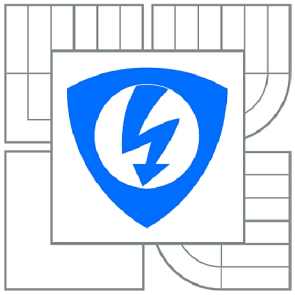
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SLEDOVÁNÍ OBJEKTU VE VIDEOSEKVENCI

OBJECT TRACKING IN VIDEOSEQUENCE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

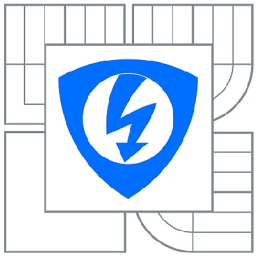
AUTOR PRÁCE — Bc. ZDENĚK NEŠPOR

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR ČÍKA, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Zdeněk Nešpor ***ID:*** 77667
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Sledování objektu ve videosekvenci

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte algoritmy pro sledování předem definovaných objektů ve videosekvenci. Vyberte vhodný způsob pro detekci a sledování předem definovaného objektu a implementujte jej v programovacím jazyce JAVA. Vaši implementaci experimentálně ověřte a zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] BURGER, Wilhelm; BURGE, Mark J. Principles of Digital Image Processing: Fundamental Techniques. Londýn : Springer, 2009. 272 s. ISBN 978-1848001909.
[2] GONZALEZ, Rafael C.; WOODS, Richard E. Digital Image Processing. 3. Londýn : Pearson Pentice Hall, 2008. 954 s. ISBN 978-0-13-505267-9.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Petr Číka, Ph.D.
Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

Abstrakt

Diplomová práce se zabývá problematikou sledování předem definovaného objektu ve videosekvenci. Po krátkém úvodu je popsán postup vhodný k detekci objektu ve videosekvenci, jehož metody jsou dále podrobně rozebrány. Je zde řešena problematika předzpracování obrazu, segmentace obrazu a detekce objektu v obraze. Hlavní důraz je kladen na detektory využívající body zájmů a deskriptory oblastí - SURF a SIFT.

Druhá část práce se zabývá praktickou realizací programu vhodného ke sledování předem definovaného objektu ve videosekvenci. Nejprve jsou analyzovány knihovny vhodné ke sledování objektu ve videosekvenci v prostředí jazyku JAVA, následuje podrobný popis vybrané knihovny OpenCV spolu s wrapperem JavaCV. Dále je popsána vlastní aplikace z hlediska ovládání a funkčnosti, jsou popsány klíčové metody. Výstupy aplikace spolu s diskusí a zhodnocením jsou prezentovány na konci práce.

Abstract

This thesis deals with tracking a predefined object in the movie. After a brief introduction describes the procedure suitable for the detection of an object in a video sequence, where the methods are also discussed in detail. There is dealt with issues of image preprocessing, image segmentation and object detection in the image. The main emphasis is laid on using detectors of interest points and descriptors of areas - SURF and SIFT.

The second part deals with the practical implementation of a program suitable to monitor predefined object in the movie. First are analyzed libraries suitable for object tracking in a video sequence in an environment of Java, followed by a detailed description of the selected library OpenCV along with wrapper JavaCV. Further described is own application in terms of control and functionality are described key method. Outputs along with discussion and evaluation are presented at the end of work.

Klíčová slova

detekce objektu, sledování objektu, SIFT, SURF, Java, OpenCV, JavaCV

Keywords

object detection, object tracking, SIFT, SURF, Java, OpenCV, JavaCV

NEŠPOR, Z. *Sledování objektu ve videosekvenci.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 46 s. Vedoucí diplomové práce Ing. Petr Číka, Ph.D..

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Sledování objektu ve videosekvenci jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Břeclavi dne 27.05.2013

podpis autora

Obsah

Seznam obrázků

1 Úvod

Tato práce se zabývá algoritmy pro sledování předem definovaných objektů ve videosekvenci. Jejími cíli bylo:

- prostudování algoritmů pro sledování předem definovaných objektů ve videosekvenci,
- vybrání vhodného způsobu pro detekci a sledování předem definovaného objektu,
- implementace vybraného způsobu v jazyce JAVA,
- experimentální ověření a zhodnocení výsledků.

Tato práce je organizována tak, že po krátkém úvodu do problematiky následuje seznámení s jednotlivými kroky detekce objektu ve videosekvenci a s možnými požadavky, které kladou řešené úlohy z této oblasti na výběr použité metody pro jejich řešení. Poté jsou již probírány jednotlivé kroky detekce objektu, kdy je nejprve probíráno předzpracování obrazu vstupujícího do procesu detekce, a to konkrétně způsoby potlačení šumu a zvýraznění hran.

V další části je rozebírána problematika segmentace obrazu, po níž jsou v následující části rozebírány některé možné způsoby detekce objektu ve scéně. Zde je kladen hlavní důraz na detektory využívající bodů zájmů a deskriptorů oblastí. V této části jsou podrobně popsány metody SIFT a SURF, neboť SURF je využita v další části práce.

V sedmé kapitole jsou diskutovány možnosti sledování objektu ve videosekvenci v prostředí jazyka JAVA z hlediska výběru vhodných knihoven. Zde je také představena knihovna OpenCV spolu s wrapperem JavaCV.

Osmá kapitola se zabývá vytvořenou aplikací, popisem její činnosti a jejími klíčovými funkcemi. Její výstupy jsou pak diskutovány v kapitole deváté.

V závěrečné kapitole jsou pak shrnuty výsledky práce a zhodnoceny rozdíly ve výstupu práce a cíli a očekáváními na začátku projektu.

2 Úvod do problematiky

Sledování objektu je jednou z úloh z oblasti počítačového vidění. Počítačové vidění (z anglického Computer Vision) je obor, který se zabývá metodami získávání, zpracování, analýzy a porozumění obrazům. Rozvoj počítačového vidění je umožněn obzvláště rozvojem výpočetní kapacity procesorů a dalších komponent, jež dokáží zpracovávat velké množství obrazových dat.

Videosekvence je řada po sobě jdoucích snímků. Pro plynulé zobrazení pohybu jsou užívány snímkové frekvence minimálně 25-30 snímků za sekundu. Snímek se skládá z jednotlivých obrazových bodů – pixelů. Zatímco člověk v těchto pixelech na základě dříve získaných znalostí rozeznává jednotlivé objekty a dokáže je identifikovat, stroje mají s tímto úkolem problém. Řešením tohoto problému se zabývá právě počítačové vidění.

Sledování objektů má mnoho aplikací v praxi. Jedním z nich je automatický dohled nad silničním provozem. Pomocí sledování objektu je možno za využití kamer vykonávajících dohled nad silničním provozem sledovat vozidlo, které porušilo dopravní předpisy. Další aplikací je automatické řízení strojů, ať už jde o řízení robotů (např. sondy pohybující se po planetě Mars), nebo o stroj umisťující integrované obvody do desek plošných spojů. Další příklady využití jsou uvedeny v lit. [2], [14].

3 Sledování objektu ve videosekvenci

Sledování objektů ve videosekvenci v sobě zahrnuje několik po sobě jdoucích kroků. Těmito kroky jsou:

a) získání digitálního obrazu – obraz je nejčastěji získáván pomocí kamer. U těchto je užitečné vědět jejich vlastnosti, které do obrazu vnášejí určité zkreslení (např. u širokoúhlých kamer je to druh geometrického zkreslení). Tato problematika není v této práci rozebírána, avšak podrobnější informace je možno čerpat v lit. [5].

b) předzpracování obrazových dat – zahrnuje přípravu sekvence pro detekci objektů.

c) segmentace – je proces identifikace složek obrazu. Cílem procesu segmentace je odlišit objekty ve scéně od pozadí.

d) detekce daného objektu ve scéně – na základě určitých charakteristických vlastností objektu je ve scéně tento objekt vyhledán.

Jednotlivé stupně zpracování obrazových dat při sledování objektu ve videosekvenci budou rozebrány dále v práci.

Předtím je však nutno si uvědomit, že jednotlivé druhy aplikací procesu sledování objektu mají různé požadavky a omezení pro tuto úlohu. Těmito jsou dle [14] např.:

- čas zpracování – v mnoha případech je nutné, aby obrazová data byly zpracovány a vyhodnoceny v reálném čase.
- přesnost – některé aplikace vyžadují, aby pozice objektu byla detekována velmi přesně.
- spolehlivost rozpoznání objektu – v některých aplikacích je tento požadavek velmi důležitým. Při tom rozlišujeme dva druhy chyb – buď je správný objekt označen chybně jako nevyhovující anebo je nevyhovující objekt označen jako správný.
- invariance – v závislosti na aplikaci je žádoucí dosáhnout invariance v souvislosti s následujícími jevy:
  a) vliv osvětlení – vzhled objektů se mění v závislosti na síle, úhlu nebo barvě osvětlení objektu. Hledaný objekt v obraze by měl být rozpoznán nezávisle na vlivu osvětlení a jeho změn.

b) vliv měřítka – v případě, že hledáme předem daný objekt v obraze, narážíme na problém vzdálenosti objektu od zařízení, kterým získáváme obraz. Důvodem je fakt, že prezentace jednoho a toho samého objektu snímaného v různé vzdálenosti od kamery se od sebe liší.

c) rotace – pokud dojde k rotaci objektu, tato skutečnost by měla být detekována a sledovaný objekt by měl být správně detekován.

d) nepořádek v pozadí – proces rozpoznávání by neměl být ovlivněn rozmanitostí pozadí.

e) částečný výskyt – v některých případech se sledovaný objekt nenachází celý ve scéně, případně mohou být části sledovaného objektu zakryty jiným objektem.

f) změna úhlu pohledu – vzhledem k tomu, že při získávání videosekvence dochází k projekci 3D scény do 2D obrazu a tudíž i ke ztrátě jednoho rozměru. Proto dvourozměrný vzhled objektu závisí silně na relativní pozici kamery vůči objektu.

Jaké požadavky budou konkrétní aplikací vyžadovány závisí na konkrétním zadání úlohy. Proto také problematika výběru vhodné metody závisí na tomto konkrétním zadání úlohy a univerzální metoda, která by byla nejvhodnější pro všechny typy úloh neexistuje.

4 Předzpracování videosekvence

Po získání obrazových dat v digitální podobě je předtím, než je obraz zpracováván, je provedeno předzpracování těchto dat. Předzpracování videosekvence není nutné ve všech případech. Provedením těchto operací lze zlepšit výsledek detekce objektu. Na druhou stranu je nutno podotknout, že nevhodným nasazením těchto technik podstatně zhoršíme detekci objektů v obraze. Pro tyto účely se používají různé filtry. Filtrem se rozumí operace, kdy se pro výpočet každého nového pixelu použije více než jeden pixel obrazu původního. Použití konkrétní metody předzpracování obrazu pak závisí na konkrétním případě dle řešené úlohy. V dalším textu popíšeme dvě z možných technik předzpracování obrazu, a to potlačení šumu a zvýraznění hran.

4.1 Potlačení šumu

Šumem rozumíme nechtěnou součást obrazu. Některé druhy šumu jsou vyjmenovány v lit. [6]. Těmito šumy jsou:

- bílý šum – jedná se o idealizovaný šum, kdy se jedná o náhodný signál s rovnoměrnou výkonovou spektrální hustotou.
- Gaussův šum – je součástí téměř každého signálu. Jedná se o statistický šum, jehož hustota rozdělení pravděpodobnosti je rovna normálnímu rozdělení.
- aditivní – vznikající při přenosu obrazu nebo snímání, tento šum se přičítá k původnímu obrazu.
- Pepř a sůl – v podobě černých a bílých bodů v obraze, příkladem, kde může tento druh šumu vzniknout je přenos obrazu přes zašuměnou digitální linku.
- impulsní šum – v podobě zrnění u obrazů s více jasovými úrovněmi, dochází k nahrazení pixelů z původního obrazu.
- kvantizační – dochází k němu při převodu obrazu do digitální podoby.

K potlačení šumu se užívají filtry pro vyhlazování obrazu. Při tomto však dochází k potlačení vyšších frekvencí a tudíž i k potlačení jiných náhlých změn v obraze, jako jsou ostré čáry a hrany [6].

Rozlišujeme filtry lineární a nelineární. Lineární filtry mají často charakter dolní propusti. Výpočet může být proveden pomocí konvoluce s maskou. Rozměry masky mohou být libovolné a platí, že čím větší maska, tím vice pixelů se podílí na pixelu nového obrazu. S rostoucí velikostí masky taktéž dochází k většímu rozmazání obrazu. Mezi lineární filtry řadíme např.:

a) průměrování – hodnota každého pixelu nového obrazu je určena průměrem hodnoty tohoto pixelu a pixelů v jeho nejbližším okolí. Jedná se o nejjednodušší typ filtru. Je efektivní k potlačení Gaussova šumu. Příklad masky filtru tohoto typu je uvedena v lit. [2] jako

$$H(i,j)=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{4.1}$$

b) Gaussův filtr – jedná se o masku, jejíž prvky jsou určeny Gaussovou funkcí. Tato funkce uvedená v lit. [2] odpovídá 2D Gaussově funkci

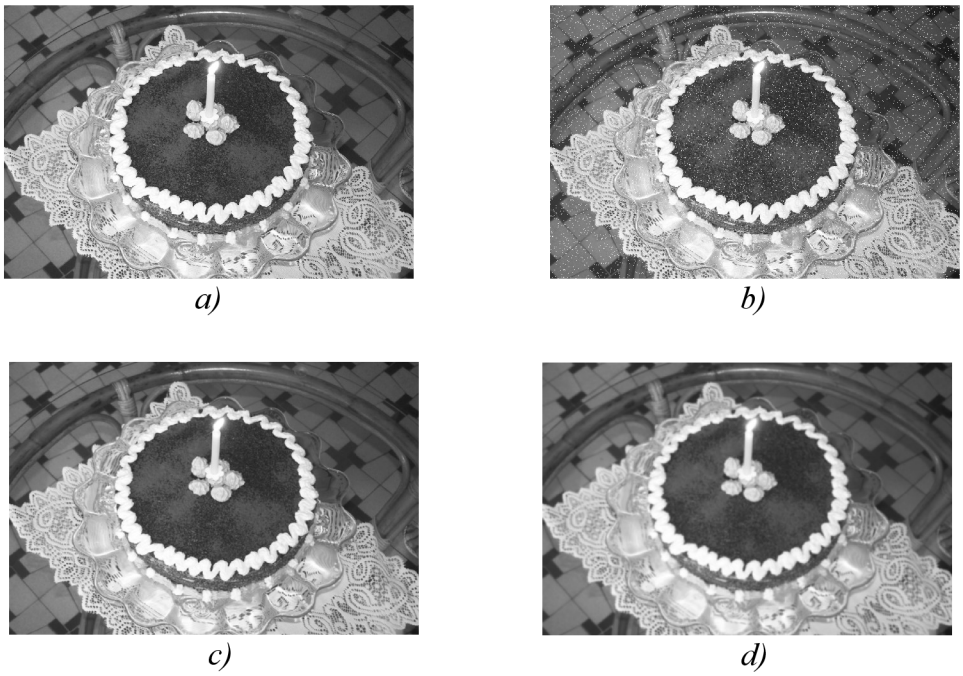
$$G_\sigma(x,y)=\mathrm{e}^{\frac{-(x^2+y^2)}{(2\sigma^2)}}, \tag{4.2}$$

kde σ značí standardní odchylku a x,y značí vzdálenost od středu.

V tomto případě středový pixel získá maximální váhu a zbývající koeficienty klesají postupně se zvyšující se vzdáleností od středu [2].

Mezi nelineární filtry patří mediánový filtr. Principem tohoto filtru je, že se vezmou hodnoty pixelu v okolí zpracovávaného pixelu. Tyto hodnoty jsou pak seřazeny a následně je vybrána hodnota uprostřed (medián). Tento filtr je vhodný k odstranění náhodného šumu. Výhodou filtru je, že redukuje rozmazávání hran. Jeho nevýhodou naproti tomu je, že poškozuje tenké čáry a ořezává ostré rohy [2].

Na obrázcích výše jsou uvedeny vedle originálního obrazu také ukázka bílého šumu v obraze a ukázka dvou lineárních filtrů – průměrného a Gaussova.



a) *b)*

c) *d)*

Obrázek 4.1 : Ukázky šumu a použití filtrů: a) originální obrázek, b)bílý šum v obraze, c) filtr průměrování, d) Gaussův filtr

Další typy filtrů a jejich podrobnější popis lze nalézt v lit. [6].

4.2 Zvýraznění hran

Jestliže některé koeficienty filtru jsou záporné, výpočet filtru může být interpretován jako rozdíl dvou součtů: vážený součet všech pixelů s pozitivními koeficienty minus vážená suma všech pixelů se zápornými koeficienty [2]. Tento druh filtru se nazývá diferenčním filtrem. Příkladem může být filtr Laplacian of Gaussian, jeho maska o rozměru 5x5 je uvedena v lit. [2] jako

$$H(i,j)=\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{4.3}$$

Výhodou tohoto filtru je, že změny místní intenzity jsou zvýrazněny. Z tohoto důvodu jsou tyto filtry uplatňovány při detekci hran a ostření obrazu.

5 Segmentace obrazu

Jak již bylo řečeno, cíle segmentace obrazu je oddělit objekty ve scéně od pozadí. Přesněji řečeno slouží k automatickému rozdělení obrazu na oblasti se společnými vlastnostmi a které mají smysluplný význam. Tímto krokem zároveň redukujeme množství dat, které je nutno zpracovávat v dalším kroku, detekci daného objektu. Jedním z problémů segmentace je fakt, že různé metody nebo metoda s různými parametry dávají různé výsledky [6]. V literatuře [6] je dále uvedeno následující dělení metod segmentace:

1) segmentace prahováním
2) segmentace na základě detekce hran
3) segmentace narůstáním oblastí
4) segmentace srovnáním se vzorem

Jednotlivými metodami se bude zabývat následující část práce.

5.1 Segmentace prahováním

Jde o nejjednodušší metodu prahování. Základem metody je hodnocení jasu každého pixelu. Tuto metodu můžeme použít, pokud lze najít v histogramu obrazu takovou hodnotu

prahu, že všechny hodnoty jasu nižší než práh odpovídají pozadí a hodnoty vyšší než hodnota prahu odpovídají popředí (objektu či objektům).

Matematicky lze toto popsat pomocí rovnice

$$f(i,j) = 1 \quad \text{je-li } g(i,j) \geq T \qquad (5.1)$$

$$0 \quad \text{jinak}$$

kde $f(i,j)$ je pixel na pozici i,j transformovaného obrazu, $g(i,j)$ je pixel na pozici i,j zpracovávaného obrazu a T je práh.

Pokud však histogram obsahuje různé úseky hodnot jasu, nelze určit globální práh přímo. Tohoto lze však dosáhnout pomocí metod adaptivního prahování [10].

Tuto metodu však nelze použít, pokud rozložení hodnot jasu v histogramu pro pozadí i popředí se překrývá. Tato situace může být způsobena, pokud je obraz výrazně zašuměný.

5.2 Segmentace na základě detekce hran

Hrany mohou být popsány jako místa v obraze, kde dochází k prudkým změnám jasu podél určitého směru. Čím je změna jasu v místě větší, tím je pravděpodobnější, že se zde nachází hrana. Hrany mohou, ale nemusí ohraničovat objekty ve scéně. Detekce hran je velmi náročným procesem jež může být ovlivněn např. šumem, rozostřením obrazu atd. V případě nekvalitního obrazového materiálu mohou být hrany detekovány na pozicích, kde se nenachází. Další možností je, že v místě, kde se nachází hrana, není tato detekována. Z tohoto důvodu je většinou nutno obrazové data předupravit filtrací, jak již bylo zmiňováno v předchozí kapitole. Pro detekci hran se užívají hranové detektory. Po detekci hran jsou některé hrany rozpojeny a je nutno tyto hrany spolu propojit. Nejprve však probereme detektory hran.

5.2.1 Detektory hran

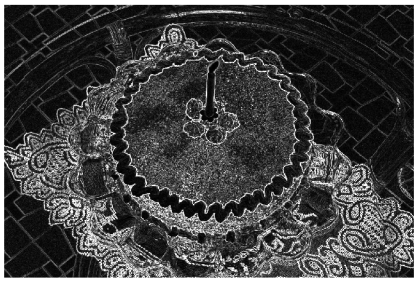
Vzhledem k tomu, že se hrana nachází v místě, kde dochází k velké změně jasu, bude v tomto místě i velká hodnota derivace jasové funkce [3]. Přičemž ve směru kolmo na hranu bude hodnota derivace maximální. Pro zjednodušení výpočtu jsou hrany detekovány pouze ve dvou, příp. čtyřech směrech [3]. V mnoha případech je derivace jasové funkce aproximována pomocí konvoluce s vhodným jádrem.

Podle způsobu, kterým jsou hrany vyhledávány, můžeme rozdělit detektory hran na detektory založené na:

1) první derivaci – hledá se gradient a zjišťuje se, jestli je dostatečně velký. Díky tomuto jsou taktéž nazývány metodami gradientními. Pro účely nalezení hran jsou používány následující konvoluční jádra:

   a) Robertsova konvoluční jádra – tento operátor o velikosti 2x2 je citlivý na šum, protože odečítá blízké pixely. Je to dáno faktem, že okolí použité pro aproximaci je malé. Odpovídá derivaci podle šikmých směrů. Konvoluční masky jsou:

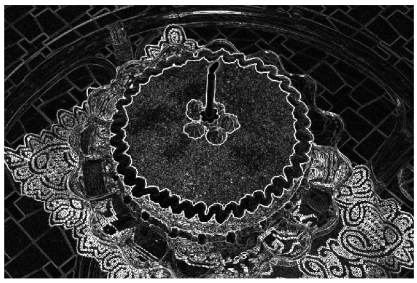
$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{5.2}$$



Obrázek 5.1: Detekce hran pomocí Robertsova operátoru

   b) Konvoluční jádra Prewittové – operátor je definován takto

$$h_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \quad h_3 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \tag{5.3}$$



Obrázek 5.2: Detekce hran pomocí operátoru Prewittové

c) Sobelova konvoluční jádra – operátor je definován následovně

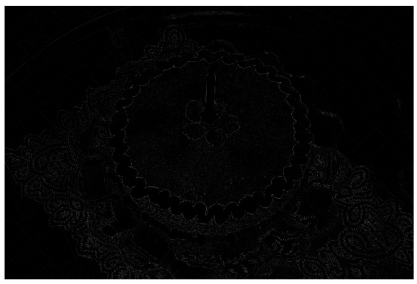
$$h_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad h_3 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{5.4}$$

Tento operátor preferuje středové body a méně body v okolí. Je to kombinace Gaussova filtru a diferenciálu. U tohoto operátoru (a taktéž i u operátoru Prewittové) lze s úspěchem užít separability (operátor se rozloží na konvoluci dvou 1D jader), a to z důvodu větší efektivity výpočtu. Výpočet konvoluce se dvěma 1D jádry je časově mnohem méně náročný než s jedním 2D jádrem, což platí pro všechna jádra o velikosti větší než 2x2. Nedostatkem metody je, že vytváří tlusté hrany [11].

Obrázek 5.3: Detekce hran pomocí Sobelova operátoru

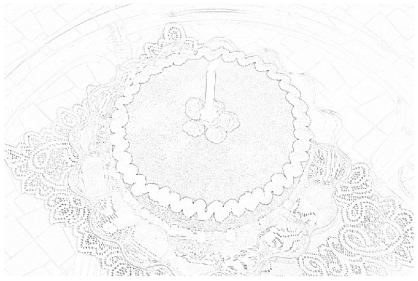
Tento operátor se často užívá pro detekci vodorovných a svislých hran. K této operaci jsou zapotřebí pouze operátory h_1 a h_3.

2) druhé derivaci - druhá derivace zjišťuje, kde ten gradient nejméně mění svůj směr a takové místo se označí jako hrana. Je to místo, kde druhá derivace obrazové funkce prochází nulou a kde první derivace dosahuje lokálního maxima. Užívá se v případech, kdy nepotřebujeme nutně znát směr a velikost hran [11]. Pro tyto účely je možno použít buď filtr Laplacian of Gaussian (bývá označován jako LoG) nebo Difference od Gaussian (tento bývá označován zkratkou DoG).



a)

b)



c)

Obrázek 5.4: Použití hranových operátorů využívajících druhé derivace: a) detekce hran pomocí Laplaceova operátoru, b) Detekce hran pomocí operátoru Difference of Gaussian – neinvertovaný, c) invertovaný obrázek c)

Mezi nejznámější hranové detektory patří Cannyho hranový detektor. Tento detektor vytvořil v roce 1986 J. F. Canny. Základem Cannyho práce bylo matematické vyjádření tří hlavních cílů:

1. nízká chybovost, což znamená, že musí být nalezeny všechny hrany, kdy nepravé hrany nesmí být označeny jako hrany,
2. dobrá lokalizace hran, kdy lokalizované hrany musí být co nejblíže skutečným hranám,
3. jednoznačná odezva, tedy detektor má vrátit pouze jeden detekovaný bod hrany ke každému jednomu bodu hrany. Toto znamená, že detektor nemá detekovat vice bodů hrany v místě, kde hrana má pouze jeden bod..

Tyto cíle jsou však protichůdné, neboť platí, že čím lepší lokalizace se budeme snažit dosáhnout, tím více nalezneme hran, které ve skutečnosti hranami nejsou.

Detekce hran dle Cannyho v sobě zahrnuje několik kroků. Po odstranění šumu např. Gaussovým filtrem jsou detekovány hrany v obraze gradientní metodou, nejlépe Sobelovým operátorem, neboť tento není citlivý na šum. Nalezené hrany jsou poté tenčeny,

kdy jsou z nalezených gradientů odebrány body, které nejsou lokálním maximem. Pak je provedeno prahování s hysterezí. Při tomto jsou použity dva prahy, minimální a maximální. Následně jsou body, které jsou označeny jako hrana posuzovány tímto způsobem:

- pokud je hodnota gradientu posuzovaného bodu menší než minimální práh, je tento bod označen, že není hranou.
- pokud je hodnota gradientu posuzovaného bodu větší než maximální práh, je tento bod označen, že je hranou.
- pokud je hodnota gradientu posuzovaného bodu mezi minimálním a maximálním prahem, je tento bod označen, že je hranou pouze v případě, že sousedí s bodem, který již byl označen jako hrana. V opačném případě je označen, že není hranou.

Tímto odstraníme krátké řetězy hranových bodů, aniž bychom fragmentovali dlouhé řetězy hranových bodů [5].

5.2.2 Zpracování obrazu s detekovanými hranami

Po detekování hran je nutno ještě tyto hrany, které jsou rozpojeny, pospojovat. Lze použít několik metod, kterými jsou:

1. sledování hranice – užívá se v případě, kdy kdy není znám tvar hranice, ale jsou v obraze určeny oblasti. Cílem je určit vnitřní hranice pro každou oblast v obraze [4].
2. heuristické vyhledávání hranice – využívá postupů prohledávání grafů. Hrany jsou spojovány do řetězů lépe odpovídající průběhu hranic [6].

5.3 Segmentace narůstáním oblastí

U těchto metod se budeme snažit o rozčlenění obrazu do maximálních souvislých homogenních (z hlediska zvoleného parametru) oblastí. Kritérium homogenity mohou být jasové vlastnosti, textura nebo barva [6]. Výhodou těchto metod je jejich odolnost pro šumu, kdy tyto metody u velmi zašuměných obrazů vrací lepší výsledky než metody detekce hran. U tohoto druhu segmentace obrazu můžeme použít jednu ze tří metod uvedených v lit. [6], a to:

1. metoda spojování oblastí – u této metody je na začátku obraz rozdělen do velkého

množství malých oblastí. Pak je definováno kritérium pro spojování dvou sousedních oblastí. Poté jsou spojovány sousední oblasti vyhovující zvolenému kritériu a to až do doby, kdy již nelze spojit žádné dvě oblasti bez porušení kriteria [6]. Výsledky metody spojování oblastí se liší pro různé definice počátečních oblastí, kritéria, počátky spojování, pořadí předpokládaných oblastí, postupy spojování.

2. metoda štěpení oblastí – u této metody je na začátku jediná oblast, která odpovídá celému obrazu. Tato oblast je pak dělena, dokud podoblasti nesplňují kritérium [9]. Tak jako u předchozí metody, je při jejím užití dosahováno různých výsledků.
3. metoda štěpení a spojování – u této metody je štěpení a spojování realizováno v rámci čtvercových oblastí pyramidální datové struktury, jenž musí být (stejně jako kritérium homogenity) nadefinována před vlastní segmentací. Mohou nastat při tomto dva případy:
   a) oblast v dané úrovni pyramidy je nehomogenní – v tomto případě je oblast rozdělena na čtyři podoblasti,
   b) oblasti jsou navzájem homogenní – v tomto případě dojde ke spojení oblastí do jedné oblasti.

5.4 Segmentace srovnáváním se vzorem

Při tomto způsobu segmentace vyhledáváme známé objekty v obraze na základě srovnání se vzorem. U tohoto způsobu segmentace hledáme maximu vhodného kritéria pro všechny možné transformace obrazu (natočení, změna měřítka, zkreslení atd.). Při tom používáme různé metody, např. uvedené v lit. [6]:

- vzájemná korelace,
- srovnávání grafů vytvořených podle vlastností sledovaných objektů,
- hledání shody částí obrazu s jen jednotlivými částmi vzoru.

6 Detekce objektu ve scéně

Poté, co jsme provedli segmentaci obrazu, můžeme přistoupit k nalezení sledovaného objektu ve videosekvenci. Popis některých metod k nalezení daného objektu bude obsahem

následující kapitoly. Bližší popis těchto metod, včetně pseudokódů, je možné nalézt v lit. [14].

6.1 Globální metody

U těchto metod je model rozpoznáván jako celek. Objekt je charakterizován souborem dat obsahujícím několik globálních charakteristik, kterými mohou být plocha, obvod apod. U těchto metod je využíván vzor hledaného objektu.

6.1.1 2D korelace

Tato metoda přistupuje k problému rozpoznání objektu v obraze prostřednictvím 2D křížové korelace obrazu se vzorovou reprezentací objektu.

Principem metody je, že spočítán normalizovaný koeficient křížové korelace pixel po pixelu obrazu se vzorem. Potom každé lokální maximum této funkce označuje možný výskyt hledaného objektu. Jestliže hodnota maxima přesáhne určitý prahovou hodnotu, znamená to, že byl nalezen objekt v obraze. Jeho pozice je určena pozicí maxima.

Je snadno implementovatelná díky své jednoduchosti, avšak má několik nevýhod. Těmito nevýhodami jsou:

- není invariantní vůči rotaci objektu a změně měřítka,
- je náchylné na nelineární změny osvětlení,
- je citlivé na změť (angl. Clutter) a zákryt (angl. Occlusion)

Tato metoda není vhodná pro rozpoznávání s větším množstvím vzorů. Z tohoto důvodu byly vymyšleny varianty této metody, a to:

- předzpracování – vzor i obraz jsou před rozpoznávací fází zpracovány detektorem hran a korelace je počítána z filtrovaných obrazů. Tímto se zvýší odolnost proti nelineárním změnám osvětlení v obraze a také dojde ke zvýšení přesnosti rozpoznání pozice hledaného objektu.
- podvzorkování/ pyramida obrazů – obraz je postupně podvzorkován na určitou výchozí úroveň. Na této úrovni je v podvzorkovaném obraze vyhledáván vzor. Poté se pokračuje na vyšší úrovni v oblasti, kde by se mohl vyskytovat hledaný objekt.

Takto se pokračuje, až je zpracován obraz v původním rozlišení, kdy může být nalezen hledaný objekt. Tato metoda je velmi rychlá, neboť oblasti nižších úrovních, ve kterých se hledaný objekt nevyskytuje, pak nejsou ve vyšších úrovních zpracovávány.

6.2 Metody založené na hledání transformací

Následující metody užívají modely sestávající se z konečného počtu souboru bodů a jejich pozic. Rozpoznávání souboru bodů může být provedeno tak, že nejprve je soubor bodů extrahován z obrazu spolu s jejich pozicemi. Tento krok se nazývá rozpoznávací fáze. Následně jsou odhadnuty parametry transformace, které určují mapování souboru bodů modelu na soubor bodů extrahovaným z obrazu. Toto se provádí maximalizací podobnosti mezi souborem scény obrazu a transformovaným souborem bodů modelu nebo minimalizací jejich vzdálenosti. Poté je prohledáván takzvaný transformační prostor, jenž zahrnuje soubor všech možných kombinací transformačního parametru.

U těchto metod rozlišujeme následující transformační třídy (určují jaké transformace budou odhadovány):

- translace (posun)
- rigidní transformace – zahrnuje posun a rotaci
- podobnostní transformace – zahrnuje rotaci a změnu měřítka
- affinní transformace
- pespektivní transformace

6.2.1 Zobecněná Houghova tranformace

Metoda původně vyvinutá k detekci přímých čar, avšak může být zobecněna k detekci libovolně tvarovaných objektů (v případě, že je předem znám tvar objektu). Je určena pro objekty, které není možné jednoduše analyticky. Popis hranice hledaného vzoru je realizován pomocí explicitního seznamu všech bodů hranice.

6.3 Detekce bodů zájmu a deskriptorů oblastí

Úloha rozeznávání objektů v reálném světě může být splněna dvoustupňovou strategií

popisu obsahu obrazu:

1. Nejprve jsou detekovány body zájmu, které jsou pokládány za charakteristické.
2. Následně jsou odvozeny tzv. deskriptory oblasti, každý reprezentující obrazovou informaci dostupnou v lokálním okolí bodu zájmu.
3. Rozeznávání objektů poté může být provedeno porovnáním deskriptorů oblastí, příp. jejich lokacemi (prostorovým rozložením), s databází modelu. Tato databáze bývá většinou vytvářena automaticky během fáze učení.

V následujícím textu budou představeny dvě metody rozpoznávání objektů, a to metoda SIFT a metoda SURF .

6.3.1 Popis metody Scale invariant feature transform

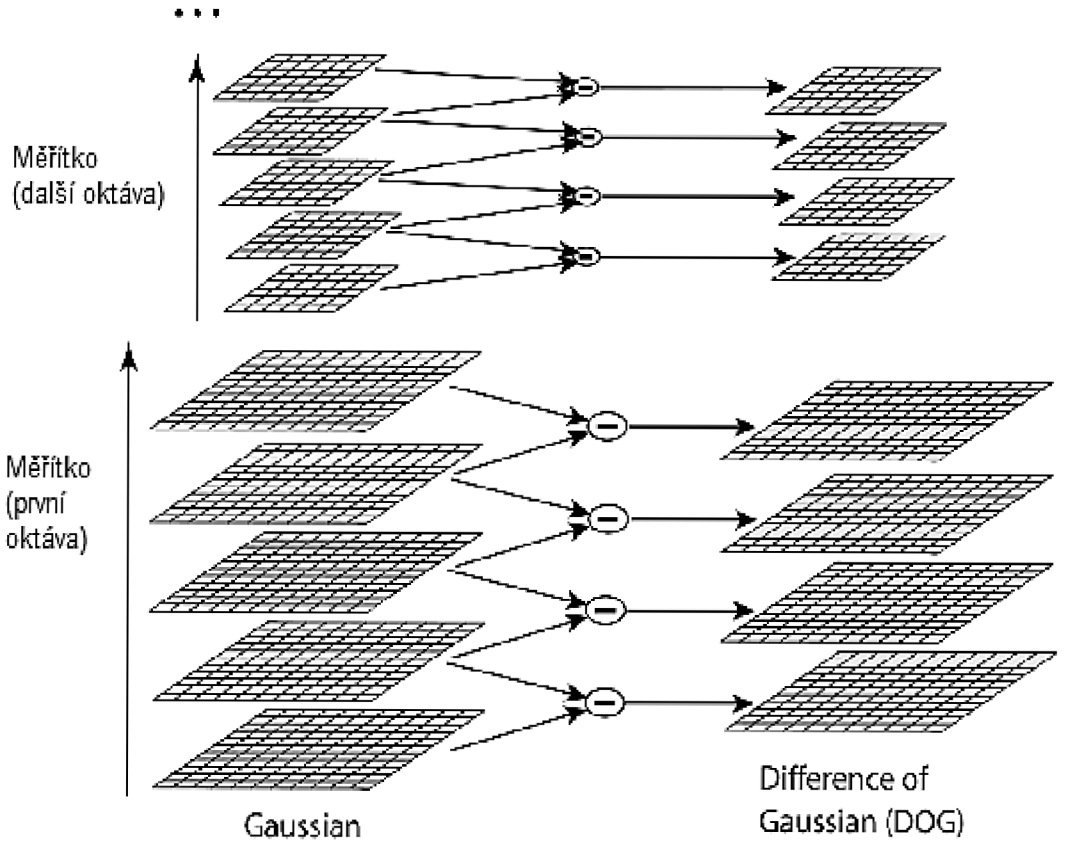
Scale invariant feature transform (dále jen SIFT) je algoritmus pro detekci a popis lokálních prvků v obrazech. Metoda je invariantní vůči změně měřítka, orientace a částečně i vůči affinní poruše a změnám v osvětlení. Tuto metodu popsal David G. Lowe v roce 1999. Stránky týkající se této metody jsou volně přístupné v síti internet[1], kde je také možno získat jak původní verzi z roku 1999, tak i poslední verzi dokumentace metody. Je zde umístěn i demonstrační program předvádějící možnosti metody. Tento algoritmus je patentovaný v USA, kdy držitelem patentu je Univerzita v Britské Kolumbii.

V lit. [7] jsou popsány hlavní fáze výpočtů užitých k získání sady příznaků (features) obrazu:

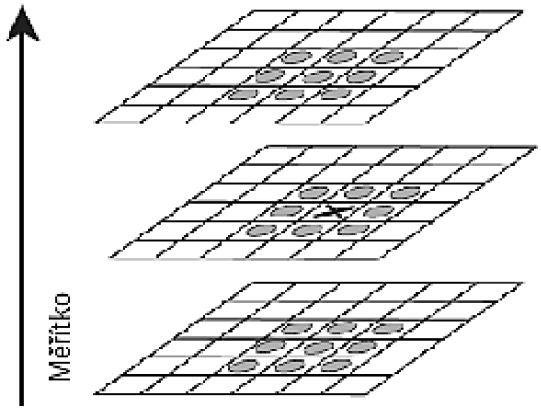
1. Nejprve jsou prohledány všechny pozice v obrazu v různých měřítcích (scale-space), aby byly nalezeny potencionálně zajímavé body, které jsou invariantní vůči měřítku a orientaci. Tyto body jsou nazývány klíčové body (keypoints). Přitom je vytvořena pyramida měřítek. Tímto krokem je zajištěna invariance vůči změně měřítka. Každý stupeň této pyramidy se nazývá oktáva a jejich počet lze měnit. Při tvorbě pyramidy je nejdříve na každé oktávě několikrát spočítána konvoluce obrazu s Gaussovými filtry, přičemž následující obraz se počítá z obrazu předchozího. Poté jsou tyto vzniklé obrazy jsou poté od sebe odečteny. Vždy se odečítají sousední rozostřené obrazy. Tento rozdíl je nazýván Difference od Gaussians (DoG). Poté je podvzorkován obraz, jenž má dvojnásobnou hodnotu směrodatné odchylky

1 http://www.cs.ubc.ca/~lowe/keypoints/

Gausssova rozdělení oproti její počáteční hodnotě. Provedeno je to tím způsobem, že vybrán každý druhý pixel v každém řádku a sloupci obrazu. Klíčové body jsou poté brány jako maxima, příp. minima obrazu v různých měřítcích. Toto je vypočteno tak, že jsou porovnávány všechny pixely v DoG obrazech, kdy každý pixel je porovnán s jeho 8 sousedy v daném měřítku a s 9 sousedy v sousedních obrazech v měřítku. Bod je vybrán jako klíčový bod, pokud je větší, případně menší, než všechny jeho sousední body. Výpočetní náročnost tohoto kroku snižuje fakt, že většina bodů je vyřazena po prvních pár porovnání se sousedními body. Na obrázku Chyba: zdroj odkazu nenalezen je v lit. [7] uvedený obrázek, v němž je znázorněn postup při vytváření DoG obrazů. Pod tímto obrázkem je v téže literatuře uvedený obrázek znázorňující postup při detekci maxima a minima v bodě.



Obrázek 6.1: SIFT - postup vytváření Difference of Gausian



Obrázek 6.2: SIFT - postup při detekci maxima a minima v bodě

2. Vzhledem k tomu, že předchozí operace vyprodukuje velké množství klíčových bodů, jsou odfiltrovány body, které jsou nestabilní. Jsou to například body s nízkým kontrastem, které jsou náchylné k šumu, nebo body lokalizované na hraně. To je provedeno tím způsobem, že u každého lokálního extrému je vzat bod spolu s jeho 3x3 okolím a tyto body jsou proloženy trojrozměrnou kvadratickou funkcí. Na základě tvaru funkce je potom rozhodnuto, zda se jedná o stabilní nebo nestabilní bod. Dále je možno pomocí minima, případně maxima této kvadratické funkce určit polohu klíčového bodu se subpixelovou přesností.

3. Následně je klíčovým bodům přiřazena orientace a velikost. Tyto jsou určeny v rozostřeném obraze pomocí první derivace v místě klíčového bodu ve směru os x a y, kdy jsou spočítány d_x a d_y. Pomocí těchto derivací je pak spočítána orientace α a velikost L pomocí následujících vztahů z lit. [7]:

$$L = \sqrt{d_x^2 + d_y^2} \tag{6.1}$$

$$\alpha = \arctan\left(\frac{d_y}{d_x}\right) \tag{6.2}$$

4. Dále jsou vypočteny deskriptory. Deskriptor popisuje okolí feature o velikosti L. Jeho výpočet je proveden relativně vůči orientaci feature α. Deskriptor je získán tak, že je nejprve okolí klíčového bodu rozděleno na 4x4 oblasti. Pro každou oblast je pak spočítán histogram orientace gradientů s 8 položkami. Výsledky jsou pak vynásobeny Gaussovou funkcí a normalizovány. Výsledných 128 reálných čísel je pak SIFT deskriptor daného klíčového bodu.

5. Poté, co jsou určeny deskriptory každého klíčového bodu, jsou spočítány deskriptory obrazu vyhledávaného objektu. Poté jsou deskriptory obou obrazů vhodnou srovnávací metodou porovnány a na základě podobnosti je určeno, zda a kde se hledaný objekt ve scéně nachází.

6.3.2 Popis metody Speeded-up robust features

Metoda Speeded-up robust features (dále jen SURF) je algoritmus pro detekci a popis lokálních prvků v obrazech. Metoda je částečně vůči affinní poruše a změnám v osvětlení. Popis pomocí deskriptorů vygenerovaných metodou SURF je invariantní vůči rotaci a vzdálenosti kamery od popisovaného objektu [12]. Tuto metodu zveřejnil v roce 2006 Herbert Bay a kol. Tato metoda je rychlejší než SIFT a podává taktéž lepší výsledky [1]. Taktéž tato metoda je patentově chráněna.

SURF algoritmus je založen na stejných principech jako algoritmus SIFT, avšak užívá odlišný postup. Obě tyto techniky užívají k dosažení nezávislosti na měřítku přístup, že zkoumají obraz v rozdílných měřítcích, měřítkovém prostoru (scale space), za využití Gaussových filtrů. Obě metody rozdělují měřítkový prostor na stupně a oktávy. Oktáva odpovídá násobku σ. Jednotlivé oktávy jsou rozdělena dále na stupně. Metoda SURF užívá detektor oblastí založený na hessiánu (determinantu Hessovi matice) k nalezení bodů zájmu. Determinant Hessovi matice vyjadřuje rozsah odezvy a je výrazem místní změny v okolí oblasti. Tvar Hessovi matice užívaný v metodě SURF je dle [1]

$$H(x,\sigma)=\begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{6.3}$$

, kde x představuje bod ve vstupním obraze I a $L_{xx}(x,\sigma)$ je konvoluce druhé derivace Gaussovy funkce $\frac{\delta^2}{\delta} x^2 g(\sigma)$ se vstupním obrázkem I.

Základem SURF detekce je potlačení hodnot, které nejsou maximálními, determinantů hessových matic. Vzhledem k tomu, že konvoluce je velmi náročná na výpočet, je tento proces aproximován a tím i zrychlen použitím integrálních obrazů a aproximovaných jader.

Integrální obraz

Integrální obraz $I(x)$ je obraz, jehož každý bod x je součtem všech pixelů mezi počátkem a x. Toto lze matematicky vyjádřit pomocí rovnice uvedené v lit. [1]

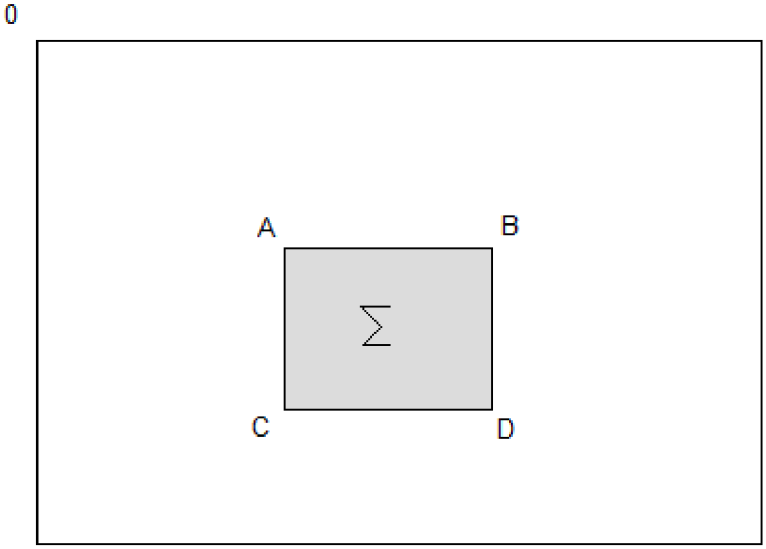
$$I_{\Sigma}(x)=\sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j) \tag{6.4}$$

, kde *I(i,j)* označuje hodnotu pixelu na dané pozici *(i,j)*, *x,y* souřadnice daného bodu

Užití integrálních obrazů umožňuje, jakmile je integrální obraz spočítán, spočítat ve třech krocích součet intenzit v jakékoliv pravoúhlé, svisle orientované, ploše. Označíme-li krajní body této plochy jako *ABCD*, bude součet dán vztahem

$$\text{Součet} = A\text{-}C\text{-}B\text{+}D \tag{6.5}$$

Čas výpočtu je přitom nezávislý na velikosti plochy.



Obrázek 6.3: Princip integrálního obrazu

Filtry

Gaussovské jádra druhého řádu použité v Hessově matici musí být diskretizována a oříznuta.

SURF algoritmus aproximuje tyto jádra obdélníkovými boxy, tvz. 2-D filtry. Tímto způsobem je možno vypočítat aproximovanou konvoluci efektivně s libovolně velkým jádrem za využití integrálního obrazu. Vzhledem k tomu, že tato aproximace zkresluje hodnotu Hessiánu, je možno použít vztah z lit. [1]

$$H = D_{xx} D_{yy} - (0{,}9\, D_{xy})^2 \tag{6.6}$$

, kde D_{xx}, D_{yy}, D_{xy} jsou aproximovaná diskrétní jádra.

Tímto vztahem zajistíme zvýšení výpočetní rychlosti, aniž by byla snížena přesnost

výpočtu.

Vytvoření měřítkového prostoru

Na rozdíl od metody SIFT, která postupně obraz podvzorkuje, metoda SURF k dosažení nezávislosti na měřítku pracuje s Gaussovským jádrem. Zvětšením Gaussovského jádra je matematicky ekvivalentní výpočtu této derivace na úměrně zmenšeném obraze. Vzhledem k tomu, že metoda SURF užívá integrální obrazy, jejichž výpočet je stejně dlouhý pro libovolnou velikost Gaussovského jádra, je tato metoda rychlejší než SIFT.

Jako úvodní vrstva v měřítkovém prostoru je brán výstup filtru o velikosti 9x9. Tento je označován jako měřítko s=1,2 (což přibližně odpovídá Gaussovskému jádru σ=1,2). Další vrstvy jsou získávány filtrováním obrazu s postupně větší maskou. Výhodou tohoto přístupu je, že pokud není podvzorkován obraz, nedochází k aliasingu. Na druhou stranu 2-D filtry zachovávají vysokofrekvenční složky, což může limitovat invarianci vůči měřítku.

Jádra při zvětšování musí mít lichou velikost, aby existoval středový pixel. Dále obdélníkové oblasti musí mít stejnou velikost.

Lokalizace bodů zájmu

K lokalizaci zájmových bodů v obraze a přes měřítka je použito potlačení hodnot, které nejsou maximálními, v okolí 3x3x3. Maxima Hessovi matice jsou pak interpolovány v měřítkovém a obrazovém prostoru, neboť na vyšších oktávách jsou oblasti pokryté filtrem velké a z toho vniká značná chyba.

Deskripce bodů zájmu

Účelem deskriptoru je poskytnout unikátní a robustní popis vlastnosti. SURF deskriptor je založen na odezvách Haarových vlnek. Může být spočítán efektivně za pomoci integrálních obrazů. Deskriptor je v podstatě 64-rozměrný vektor hodnot spočtený na okolí detekovaného bodu zájmu. Na jak velikém okolí je počítáno závisí na jakém měřítku byl daný bod detekován. Pro dosažení invariance vůči rotaci je určena orientace tohoto okolí.

Vzhledem k tomu, že v mnohých aplikacích není třeba, aby byla počítána orientace okolí, je možno tento výpočet vynechat a zrychlit tím výpočet deskriptoru. Tato verze se nazývá U-SURF. I přesto, že není počítána orientace okolí, je U-SURF robustní vůči rotaci ±15° [1].

SURF deskriptor popisuje zájmovou oblast o velikosti $20s$. Pokud byla počítána orientace okolí, je tato oblast natočena. Tato oblast je rozdělena na 4x4 podoblasti, které jsou popsány hodnotami odezvy vlnky ve směru x a y. Každá z oblastí je potom popsána 4-rozměrným vektorem, čímž získáme 64 hodnot. Poté, co je tento vektor normalizován na délku 1, získáme deskriptor klíčového bodu.

7 Prostředí JAVA a sledování objektů ve videosekvenci

Pro tuto práci byl zadán programovací jazyk JAVA, ve kterém byla později vytvořena aplikace pro sledování daného objektu ve videosekvenci. Při vývoji aplikace byla použita open source vývojová platforma Eclipse (dostupné na http://www.eclipse.org/). Tato platforma byla zvolena z důvodu předchozích zkušeností s tímto prostředím.

7.1 Volba vhodné knihovny

Po zvolení vhodného vývojového prostředí byla volena knihovna, jež byla následně použita pro vývoj aplikace ke sledování objektu ve videosekvenci. Jako možné knihovny byly vybrány:

- OpenCV – tato knihovna je zdarma k dispozici, obsahuje ze všech nejvíc knihoven s funkcemi, ať už se jedná o nízkoúrovňové nebo se jedná o vysokoúrovňové. Má za sebou dlouhá léta vývoje, kdy v počátcích byl vyvíjen firmou Intel. K této knihovně existuje spousta dostupné literatury. Jeho nevýhodou je, že se jedná o knihovny v jazyce C/C++ a je nutno použít wrapper.
- Java Motion Tracking Framework – framework zaměřený na sledování pohybu v sekvencích snímků. Na oficiálních stránkách se k tomuto nachází pouze kusé informace.
- Java Media Framework – specializován na práci s multimédii. Dostupný volně ze stránek vývojáře, firmy Sun. Je však velmi neaktuální a neudržovaná.
- BoofCV – knihovna počítačového vidění. Jedná se o knihovnu Java, kdy není nutné užívat wrapper. Množstvím dostupných funkcí však nemůže konkurovat OpenCV. Je stále vyvíjena (je v ranné fázi vývoje) a má dobrou perspektivu do budoucna.

Až na OpenCV se jedná o knihovny pro jazyk JAVA. V případě OpenCV je nutno použít

wrapper do jazyka JAVA, aby mohly být využity její funkce. Všechny knihovny jsou dostupné volně, tudíž tento argument nehraje roli.

Hlavním argumentem pro výběr byly funkce dostupné v knihovnách. Porovnáním podporovaných funkcí jednotlivých knihoven byly do užšího výběru vybrány knihovny OpenCV a BoofCV. Ačkoliv je BoofCV knihovnou pro jazyk JAVA, byla nakonec vybrána knihovna OpenCV. Důvodem byl fakt, že BoofCV je mladý projekt, který je stále ve vývojové fázi a tudíž neobsahuje tolik knihoven.

7.2 OpenCV a JavaCV

OpenCV (Open Source Computer Vision) je knihovna pro manipulaci s obrazem, se zaměřením především na počítačové vidění a zpracování obrazu v reálném čase. OpenCV je vydáváno po licencí BSD a je zdarma pro akademické i komerční účely. Tuto knihovnu je možno stáhnout na adrese http://opencv.org/. V současné době má rozhranní C++, C a Python běžícími pod operačními systémy Windows, Linux, Android a Mac [8].

Tato knihovna byla původně vyvíjena společností Intel. V současné době je vývojářem Willow Garage. Knihovna je vyvíjena v jazyce C/C++.

Značnou výhodou OpenCV je, že je v něm implementována řada funkcí umožňujících manipulaci s obrazy, a to od nízkoúrovňových až po vysokoúrovňové, jakými jsou i funkce pro SURF featur z obrazu.

Pro použití knihovny OpenCV v jazyce JAVA bylo v době psaní programu nutno použít wrapper, který zpřístupní funkce knihovny. V průběhu psaní programu však OpenCV začal podporovat i vývojové prostředí v jazyce Java ve verzi pro stolní počítače. Již předtím však byl pro tento účel vybrán wrapper JavaCV. Tento je dostupný na adrese http://code.google.com/p/javacv/.

Toto spojení však přineslo i jednu nepříjemnou věc, a tou je pravděpodobná neslučitelnost některých verzí OpenCV a JavaCV. V době, kdy byla vybírána vhodná knihovna, byla nejnovější verze této knihovny 2.4.3. Na WWW stránkách projektu JavaCV však byla k dispozici pouze verze 0.2. Ačkoliv bylo na těchto stránkách psáno, že je podporována knihovna OpenCV verze 2.4.3, přes veškeré snahy se nepodařilo OpenCV pomocí JavaCV v Eclipse zprovoznit. Problém byl vyřešen až když byla po několika dnech umístěna na WWW stránky verze 0.3 wrapperu JavaCV.

Největší nevýhodou JavaCV je nedostatek výukových materiálů. Na oficiálních stránkách existuje diskusní skupina, kde jsou zodpovídány dotazy uživatelů. Dále jsou zde uvedeny ukázkové programy z knihy OpenCV2 - Computer Vision Applications Programming Cookbook, jejímž autorem je Robert Laganière. Tyto programy jsou převedeny do jazyku java. Další oficiální materiály týkající se JavaCV bohužel neexistují.

8 Aplikace v jazyce JAVA

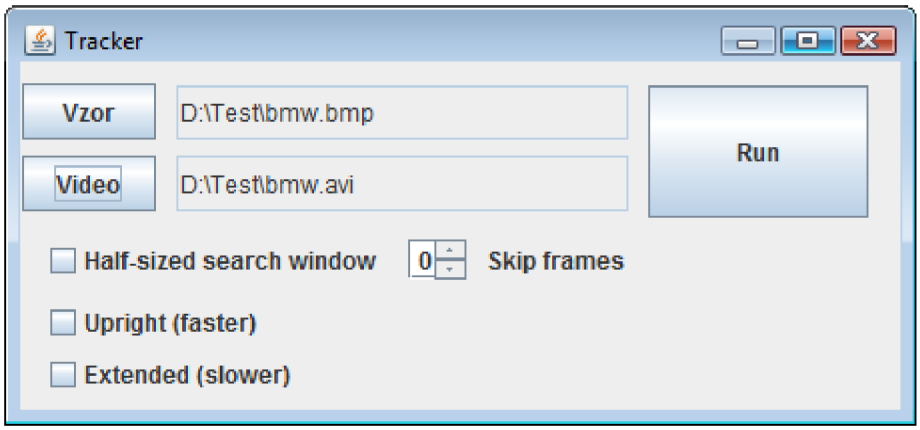
Pro ověření funkčnosti spojení vývojového prostředí Eclipse s OpenCV pomocí JavaCV byla vytvořena aplikace. Jejím úkolem je otevřít videosekvenci, postupně načítat jednotlivé snímky, upravit je a následně je uložit opět na disk.

Jak již bylo řečeno, pro vývoj aplikace byly použity knihovny OpenCV 2.4.3 s wrapperem JavaCV 0.3. Pro správný chod programu je však nutné mít nainstalovány správné kodeky pro práci s vidosekvencí, která má být zpracována.

V této části práce bude rozebrána a popsána tato vytvořená aplikace.

8.1 Hlavní okno aplikace

Aplikace je ovládána přes hlavní okno aplikace, které se zobrazí při jejím spuštění. Toto okno je zobrazeno na obrázku 8.1. Pro vytvoření ovládacích prvků okna aplikace je použita knihovna Swing, což je knihovna uživatelských prvků na platformě Java pro ovládání počítače pomocí grafického rozhraní [13].



Obrázek 8.1: Okno aplikace

Vlevo nahoře se nachází tlačítko Vzor. Po stisku tohoto tlačítka je otevřeno okno s

výběrem souborů. Tímto si uživatel volí soubor typu obrázek, jenž obsahuje hledaný vzor. Jméno souboru spolu s umístěním na disku je pak zobrazeno napravo od tlačítka Vzor. Dále je otevřeno nové okno, ve kterém je zobrazen tento obrázek.

Pod tlačítkem Vzor se nachází tlačítko označené jako Video. Po zvolení této volby je otevřeno okno s výběrem souborů. Tímto si uživatel volí soubor typu video, ve kterém bude vyhledáván objekt z obrázku se vzorem. Po vybrání požadovaného souboru k otevření a potvrzení pomocí tlačítka OK je okno s výběrem souborů zavřeno a jméno souboru spolu s umístěním na disku je pak zobrazeno napravo od tlačítka Video.

V pravém horním rohu okna aplikace je tlačítko Run, jež spouští zpracování videosekvence. Po stisknutí tohoto tlačítka je spuštěn dialog pro výběr jména a umístění zpracované videosekvence. Po výběru je spuštěno vyhledávání. Činnost programu po stisku tohoto tlačítka bude rozebrána v dalším textu.

V prostoru pod tlačítkem Video se nachází zaškrtávací tlačítka, kterými se nastavuje chování programu při vyhledávání. Jsou to následující tři tlačítka:

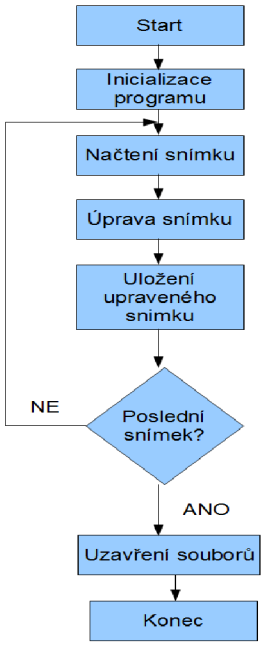
- Half-sized search window – vzhledem k tomu, že s rostoucím rozlišením roste i výpočetní náročnost vyhledávání, byla zde implementována možnost snížit rozlišení snímku, při kterém je prováděno vyhledávání. Tato volba, pokud je aktivní, způsobí, že rozlišení je sníženo na 1/2 ve směru osy x i y. Vyhledávání v menším rozlišení je rychlejší, avšak je zde větší riziko toho, že objekt nebude ve scéně nalezen. To je z toho důvodu, že při nižším rozlišení je nalezeno menší množství klíčových bodů.
- Upright – tato volba se týká nastavení detektoru. Pokud je aktivní, není u jednotlivých featur počítána orientace. Tímto způsobem dojde ke zrychlení zpracování, avšak detekce již není tak invariantní vůči rotaci. Používá se za situace, když srovnávané featury mají podobné úhly.
- Extended – když je tato volba aktivní, mají deskriptory 128 prvků oproti standardním 64, když je volba vypnuta. Se zapnutou volbou je prováděna preciznější detekce, avšak za cenu větší výpočetní náročnosti a tím i tedy delšího času zpracování snímku.

Napravo od těchto tlačítek je volba Skip frames. Tato volba udává kolik snímků bude

vynecháno ze zpracování. Tímto způsobem bude videosekvence zpracována rychleji. Na nezpracovaných snímcích je převzata poloha objektu ze snímku předchozího. Vychází se z předpokladu, že objekt ve videosekvenci bude mít v přilehlých snímcích velmi blízkou polohu.

8.2 Popis jednotlivých funkcí aplikace

Program v současné podobě zajišťuje tři hlavní funkce: načítání snímků videosekvence, manipulaci se snímky a ukládání upravených snímků do nové videosekvence. Chod programu lze vyjádřit pomocí diagramu znázorněného na obrázku 8.2.



Obrázek 8.2: Diagram programu

8.2.1 Popis bloků inicializace programu a načtení snímku

Blok inicializace programu získání jména zpracovávaného souboru, jména souboru s vyhledáváným obrázkem a jména výstupního zpracovaného souboru. Taktéž do tohoto bloku spadá vytvoření instancí objektů grabber, zodpovědné za načítání ze souboru, a recorder, zodpovědné za ukládání do souboru. Pro otevření a uložení souboru jsou použity jména a cesty získané pomocí výběru z menu. Následně jsou otevřeny soubory pro čtení a zápis.

Poté je snímek načten pomocí příkazu grabber.grab().

8.2.2 Popis bloku úprava snímku

V bloku úpravy snímku dochází nejprve k jeho převodu z barevného na šedotónový pomocí příkazu

```
cvCvtColor(frame, image, CV_RGB2GRAY);
```

, kde jednotlivými parametry jsou

- frame – zdrojový (barevný) snímek,
- image – cílový (šedotónový) snímek,
- CV_RGB2GRAY – kód konverze mezi barevnými prostory, zde mezi prostorem RGB a šedotónovým.

Tato úprava je nezbytná, neboť extraktor příznaků v OpenCV vyžaduje na svém vstupu šedotónový obraz. Dále jsou získány klíčové body a jejich deskriptory snímku a vzorového obrazu pomocí příkazu

```
cvExtractSURF(image, null, keypoints, descriptors, tempStorage, parameters, 0);
```

, kde jednotlivými parametry jsou

- image – zdrojový šedotónový snímek (příp. obraz vzoru),
- null – zde na tomto místě je volitelně zadávána maska označující oblasti, kde se mají vyhledávat příznaky,
- keypoints – vektor s klíčovými body,
- descriptors – vektor s deskriptory,
- tempStorage – pamětový prostor, kde budou uchovány vektory s klíčovými body a deskriptory,
- parameters – obsahuje parametry SURF algoritmu. Tyto parametry ovlivňují značnou měrou rychlost a přesnost vyhledávání. Vzhledem k tomu, že výsledek závisí na zpracovávaném snímku, jsou tyto parametry voleny s ohledem právě na obsah zpracovávaného snímku (výskyt šumu ve snímku, rozlišení snímku, složitost scény s ohledem na množství předmětů se v ní vyskytujících) a požadovaném

výstupu (např. je-li zde požadavek zpracování v reálném čase). Nastavitelnými parametry jsou:

- extended – určuje, jestli výstupní vektor deskriptoru má 64 položek (základní) nebo 128 položek (rozšířený). V tomto programu je možno tuto položku měnit pomocí volby Extended.
- HessianTreshold – udává práh, kdy jsou pouze u bodů s hessianem vyšším, než je tento práh, jsou extrahovány příznaky. Tímto způsobem lze omezit množství příznaků ve snímku, čímž se zkrátí čas na zpracování. Hodnota je opět velmi závislá na vlastnostech snímku. Doporučovaná hodnota je 300-500, v programu je užita hodnota 500.
- nOctaves – množství oktáv použitých k extrakci,
- nOctavesLayers – množství vrstev vytvořených v každé oktávě.
- Upright – udává, zda je nebo není u jednotlivých příznaků počítána orientace. Tímto způsobem dojde ke zrychlení zpracování, avšak detekce již není tak invariantní vůči rotaci. Používá se za situace, když srovnávané příznaky mají podobné úhly. Pokud je tento parametr pravdivý, nejsou počítány orientace příznaků. Tento parametr je možno v programu nastavit.

Poté, co jsou získány klíčové body a jejich deskriptory, je provedeno jejich srovnání mezi snímkem a vzorem. K tomuto je využita FLANN knihovna (Fast Library for Approximate Nearest Neighbors). Nejprve je vytvořen vyhledávací index metody nearest neighbor. Pak je provedeno vyhledávání metodou K-nearest neighbor pro daný bod s využitím indexu. Z výsledku jsou pak odstraněny páry, které jsou od sebe vzdáleny více, než je stanovený práh. V tomto programu je stanoven práh na hodnotu 0,6.

Následně je s využitím nalezených párů nalezena perspektivní transformace pomocí příkazu

```
cvFindHomography(pt1, pt2, H, CV_RANSAC, ransacReprojThreshold, mask)
```

, kde jsou vstupními parametry

- pt1, pt2 – vektory bodů vzoru a snímku,

- H – matice perspektivní transformace,
- CV_RANSAC – metoda založená na RANSAC (RANdom SAmples Consensus)
- ransacReprojTreshold – udává maximální povolenou chybu reprojekce, aby byl pár bodů vzat jako přilehlá hodnota (inlier). V programu je užita hodnota 3.
- mask – volitelná maska výstupu.

Pomocí matice perspektivní transformace jsou nyní získány rohové body obdélníku, ve kterém se nachází hledaný objekt. Tento obdélník je poté ve snímku nakreslen.

8.2.3 Popis bloků uložení upraveného snímku a uzavření souborů

Jakmile je snímek upraven, je uložen pomocí příkazu recorder.record(frame);. Pokud následují ve zpracovávaném souboru další snímky, program pokračuje načtením a zpracováním dalšího snímku. V opačném případě jsou ukončeny grabber i recorder.

8.2.4 Spouštění aplikace

Pro spuštění aplikace mimo vývojové prostředí je nutno mít nainstalované prostředí pro běh programů v Javě – JRE. Dále je nutné mít správně nakonfigurovánu knihovnu OpenCV verze 2.4.3. Poté je již možno aplikaci spustit.

9 Výsledky testů

Pro ověření výsledků byly zvoleny tři scény. První z nich je počítačově animované vozidlo jedoucí po silnici. Vozidlo je z pohledu z boku, kdy se na větší části videosekvence nemění jeho tvar ani velikost. Taktéž nedochází k natočení vozidla. Pouze na konci dochází ke zmenšení velikosti vozidla. V průběhu animace však dochází k částečnému zákrytu vozidla, když toto míjí jiné vozidla a lampy pouličního osvětlení. Tato scéna slouží k ověření funkčnosti detektoru, neboť vozidlo je dostatečně veliké s jednoduchými rysy a taktéž počítačově vytvořená scéna je jednoduchá. Obrázek se vzorovým vozidlem má rozměry 353x128 pixelů. Zpracovávané video má rozlišení 854x480 pixelů. Název je „bmw“.

Druhou scénou je záznam kamerového systému monitorujícího dopravu. Tato scéna je komplexní s velkým množstvím objektů. Taktéž hledaný objekt je ve srovnání se zbytkem

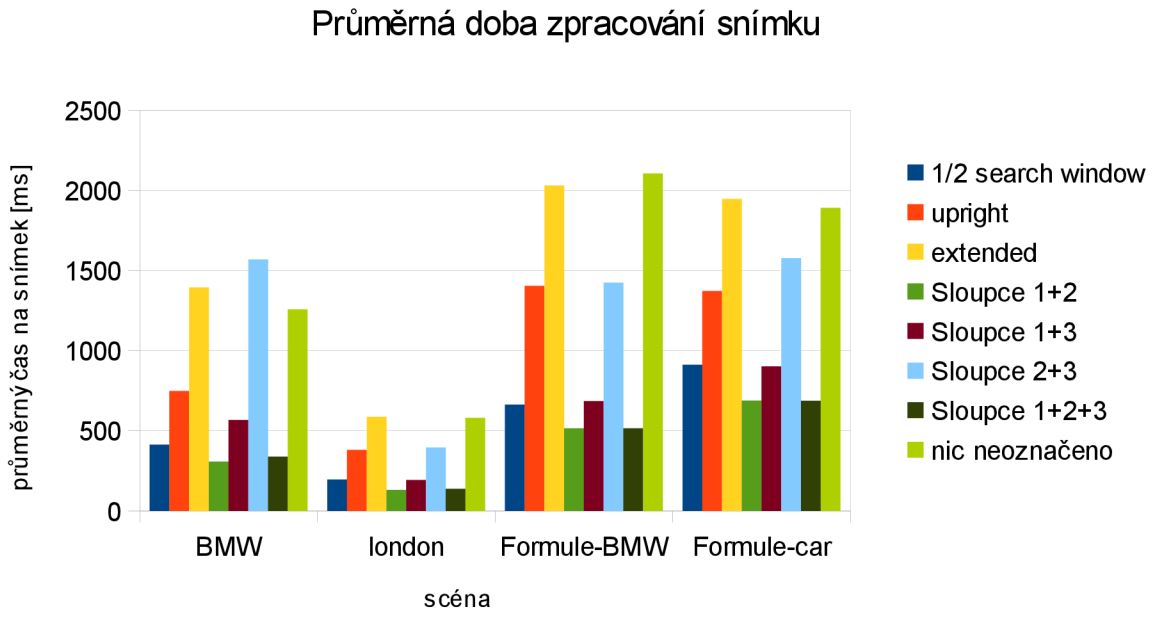
snímku menší. Vozidlo scénou projíždí, kdy na začátku a konci sekvence se na snímcích nenachází. Ve scéně dochází u hledaného objektu ke změně velikosti. Obrázek se vzorovým vozidlem má rozměry 158x89 pixelů. Zpracovávané video má rozlišení 592x416 pixelů. Název je „london“.

Třetí scénou je vozidlo na závodním okruhu. Vozidlo jede téměř čelem ke kameře, kdy začíná ve větší vzdálenosti od kamery a postupně se přibližuje. Dochází tedy ke změně měřítka. Obrázek se vzorovým vozidlem má rozměry 147x105 pixelů. Zpracovávané video má rozlišení 854x480 pixelů. Název je „formule-bmw“.

Čtvrtá scéna je další vozidlo na automobilovém okruhu. Tentokrát projíždí zatáčkou, kdy scéna začíná s vozidlem natočeným téměř na kameru a končí s vozidlem natočeným levým bokem ke kameře. Obrázek se vzorovým vozidlem má rozměry 382x164 pixelů. Zpracovávané video má rozlišení 854x480 pixelů. Název je „formule-car“.

Všechny scény byly testovány na notebooku MSI EX-610, který je osazen procesorem AMD Athlon X2 (frekvence 2x1,8 GHz), 2 GB paměti RAM a grafickou kartou ATI HD 2400.

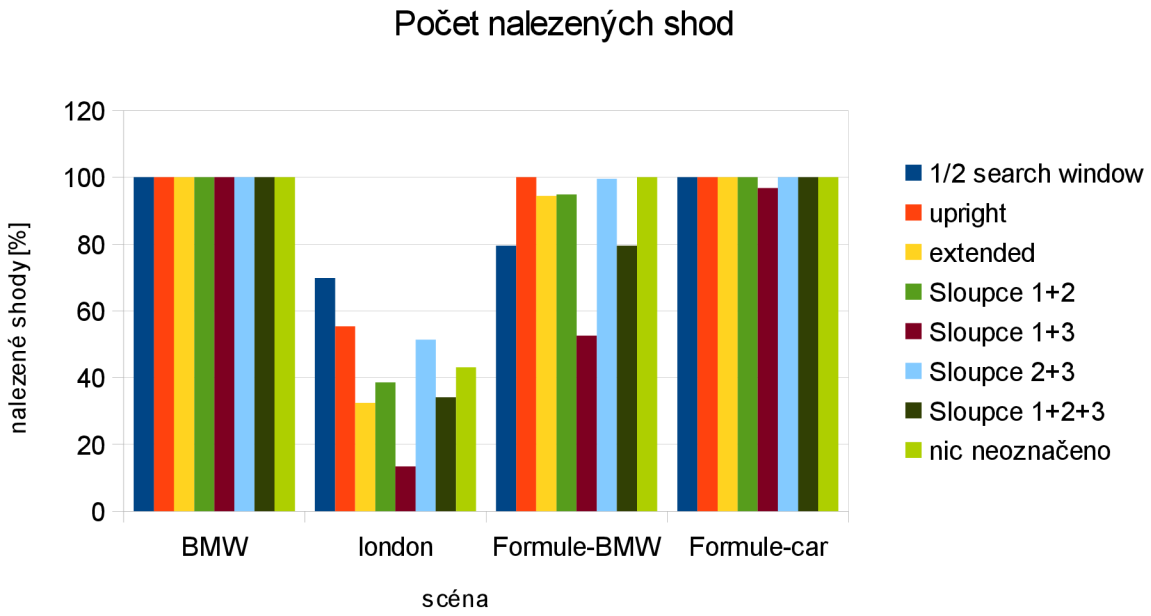
Jako první budou vyhodnoceny testy časové náročnosti, kdy je při jednotlivých možných nastaveních měřen průměrný čas na zpracování snímku.



Tabulka č. 1 : Průměrná doba zpracování

Při pohledu na výše uvedenou tabulku je patrné, že doba zpracování jednoho snímku je příliš vysoká. Ačkoliv nebylo zadáním, aby aplikace pracovala v reálném čase, měly by být časy lepší. Pro zpracování v reálném čase je nutno při snímkové frekvenci 25 snímků/s dosáhnout času 40 ms. Přitom nejlepší výsledek, který byl dosažen, byl 130 ms a nejhorší 2105 ms, což je přes 2 s na snímek. Možností, proč jsou výsledky takové, je více. Jednou z možností je, že testovací sestava byla málo výkonná. Další, avšak ne poslední možností je, že byla špatně naprogramována aplikace.

Dalším kriteriem, které bude hodnoceno, je počet nalezených shod. Toto je shrnuto v následujícím grafu.



Tabulka č. 2: Počet nalezených shod

Při pohledu na výše uvedenou tabulku by se mohlo zdát, že úspěšnost aplikace při vyhledávání je, až na druhou scénu, celkem vysoká. Tyto výsledky se skutečnosti bohužel blíží pouze u první scény. U ostatních scén se vyskytuje jev, kdy aplikace snímek správně označí, že se na něm objekt nachází, avšak určí chybně lokaci objektu.

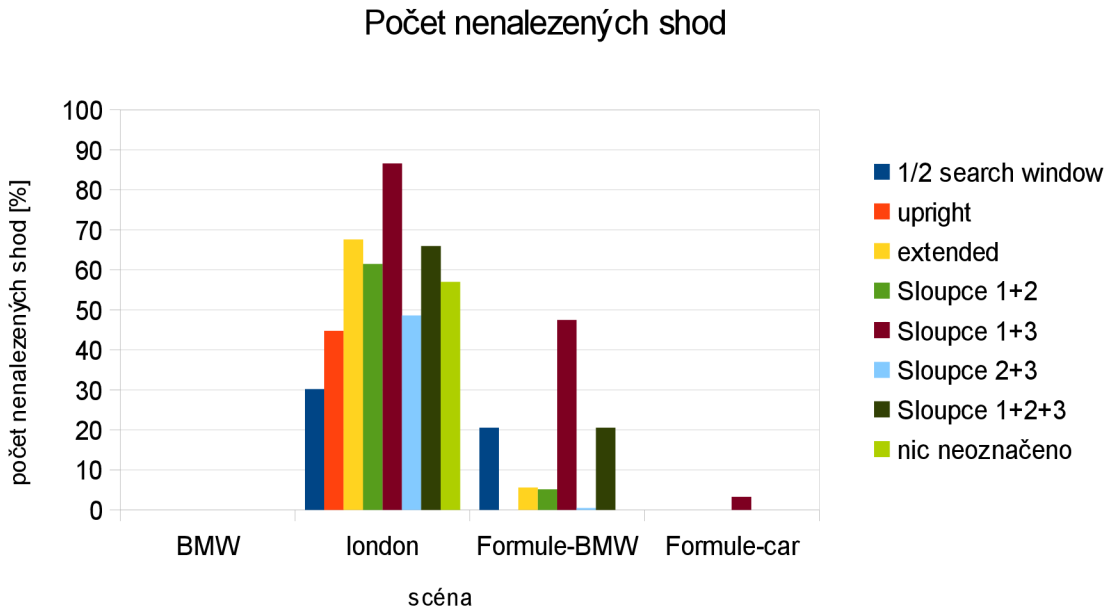
Obdobný fakt platí i při pohledu na další tabulku, kde jsou vyjádřeny procentuelně případy, kdy aplikace objekt nenašla, i když se na scéně nacházel.

Při pohledu na uvedené výsledky vyplývá, že použití rozšířených deskriptorů o délce 128 oproti standardním 64, prodlužuje celkem výrazně dobu zpracování. Přitom přesnost

vyhledávání tímto není o tolik lepší, aby to vyvážilo snížení rychlosti zpracování.

Co však příjemně překvapilo, byla volba Upright, která, pokud byla aktivní, výrazně snížila čas zpracování, aniž by se to projevilo výrazně negativně na přesnosti vyhledávání.

Poslední z voleb, a to snížení rozlišení, sice zvýšila rychlost zpracování, avšak zároveň snížila rozpoznávací schopnost.



Tabulka č. 3: Počet nenalezených shod

Příčin, proč nejsou detekovány objekty ve scéně dle očekávání, může být několik. Jednou z možností je, že obrázky se vzorovými objekty, případně i video, jsou v příliš malém rozlišení. Tím pádem není nalezeno dostatečné množství klíčových bodů a z nich vyplývajících deskriptorů. Že metoda funguje, je ovšem ověřeno na první scéně, kdy detekovaný objekt byl detekován bez problémů i přes to, že místy docházelo k částečnému zákrytu objektu.

10 Závěr

V této práci byla probírána problematika sledování předem daného objektu ve videosekvenci. Po úvodní diskusi kolem problematiky sledování objektu ve videosekvenci, kde byly mimo jiné naznačeny požadavky a omezení této úlohy, jsou rozebrány postupně jednotlivé kroky při rozpoznávání objektů ve snímku.

V další části jsou pak shrnuty možnosti sledování objektu ve videosekvenci v programovacím jazyce JAVA s ohledem na knihovny funkcí, přičemž k tomuto účelu pak byla vybrána knihovna OpenCV. Vzhledem k tomu, že tuto knihovnu nelze použít přímo, byl vybrán wrapper JavaCV.

Poté byla vytvořena aplikace v jazyce Java za využití knihovny OpenCV prostřednictvím wrapperu JavaCV. Ačkoliv její výsledky nejsou úplně přesné, umožňuje se přesvědčit o možnostech nastavení SURF detektoru a vlivu tohoto nastavení na rychlost zpracování a přesnost detekce objektu.

11 Použitá literatura:

[1] BAY, Herbert, Andreas ESS, Tinne TUYTELAARS a Luc LUC VAN GOOL. SURF: Speeded Up Robust Features. In: Computer Vision and Image Understanding (CVIU)[online]. 2006 [cit. 2013-05-20]. Dostupné z: http://www.vision.ee.ethz.ch/~surf/papers.html

[2] BURGER, Wilhelm; BURGE, Mark J. Principles of Digital Image Processing: Fundamental Techniques. Londýn : Springer, 2009. 272 s. ISBN 978-1848001909.

[3] Detekce hran. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-12-12]. Dostupné z: http://cs.wikipedia.org/wiki/Detekce_hran

[4] GABRIEL, P. Klasifikace objektů v obrazech. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 77 s. Vedoucí diplomové práce Ing. Ilona Kalová, Ph.D.

[5] GONZALEZ, Rafael C a Richard E WOODS. Digital image processing. 3rd ed. Upper Saddle River: Pearson, c2008, xxii, 954 s. ISBN 01-316-8728-X.

[6] HORÁK, PH.D., Ing. Karel. A KOL. *Počítačové vidění* [online]. Brno, 2008 [cit. 2012-12-12].

[7] LOWE, David G. Distinctive image features from scale-invariant keypoints. In: International Journal of Computer Vision [online]. 2004 [cit. 2013-05-20]. Dostupné z: http://www.cs.ubc.ca/~lowe/keypoints/

[8] OpenCV Wiki. *OpenCV Wiki* [online]. 2012 [cit. 2012-12-12]. Dostupné z: http://opencv.willowgarage.com/wiki/

[9] ŘÍHA, K. Pokročilé zpracování obrazu. [skriptum] Brno : Vysoké učení technické v

Brně, FEKT, UTKO, 2007.

[10] Segmentace obrazu. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-12-12]. Dostupné z: http://cs.wikipedia.org/wiki/Segmentace_obrazu

[11] STRAKA, Stanislav. *Segmentace obrazu*. Brno, 2009. Diplomová práce. Masarykova univerzita. Vedoucí práce Mgr. Radka Pospíšilová.

[12] SURF. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-12-12]. Dostupné z: http://cs.wikipedia.org/wiki/SURF

[13] Swing (Java). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-12-12]. Dostupné z: http://cs.wikipedia.org/wiki/Swing_(Java)

[14] TREIBER, Marco. *An introduction to object recognition: selected algorithms for a wide variety of applications*. London: Springer, 2010, xvii, 201 s. Advances in pattern recognition. ISBN 978-1-84996-234-6.

[15] WATVE, Alok K. *A Seminar On Object tracking in video scenes* [online]. Indian Institue of Technology, Kharagpur, 2004 [cit. 2012-12-12]. Seminární práce. Indian Institue of Technology, Kharagpur. Vedoucí práce Dr. Shamik Sural.

Seznam zkratek a pojmů

pixel – z ang. Picture element – obrazový bod

RANSAC – Random Saples Consensus – metoda na vyčlenění okrajových bodů

SIFT – algoritmus pro detekci a popis lokálních prvků v obraze

SURF - algoritmus pro detekci a popis lokálních prvků v obraze

Seznam příloh

A – obsah CD

B – přiložené CD

A – Obsah CD

Přiložené CD obsahuje elektronickou podobu této práce ve formátu PDF. Dále obsahuje zdrojové texty a přeloženou aplikaci a dále testovací data. CD je v následující struktuře:

Adresář	Popis
Bin	přeložený kód vytvořené aplikace
Source	zdrojový kód vytvořené aplikace
Vstupy	data, pomocí kterých byla aplikace testována
Výstupy	data zpracovaná aplikací