základních barev: červená, zelená a modrá. Kombinace všech tří barev při maximální intenzitě vytvoří bílou barvu. Barvy lze reprezentovat vektorem, u kterého jednotlivé složky nabývají hodnoty z intervalu <0,1>. Kódování bývá obvykle u každé složky do jednoho bytu, což představuje celočíselnou hodnotu od 0 do 255. Poté se počet odstínů, které lze vytvořit, rovná $256^3 = 16777216$. U některých zařízení je nutno počet odstínu uměle redukovat, protože nejsou schopná jich tolik zobrazit. Používá se transformace barev pomocí několika různých metod. Barevný model RGB můžeme v prostoru reprezentovat krychlí, která má v počátku souřadnic černou barvu [0,0,0] a v protilehlém vrcholu barvu bílou [1,1,1]. Vrcholy, které leží na hlavních osách, mají základní barvy červenou, zelenou a modrou a ostatní vrcholy jsou barvy doplňkové. Čím větší součet jednotlivých složek dostaneme, tím světlejší bude výsledná barva.

Pro převod barevného modelu **RGB na šedotónový** se používá vztah pro výpočet jasu, založený na tom, že lidské oko vnímá různě stejnou intenzitu různých barev. Nejcitlivěji

vnímá zelenožlutou. Vztah s vyjádřenými konstantami pro intenzitu je následující: $I = 0{,}299R + 0{,}587G + 0{,}114B$.

Převod barevného modelu **RGB na HSV** má tvar algoritmu. Pokud jednotlivé složky RGB nabývají hodnot z intervalu $r, g, b \in \langle 0,1 \rangle$ a v maximu bude jedna složka a zároveň jedna bude minimální, tak pro výpočet barevného úhlu $h \in \langle 0,360 \rangle$ použijeme následující algoritmus:

$$h = \begin{cases} 0 & \max = \min \\ \left(60^\circ \cdot \dfrac{g-b}{\max - \min} + 360^\circ \right) \bmod 360^\circ & \max = r \\ 60^\circ \cdot \dfrac{b-r}{\max - \min} + 120^\circ & \max = g \\ 60^\circ \cdot \dfrac{r-g}{\max - \min} + 240^\circ & \max = b \end{cases} \qquad (1)$$

Pro výpočet sytosti a hodnoty jasu $s, v \in \langle 0,1 \rangle$ použijeme následující dva algoritmy:

$$s = \begin{cases} 0 & \max = 0 \\ \dfrac{\max - \min}{\max} = 1 - \dfrac{\min}{\max} & \textit{jinak} \end{cases} \qquad (2)$$
$$v = \max$$

2.2 Subtraktivní míchání barev

Sem řadíme barevné modely, které při přidávání barev snižují intenzitu původního světla. Základní barvy subtraktivního míchání jsou komplementární (doplňkové) k základním barvám při jejich aditivním míchání.

2.2.1 Barevný model CMYK

Barevný model CMYK je zástupcem subtraktivního míchání barev, který se používá v tiskařské technice. Tento model je mnohem bližší lidským zkušenostem s mícháním barev. Ideální model by obsahoval pouze první tři barvy CMY, bohužel smícháním těchto barev nevznikne černá, proto se tato barva přidává k barevnému modelu.

2.3 Barevný model HSV

Tento model nejvíce odpovídá lidskému vnímaní barev. Zatímco předešlé barevné modely vycházely spíše z technické praxe, tento barevný model je intuitivní. Nejčastěji používaná prezentace modelu HSV je kužel, který je vhodný, protože zobrazuje všechny barvy v jediném objektu. Pro vytvoření máme tři rozměry. Barva je úhel, kde počáteční hodnota je červená. Sytost je reprezentovaná vzdáleností od centra kruhového průřezu a hodnota je vzdálenost středu od špičatého konce kuželu. Tento model je vhodný pro představu o barevném modelu v jednom objektu, ale není vhodný v dvojdimenzionálních rozhraních pro výběr barvy. [2]

Převod barevného modelu **HSV** na **RGB** má tvar algoritmu, v němž definujeme hodnoty h, s, v jako hodnoty barevného modelu HSV a barevný úhel $h \in \langle 0,360 \rangle$ a $s, v \in \langle 0,1 \rangle$ jako hodnoty sytosti a jasu. Podobně jako trojici u barevného modelu HSV nadefinujeme trojici pro barevný model RGB jako hodnoty $r, g, b \in \langle 0,1 \rangle$. Nakonec má algoritmus tento tvar:

$$\begin{aligned} h_i &= \left(\frac{h}{60}\right) \bmod 6 \\ f &= \frac{h}{60} - \left(\frac{h}{60}\right) \\ p &= v \cdot (1 - s) \\ q &= v \cdot (1 - f \cdot s) \\ t &= v \cdot (1 - (1 - f) \cdot s) \end{aligned} \tag{3}$$

Výpočet barevného vektoru:

$$(r, g, b) = \begin{Bmatrix} (v, t, p) & h_i = 0 \\ (q, v, p) & h_i = 1 \\ (p, v, t) & h_i = 2 \\ (p, q, v) & h_i = 3 \\ (t, p, v) & h_i = 4 \\ (v, p, q) & h_i = 5 \end{Bmatrix} \tag{4}$$

3 DETEKCE HRAN

3.1 The Harris & Stephens / Plessey algoritmus detekce hran

Tento algoritmus detekce hran pracuje s posuvným políčkem, které se posouvá různým směrem a u každého bodu tohoto políčka se počítá gradient. Na rozdíl do algoritmu Moravec tento algoritmus počítá přímo s určeným směrem. Z tohoto algoritmu vychází mnoho novějších algoritmů pro zpracování detekce hran.

Na každý bod políčka se aplikuje základní metoda SSD, neboli rozdíl čtverců jasu. Taky se tato metoda nazývá autokorelační.

Budeme předpokládat, že při použití šedotónového dvojdimenzionálního obrazu nedojde ke ztrátě. Vycházíme z toho, že tento obraz je I. Použijeme obrazové políčko na plochu (u, v) a přesouváme je pomocí (x, y). Uvažovaný součet čtverečních rozdílů mezi těmito dvěma políčky, označovanými S, je dán vztahem:

$$S(x,y)=\sum_u\sum_v w(u,v)\left(I(u,v)-I(u+x,v+y)\right)^2 \tag{5}$$

Aproximace $I(u+x,v+y)$ pomocí Tailorova rozvoje,

$$I(u+x,v+y)\approx I(u,v)+I_x(u,v)x+I_y(u,v)y \tag{6}$$

získáme

$$S(x,y)\approx\sum_u\sum_v w(u,v)\left(I_x(u,v)x+I_y(u,v)y\right)^2 \tag{7}$$

kde I_x a I_y jsou parciální derivace I, nebo podobné

$$S(x,y)\approx\begin{pmatrix}x & y\end{pmatrix}A\begin{pmatrix}x\\y\end{pmatrix} \tag{8}$$

kde A je tensor struktury, pokud existují všechny druhé parciální derivace, jedná se o Hessovu matici,

$$A=\sum_u\sum_v w(u,v)\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}=\begin{bmatrix}\langle I_x^2\rangle & \langle I_xI_y\rangle\\ \langle I_xI_y\rangle & \langle I_y^2\rangle\end{bmatrix} \tag{9}$$

významné body se stanoví pomocí výpočtů hodnot lambda:

$$\lambda_{1,2} = \frac{\left\langle I_x^2 \right\rangle + \left\langle I_y^2 \right\rangle \pm \sqrt{\left(\left\langle I_x^2 \right\rangle - \left\langle I_y^2 \right\rangle\right)^2 + 4 \cdot \left\langle I_x I_y \right\rangle \cdot \left\langle I_x I_y \right\rangle}}{2} \tag{10}$$

lambda může nabývat těchto hodnot:

- pokud $\lambda_1 \approx 0$ a $\lambda_2 \approx 0$, potom tento pixel (x, y) nemá vlastnosti hledaného bodu
- pokud $\lambda_1 \approx 0$ a λ_2 má libovolně velkou pozitivní hodnotu, potom je nalezena hrana

pokud λ_1 a λ_2 mají libovolně velkou pozitivní hodnotu, potom je nalezen roh.

Harris a Stephans si uvědomili, že výpočet vlastních hodnot je početně a časově náročný, protože vyžaduje výpočet druhé odmocniny. Proto navrhli funkci Mc, kde κ je laditelný citlivostní parametr. Hodnota κ byla určena experimentálně. Nejlepších výsledků má metoda dosahovat pro $\kappa = 0{,}04...0{,}15$.

$$M_c = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det(A) - \kappa \operatorname{trace}^2(A) \tag{11}$$

Z tohoto vyplývá, že není nutné počítat vlastní číslo rozkladem matice A a že stačí spočítat determinant a sledovat A pro nalezení rohů nebo obecně bodů zájmu. [3]

3.2 Přehled detektorů hran

Tabulka 1. Volně šiřitelé detektory a jejich klasifikace.

detektor	hrana	roh	skvrna
Canny	x		
Sobel	x		
Harris & Stephens / Plessey	x	x	
SUSAN	x	x	
Shi & Tomasi		x	
Level curve curvature		x	
FAST		x	

Laplacian of Gaussian	x	x
Difference of Gaussians	x	x
Determinant of Hessian	x	x
MSER		x
Grey-level blobs		x

4 ZPRACOVÁNÍ OBRAZU

4.1 Místní předzpracování

Objektem našeho zájmu je metoda předpracování, která využívá malou vzdálenost pixelů ve vstupním obrazu, aby vytvořila novou hodnotu jasu ve výstupním obrazu. Takovéto předzpracující operace jsou nazývány **filtrací**, pokud se užívá terminologie signálního zpracování.

Metody místního předzpracování lze rozdělit do dvou skupin podle cíle zpracování. Zaprvé, **vyhlazování** má za úkol potlačit šum nebo jiné drobné změny v obrazu: je to obdobné jako při potlačení vysokých frekvencí ve Fourierově transformační doméně. Naneštěstí vyhlazení rozostří všechny ostré okraje, které jsou nosiči důležitých informací o obrazu. Zadruhé, **gradientní operátory** jsou založeny na místních odvozeninách obrazové funkce. Deriváty jsou větší na místech obrazu, kde obrazové funkce podléhají rychlým změnám, a cílem gradientních operátorů je označit taková místa v obrazu. Gradientní operátory mají podobný efekt potlačovat nízké frekvence ve Fourierově transformační doméně. Šum je často vysoká frekvence v přírodě: naštěstí jestliže gradientní operátor je použit na obrazu, stoupá zároveň úroveň šumu.

Je zřejmé, že vyhlazení a gradientní operátory mají protichůdné cíle. Některé předzpracující algoritmy řeší tento problém a dovolují současně vyhlazení a vylepšení.

Další klasifikace místních předzpracujících metod je podle transformačních vlastností: lze rozlišit **lineární** nebo **nelineární** změny. Lineární operace vypočítávají výslednou hodnotu ve výstupním obrazovém pixelu $g(i, j)$ jako lineární kombinaci jasu v sousedství Θ pixelu $f(i, j)$ ve vstupním obrazu. Příspěvek pixelů v sousedství Θ je zhodnocen koeficienty *h*:

$$f(i, j) = \sum_{(m,n)\in\Theta}\sum h(i-m, j-n)g(m, n) \qquad (12)$$

Rovnice je shodná s diskrétní konvolucí s kernelem *h*, která se nazývá **konvoluční maska**. Pravoúhlá sousedství Θ jsou často použita s lichým počtem pixelů v řádcích a sloupcích, umožňující specifikaci prostředního pixelu ze sousedství.

Metody místního předzpracování používají typicky velmi malou předchozí znalost o obsahu obrázku. Je velmi těžké vyvozovat tuto znalost, zatímco je obraz zpracováván, když známí sousedé Θ zpracovávaného pixelu jsou malí. Vyhlazovací operace bude úspěšná, jestliže je dostupná nějaká obecná znalost o degradaci obrazu. Tím mohou být např. statistické parametry šumu.

Výběr místní transformace, velikost a ostrost sousedního Θ závisí přímo na velikosti objektu ve zpracovávaném obrazu. Jestliže je objekt dosti velký, obraz může být zvětšen vyhlazením malých degradací.

Operace založené na konvoluci (filtrování) mohou být použity pro vyhlazení, gradientní operátory a liniové detektory. Jsou to metody, které umožňují urychlení výpočtů k jednoduchému implementování do hardwaru.

4.1.1 Vyhlazování obrazu

Vyhlazování obrazu je síť metod místního předzpracování, jejíž převažující užití je potlačení obrazového šumu – využívá nadbytečnost v obrazových datech. Výpočet nové hodnoty je založen na zprůměrování hodnot jasu v sousedních Θ. Vyhlazování nastoluje problém rozmazávání ostrých hran v obrazu, a tak se musíme soustředit na metody vyhlazování, které **zachovávají hrany**. Jsou založeny na obecné myšlence, že průměr je vypočítán pouze z těch bodů v sousedství, které mají podobné vlastnosti jako bod, který je zpracováván.

Místní vyhlazování obrazu může účinně eliminovat impulzivní šum nebo degradace objevující se jako tenké pruhy, ale nefunguje, jestliže degradace jsou velké kapky nebo tlusté čáry. Řešením pro komplikované degradace může být užití technik obnovujících obraz.

4.1.1.1 Průměrování

Předpokládejme, že šumová hodnota ν v každém pixelu je nezávislá náhodná proměnná s nulovou hodnotou a standardní odchylkou σ. Můžeme takový obraz získat několikerým statickým zachycením téže scény. Výsledek vyhlazování je průměrem toho samého bodu n v těchto obrazech $g_1,...,g_n$ s hodnotami šumu $\nu_1,...,\nu_n$:

$$\frac{g_1 + ... + g_n}{n} + \frac{v_1 + ...v_n}{n} \tag{13}$$

Druhý termín zde popisuje efekt šumu, který je zase náhodná hodnota s nulovou hodnotou a standardní odchylkou $\sigma / \sqrt{n}$, standardní odchylka se zvyšuje faktorem $\sqrt{n}$.

Potom, jestliže jsou dostupné *n* obrazy téže scény, vyhlazování může být provedeno bez rozmazání obrazu.

$$f(i,j) = \frac{1}{n}\sum_{k=1}^{n} g_k(i,j)$$
$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{14}$$

V mnoha případech je dostupný jen jeden obraz s šumem a průměrování je pak realizováno v místním sousedství. Výsledky jsou přijatelné, pokud je šum velikostně menší než nejmenší objekt zájmu v obrazu, ale vážnou nevýhodou je rozmazání hran. V případě vyhlazování v jednom obrazu musíme předpokládat, že neexistují změny v šedých úrovních základních dat obrazu. Tento předpoklad je jasně porušen na místech obrazových hran a rozmazání hran je přímým důsledkem porušení předpokladu. Průměrování je zvláštní případ samostatné konvoluce. Pro sousedství 3 x 3 je konvoluční maska *h*:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{15}$$

Význam pixelu v centru konvoluční masky *h* nebo sousedství čtyř je někdy zvyšován, protože lépe dosahuje hodnot šumu s Gaussovou pravděpodobnou distribucí.

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{16}$$

Větší konvoluční masky pro průměrování jsou vytvořeny analogicky podle Gaussova distribučního vzorce a maskové koeficienty jsou normalizovány tak, aby měly jednotkovou velikost.

Alternativní techniky, které jsou převážně nelineární, nejsou vzaty v úvahu. Cílem je nerozmazávat ostré hrany, když se vyhneme průměrování skrze hrany.

4.1.1.2 Průměrování s omezenou datovou validitou

Metody, které zprůměrují s omezenou datovou validitou, se snaží zamezit rozmazání tím, že zprůměrují pouze ty pixely, které splňují určitá kritéria, a cílem je předcházet zapojení těch pixelů, které mají zvláštní charakteristiku.

Velmi jednoduchým kritériem je použití obrazového zprůměrování jen u těch pixelů v originálním obrázku s jasností v předdefinovaném intervalu s neplatnými daty [min, max], které typicky korespondující s intervalem šumu v šedé úrovni, nebo s jinými obrazovými chybami. S ohledem na bod (m, n) v obrazu, konvoluční maska je vypočítána v sousedním Θ z nelineárního vzorce:

$$h(i,j)=\begin{cases}1 & pro\ g(m+i,n+j)\notin[\min,\max]\\ 0 & jinak\end{cases} \qquad (17)$$

kde (i, j) specifikuje prvek masky. Proto jen hodnoty pixelů s neplatnou úrovní šedi jsou nahrazeny průměrem svých sousedů a jen platná data přispívají k sousedním průměrům.

Druhá metoda pracuje se zprůměrováním, jen když vypočtená změna jasu pixelu je v určitém předdefinovaném intervalu. Tato metoda dovoluje opravu rozsáhlých chyb, vycházející z pomalu se měnícího jasu pozadí bez ovlivnění zbytku obrazu.

Třetí metoda používá sílu hran (magnituda gradientu) jako kritérium. Magnituda určitého operátoru gradientu je zaprvé vypočtena pro celý obrázek a jen pixely ve vstupním obrázku s magnitudou gradientu menší než předdefinovaný práh jsou používány k zprůměrování. Tato metoda účinně odmítá zprůměrování na hranách, a proto potlačuje rozmazání, ale nastavení prahu je pracné.

4.1.1.3 Průměrování podle inverzního gradientu

Konvoluční maska je vypočtena u každého pixelu podle inverzního gradientu, podstatou je, že změna jasu v oblasti je obvykle menší než mezi sousedními oblastmi. Pokud necháte lokaci pixelu (m, n), aby odpovídala centrálnímu pixelu konvoluční masky s různou velikostí, inverzní gradient δ v bodu (i, j) s ohledem na (m, n) je pak

$$\delta(i, j) = \frac{1}{|g(m, n) - g(i, j)|} \tag{18}$$

Jestliže g (m, n) = g (i, j), potom definujeme δ(i, j) = 2: inverzní gradient delta je pak v intervalu (0,2] a δ je menší na hraně než uvnitř homogenní oblasti. Váhové koeficienty v konvoluční masce *h* jsou normalizovány inverzním gradientem a celý výraz je násoben 0,5, aby udržel jasové hodnoty v původní úrovni. Konstanta 0,5 má ten efekt, že určuje polovinu váhy ústředního pixelu (m,n) a druhou polovinu jeho souseda.

$$h(i, j) = 0{,}5 \frac{\delta(i, j)}{\sum_{(m,n) \in \Theta} \delta(i, j)} \tag{19}$$

Koeficient konvoluční masky korespondující s ústředním pixelem je definován jako *h*(i, j) = 0,5.

Tato metoda předpokládá ostré hrany. Když je konvoluční maska těsně u hrany, pixely z oblasti mají větší koeficienty než pixely blíže k hraně a to je nerozmazané. Izolované body šumu uvnitř homogenních oblastí mají malé hodnoty inverzního gradientu: body ze sousedství se podílejí na zprůměrování a šum je odstraněn.

5 PROJEKTIVNÍ TRANSFORMACE PRO OBRAZOVÉ DEFORMACE

5.1 Projektivní transformace

Pozoruhodnou vlastností je, že převrácení projektivní je stále projektivní transformace. Toto lze intuitivně vysvětlit obrácením roviny na transformaci roviny, kterou je projektivní transformování definováno. Matice pro inverzní transformace je inverzní, nebo sousedící s projektivním transformováním. (Sousedící matice jsou transpozice matice algebraického doplňku $\mathbf{M}^{-1} = \mathbf{adj(M)/det}(\mathbf{M})$). V homogenní algebře sousedící matice lze použít v místě inverzní matice vždy, když je nutná inverzní transformace, neboť obě jsou vzájemné skalární násobky, a sousedící matice vždy existuje, zatímco pokud matice nemá inverzní transformaci, potom je singulární. Inverzní transformace je tedy:

$$\begin{aligned}\mathbf{p_d} &= \mathbf{M_{ds}}\mathbf{p_d} \\ \begin{pmatrix} u' \\ v' \\ q \end{pmatrix} &= \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix}\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} \\ &= \begin{pmatrix} ei - fh & ch - bi & bf - ce \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{pmatrix}\begin{pmatrix} x' \\ y' \\ w \end{pmatrix}\end{aligned} \tag{20}$$

Když transformace bodu pomocí inverzní transformace spočítáme $(u,v)^T$ z $(x,y)^T$. Pokud $w \neq 0$ a $q \neq 0$, pak můžeme zvolit $w = 1$ a spočítat:

$$u = \frac{Ax + By + C}{Gx + Hy + I}, \qquad v = \frac{Dx + Ey + F}{Gx + Hy + I} \tag{21}$$

II. PRAKTICKÁ ČÁST

6 UKÁZKOVÝ PROGRAM

6.1 Metodika

Program byl vytvořen v programovacím jazyku C++, který je součástí vývojového prostředí Microsoft Visual Studia. Tento program lze rozdělit na dvě samostatné části. První část, představující hledání laserových stop, byla inspirována obecným postupem pro hledání těchto stop na wikipedii. Druhá část spočívá ve výpočtu bodu, který byl označen laserem, a jeho perspektivní transformace do místa určení.

Proces hledání laserových bodů v obrazu lze rozdělit do čtyř kroků a související režii označování bodů v obrazu:

- Převod obrazu do HSV barevného modelu
- Použití vyhlazení pro odstranění chromatické aberace
- Výpočet jednoho každého pixelu s ohledem na barevnost
- Spuštění detektoru bodů pro nalezení barevných bodů

Pro hledání jiných barev laserových stop je potřeba pouze modifikovat druhý krok. Protože toto řešení nebylo dostatečně účinné, přidal jsem k řešení ještě dva body:

- Threshold červené složky barevného modelu RGB
- Sjednocení bodů vzniklých prvním hledáním a bodů vzniklých druhým hledáním pomocí logického AND

Podobně jako první část lze rozdělit i část druhou na posloupnost kroků a následnou režii s označováním:

- Označení čtyř bodů v obrazu a porovnání s cílovým obrazem pro výpočet transformační matice
- Transformace bodu pomocí transformační matice ze zdrojového obrazu do místa určení

6.2 Ovládání programu

Nejprve je nutné uvést cestu k obrázku, který budeme zpracovávat. Tuto cestu vložíme jako první parametr. Po spuštění programu je nutné inicializovat, což znamená označení části obrazu snímaného kamerou. To se provede označením čtyř bodů. Je nutné označit body v přesném pořadí od levého horního rohu dále přes pravý horní roh, poté levý dolní a nakonec pravý dolní roh. Pokud bude obrázek otočený například vertikálně o 180° doprava, je nutné s touto změnou počítat a stále dodržovat pořadí. To se samozřejmě posouvá, stejně jako obrázek, takže v tomto případě je pořadí opačné.

Pomocí laserových bodů, kterými budeme označovat umístění na zdrojovém obrázku (tj. část obrazu označeném inicializací), můžeme „kreslit“ do cílového obrázku, který jsme uvedli jako první parametr programu. Cílový i zdrojový obrázek musí být stejný.

6.3 Hledání laserových stop v obrazu

6.3.1 Převod obrazu do barevného modelu HSV

Pro převod barevných modelů se v knihovně OpenCV používá funkce CvtColor. Tato funkce je definována jako:

```
1 void cvCvtColor( const CvArr* src, CvArr* dst, int code );
```

src – zdrojový obrázek.

dst – cílový obrázek, který musí být stejný datový typ jako zdrojový obrázek. Počet kanálů může být odlišný.

code – operace barevné konverze, lze specifikovat CV_ <zdrojový_barevný_model>2<cílový_barevný_model>

6.3.2 Použití vyhlazení pro odstranění chromatické aberace

Funkce Smooth, která je taktéž součástí knihovny OpenCV, se používá pro vyhlazení obrazu. Vyhlazením se odstraní chromatická aberace, která může být milně vyhodnocena jako bod zájmu.

```
2 void cvSmooth( const CvArr* src, CvArr* dst,
3                int smoothtype=CV_GAUSSIAN,
4                int param1=3, int param2=0, double param3=0, double
```

```
5                 param4=0 );
```

src – zdrojový obrázek

dst – cílový obrázek

smoothtype – typ vyhlazení, v našem případě CV_BLUR (jednoduché rozostření), se nejvíce hodí. Sumace přes pixely v okolí param1 × param2 s následným škálováním 1 / (param1 • param2).

6.3.3 Výpočet každého jednotlivého pixelu s ohledem na jeho sytost barvy

Pro výpočet hodnoty, jak moc je červený pixel, byla vytvořena funkce SetSaturation, podle vzorce $P = e^{-(h-\mu_h)^2/(2\sigma_h^2)} e^{-(s-\mu_s)^2/(2\sigma_s^2)} e^{-(v-\mu_v)^2/(2\sigma_v^2)}$. Pokud jsou hodnoty v barevném schématu v rozsahu 0…1, kde P je pixel a (h, s, v) je jeho barevný vektor, můžeme vytvořit dva vektory pro hledání červené barvy $\mu = (0,1,1)$ a $\sigma = (0.05,0.22,0.22)$. Při hledání jiné barvy laserových stop musíme tyto dva vektory změnit, například pro barvu zelenou budou tyto dva vektory $\mu = (0.3333,1,1)$ a $\sigma = (0.12,0.5,0.63)$.

```
6 void cvSetSaturation( const CvArr* src, CvArr* dst,
7                 double pureH, double pureS, double pureV,
8                 double sigH, double sigS, double sigV );
```

src – zdrojový obrázek 3- kanálový

dst – cílový obrázek šedotónový

pureX – první vektor, který značí čistou hledanou barvu

sigX – druhý vektor, který značí váhy pro hledanou barvu

6.3.4 Spuštění detektoru bodů pro nalezení barevných bodů

Funkce GoodFeaturesToTrack byla použita pro nalezení bodů zájmu. Nalezené body jsou seřazny podle největší pravděpodobnosti, že daný bod je právě ten hledaný.

```
9 void cvGoodFeaturesToTrack( const CvArr* image, CvArr* eig_image,
10                CvArr* temp_image, CvPoint2D32f* corners,
11                int* corner_count, double quality_level,
12                double min_distance, const CvArr* mask=NULL,
13                int block_size=3, int use_harris=0, double k=0.04 );
```

image – zdrojový obrázek

eig_image – dočasný obrázek stejné velikosti a datového typu jako image

temp_image – druhý dočasný obrázek stejné velikosti a datového typu jako eig_image

corners – výstupní parametr, detekované body

corner_count – výstupní parametr, počet detekovaných bodů

quality_level – násobitel pro maxmin eigenvalue, určuje minimální přijatelnou kvalitu bodů zájmu

min_distance – limitní vzdálenost mezi nalezenými body (nejsou zde použity Euklidovy vzdálenosti)

mask – oblast zájmu (pokud je maska NULL, funkce vyhledá body zájmu v celém obrázku, jinak vyhledá body v určité oblasti)

block_size – velikost bloku k průměrování, v závislosti na použité funkci cvCornerMinEigenVal nebo cvCornerHarris

use_harris – je-li nenulový, použije se Harrisův operátor (cvCornerHarris) místo výchozího cvCornerMinEigenVal

k – volný parametr Harrisova detektoru, použit pouze, pokud je zapnut Hartus

6.3.5 Extrahování červené složky z barevného modelu RGB

V OpenCV jsou složky barevného modelu přehozené do pořadí BGR a proto se extrahuje třetí složka. Cílové obrázky jsou jednokanálové.

```
14 void cvSplit( const CvArr* src, CvArr* dst0, CvArr* dst1,
15               CvArr* dst2, CvArr* dst3 );
```

src – zdrojový obrázek

dst0…dst3 – cílové obrázky

6.3.6 Threshold červené složky

```
16 void cvThreshold( const CvArr* src, CvArr* dst, double threshold,
17                   double max_value, int threshold_type );
```

src – zdrojový obrázek

dst – cílový obrázek

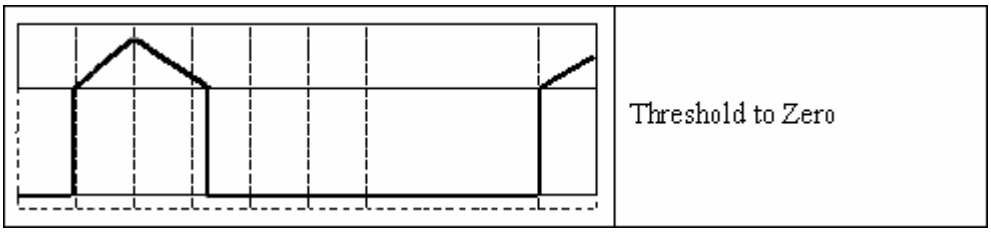
threshold – výška prahu

max_value – používá se u jiných prahových typů, než který je použit v programu

threshold_type – prahový typ, konkrétně CV_THRESH_TOZERO

$$dst(x, y) = \begin{cases} src(x, y), \text{if } src(x, y) > threshold \\ 0, \text{ jinak} \end{cases}$$

Nejlépe tento typ charakterizuje obrázek 2, nižší hodnoty než prahové se vymažou, vyšší zůstávají.



Obrázek 4. Metoda Threshold to Zero.

6.3.7 Sjednocení bodů pomocí logického AND

Pro sjednocení obrázků byla vytvořena funkce MergeDown. Tato funkce pomocí logické funkce AND a dvou různých vah vypočítá ze dvou zdrojových obrázků jeden cílový. První váha je 0,6 pro první obrázek a 0,4 pro druhý obrázek, tím je dán i vstup, kde první bude obrázek s body po thresholdu a druhý vypočítaný pomocí funkce SetSaturation.

```
18 void cvMergeDown( const IplImage* src1, IplImage* src2,
19                    IplImage* dst )
```

src1 – zdrojový obrázek s váhou 0,6

src2 – zdrojový obrázek s váhou 0,4

dst – cílový obrázek

6.4 Výpočet bodu, který byl označen laserem, a jeho perspektivní transformace do místa určení

6.4.1 Označení čtyř bodů ve zdrojovém obrazu

Tato metoda se přidá k oknu, ve kterém se používá klikání. V této metodě se vypočtou souřadnice kliknutí. Body tímto způsobem získané se v obrazu označí zeleným bodem. Po čtvrtém kliknutí je inicializace ukončena.

```
20 void on_mouse(int event, int x, int y, int flags, void* param)
```

event – událost, např. kliknutí levým tlačítkem CV_EVENT_LBUTTONDOWN

x – x-ová souřadnice

y – y-ová souřadnice

6.4.2 Výpočet transformační matice

Body získané označením metodou on_mouse se použijí jako zdrojové souřadnice. Cílové souřadnice se vypočítají pomocí rozměrů vstupního obrázku.

```
21 cvPerspective( const IplImage* src )
```

src – zdrojový obrázek, který může mít libovolné rozměry

Zdrojové a cílové body se vloží do funkce cvGetPerspectiveTransform, která slouží pro výpočet transformační matice.

```
22 CvMat* cvGetPerspectiveTransform( const CvPoint2D32f* src,
23               const CvPoint2D32f* dst, CvMat* map_matrix );
```

src – souřadnice vrcholů čtyřúhelníku ve zdrojovém obrazu

dst – souřadnice vrcholů odpovídajícího čtyřúhelníku v cílovém obrazu

map_matrix – ukazatel na transformační matici 3x3

6.4.3 Transformace bodu

Dosazení bodu, určeného laserovým bodem, do rovnice s koeficienty z transformační matice, kde x a y jsou zdrojové body a u a v cílové body.

$$u = \frac{Ax + By + C}{Gx + Hy + I}, \qquad v = \frac{Dx + Ey + F}{Gx + Hy + I} \tag{22}$$

Vykreslení bodu do cílového obrázku probíhá pomocí režie.

6.5 Rozšiřitelnost

Menší úpravou by se dalo využít tento program, který používá detekci laserových stop v obrazu a jejich transformaci ze snímaného obrazu do obrazu v počítači, například pro ovládání myši laserovým ukazovátkem nebo kreslení poznámek do obrazu. Obojí je možné využívat při promítání prezentací. Program by musel běžet na pozadí a snímaná plocha kamerou by byl výstup z projektoru.

ZÁVĚR

Díky tomu, že konstrukce knihovny OpenCV byla již od počátku vyvíjena pro sestavování pomocí standardních kompilátorů a také že byla vyvíjena v programovacím jazyku C++, si uchovala přenositelnost mezi různými zařízeními a operačními systémy, včetně embedded zařízení. Výpočetní výkon těchto zařízení ale nebývá velký, proto je vhodné využít jiné řešení. Přenositelnosti knihovny se věnuji v oddílu 1.1.3 Přenositelnost.

Je možné použít osobní počítač pro zpracování obrazu z embedded zařízení, které bude připojeno pomocí rychlého připojení např. kabelem, či vysokorychlostním bezdrátovým přenosem wi-fi. Tyto přenosy musí zvládat velké objemy multimediálních dat tzn. obrazu. Obraz z embedded zařízení se odešle do osobního počítače, kde se provedou nutné operace s obrazem, a dále se zpracovaný obraz odešle zpět do embedded zařízení.

Existují knihovny, které jsou primárně určené pro zpracování obrazu ve specifických zařízeních, jako jsou mobilní telefony, které mají omezenou výpočetní kapacitu. Tyto knihovny jsou od základu vyvíjeny se specifickým určením. Použití je zatím velmi malé vzhledem k výpočetní kapacitě, ale nejčastější použití je v zábavních aplikacích.

V praktické části jsem se věnoval konstrukci ukázkového program za použití knihovny OpenCV. Program byl zkonstruován v programovacím jazyku C++ a má demonstrovat schopnosti knihovny počítačového vidění. Program detekuje laserové body v obrazu snímaném kamerou. Po označení části obrazu těmito body, tzn. inicializaci, je možné zakreslovat do obrázku otevřeného v počítači. Menší úpravou by se dalo toto využít například pro ovládání myši laserovým ukazovátkem nebo kreslení poznámek do obrazu. Obojí je možné využívat při promítání prezentací.

ZÁVĚR V ANGLIČTINĚ

Thanks to the fact that the construction of OpenCV library was developed from the outset to compile with the standard compiler, and was developed in the programming language C++, you keep the portability between different devices and operating systems, including embedded devices, but the computing power of these devices is not large, so it is appropriate to use a different solution. I aim my attention to the portability of the library in Section 1.1.3 Portability.

It is possible to use a personal computer for image processing from embedded devices to be connected via high-speed such as cable or wireless transmission of high-speed wi-fi. These transfers must manage large volumes of multimedia data, ie. image. The image of the embedded device is sent to a personal computer, where it is necessary to carry out operations with the image, and the processed image is sent back to the embedded device.

There are libraries that are primarily designed for image processing in specific devices such as mobile phones, which have limited computing capacity. These libraries are completely developed with a specific purpose. Use is still very small due to the computational capacity, but the most common use is in entertainment applications.

In the practical part, I aimed my attention to the construction of demonstration program using OpenCV libraries. The program has been designed in a programming language C++ and has demonstrated the ability of computer vision library. The program detects the laser points in an image taken by camera. After selecting the part of the image by these points, i. e. initialization, there is possible to draw the image open on your computer. This could be used by minor editing, for example, to control the mouse laser pointer or drawing notes and it can be used in projection presentations.

SEZNAM POUŽITÉ LITERATURY

[1] BRADSKI, Gary. KAEHLER, Adrian. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly, 2008. 555s. ISBN 9780596516130.

[2] ŽÁRA, Jiří. BENEŠ, Bedřich. SOCHOR, Jiří. FELKEL, Petr. Moderní počítačová grafika (2. vydání). Computer Press, 2005. 609s. ISBN 80-251-0454-0.

[3] HARRIS, Chris. STEPHENS, Mike. A combined corner and edge detektor. Proceedings of The Fourth Alvey Vision Konference. University of Manchester, England. 1988. p. 147-151.

[4] SONKA, Milan. HLAVAC, Vaclav. BOYLE, Roger. Image Processing, Analysis, and Machina Vision Second Edition. PWS - an Imprint of Brooks and Cole Publishing, 1998. 770s. ISBN 9780534953935.

[5] HECKBERT, Paul. Fundamentals of Texture Mapping and Image Warping. University of California at Berkeley, 1989. 86s. Master's thesis. Dept. Of Electrical Engineering and Computer Science. University of Kalifornia Berkeley. Under the direction of Carlo Séquin.

[6] GONZALEZ, Raphael. WOODS, Richard. Digital Image Processing, , 2nd ed. Prentice Hall Press, 2002. 295s. ISBN 0-201-18075-8.

[7] PRATA, Stephan. Mistrovství v C++, 2. aktualizované vydání. Computer Press, 2004. 295s. ISBN: 80-251-0098-7.

[8] DUDEK, Gregory. RICHARD, Michael. JENKIN, MacLean. JENKIN, Michael. Computational Principles of Mobile Robotic. Cambridge University Press, 2000. 280 s. ISBN 0521568765.

[9] *Corner detection* [online]. [cit. 2009-04-05]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Corner_detection>.

[10] *Finding Laser dot* [online]. [cit. 2009-01-18]. Dostupný z WWW: <http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Finding_a_Laser_Dot >.

[11] BADAWI, Sami. *Computer Vision C++ libraries review* [online]. 17.11.2008 [cit. 2009-03-25]. Dostupný z WWW:

<http://samibadawi.blogspot.com/2008/11/computer-vision-c-libraries-review.html>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CV	Computer Vision
IPP	Integrated Performance Primitives
MMX	MultiMedia eXtensions
STL	Standard Template Library.
BSD	Berkeley Software Distribution.
GIL	Generic Image Library
VXL	Vision *something* Library
GUI	Graphic user interface
SSD	Sum of squared differences
C++	
δ	Inverzní gradiet

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD

Obsah Přiloženého CD k práci.

- Bakalářská práce v elektronické formě.
- Prezentace bakalářské práce, vytvořená v programu MS Power Point.
- Ukázkový program vytvořený v programovacím jazyku C++, včetně zdrojových kódů.