VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

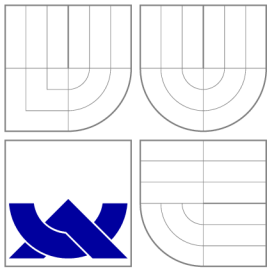
DETEKCE POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEO SEKVENCI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

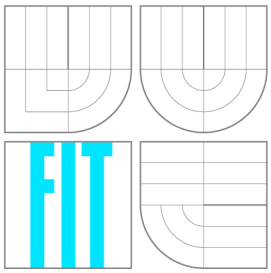
AUTOR PRÁCE
AUTHOR

Bc. JAN HAVELKA

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEO SEKVENCI

MOVING OBJECTS DETECTION IN VIDEO SEQUENCES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN HAVELKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2011

Abstrakt

Tato práce se zabývá problematikou rozpoznávání a detekce objektů a osob ve video sekvenci i ve statickém obraze. Navržená aplikace využívá kombinaci detekce pohybu pomocí background modelu, rozpoznání osob pomocí histogramů orientovaných gradientů a sledování objektů pomocí Lucas-Kanadeho metody.

Abstract

The topic of this thesis is the recognition and detection of moving object and persons in video sequence and in the static image. Designed application uses the combination of background model for movement detection, histograms of oriented gradients method for person recognition and Lucas-Kanade method for object tracking.

Klíčová slova

detekce pohybujících se objektů, detekce osob, HOG deskriptor, Lucas-Kanadeho metoda, Gaussův filtr

Keywords

moving object detection, pedestrian detection, HOG descriptor, Lucas-Kanade method, Gaussian filter

Citace

Jan Havelka: Detekce pohybujících se objektů ve video sekvenci, diplomová práce, Brno, FIT VUT v Brně, 2011

Detekce pohybujících se objektů ve video sekvenci

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla, Ph.D.

.....
Jan Havelka
25. května 2011

Obsah

Kapitola 1

Úvod

Detekce pohybujících se objektů je pro člověka základní dovedností, kterou si vlastně ani neuvědomuje a dělá ji naprosto automaticky. Člověk je schopen ve scéně objekty nejen pozorovat, ale i rozpoznávat. Nedělá mu problém určit, zda se jedná o letadlo, zvíře, míč, osobu, atd.

V počítačovém vidění je tato dovednost pouze simulována tak, aby se co nejvíce přiblížila lidskému perfektnímu systému.

Oproti lidskému vnímání má počítač velmi ztíženou práci. Musí se vypořádat s velkou řadou problémů. Musí například nějak filtrovat vstupní video, aby neobsahovalo šum, který může vzniknout třeba na nekvalitních kamerách nebo velkou citlivostí čipu při nízké hladině osvětlení.

Počítač musí pohyb ve videu detekovat pouze na základě omezeného množství informací, které získá z několika málo obrázků jdoucích rychle za sebou. Naopak člověk má informací mnohem více. Ví, například, kde se nachází, takže může předpokládat, jaké objekty se před ním budou pohybovat. Oči mají mnohem větší rozlišovací schopnost než objektiv kamery, atd.

Když už počítač získá informaci o uskutečněném pohybu, měl by být schopen objekt rozpoznat. Člověku to opět nedělá velké problémy, už kvůli tomu, že má celý život na trénování sady objektů pro rozpoznání. Počítač má v tomhle ohledu o dost ztíženou práci díky omezené sadě trénovacích dat.

Dalším omezením je pro stroj rozpoznávací metoda. V dnešní době neexistuje stoprocentní metoda pro rozpoznávání všech objektů. Za to existuje velké množství různých metod rozpoznávajících různé druhy objektů s různou úspěšností.

Tato práce se specializuje na detekci lidí, a tím pádem i výběr metod pro rozpoznávání objektů je ovlivněn touto oblastí.

Aplikace detekování pohybu ve videu mají velký potenciál co do využitelnosti. Od jednoduchých výstražných zařízení, jako jsou jednoduché detektory pohybu, až po složité, například dopravní systémy. Ve všech případech se snaží o jedno, zastoupit co nejvíce funkci člověka. Detekují a rozpoznávají různé objekty a zapisují jejich chování. Můžou sloužit i tam, kde lidský mozek už nestačí, například v předpovídání přesné trajektorie pohybu.

Detekce osob ve videu má také velké množství využití. Umístěním kamer do obchodních domů je možné sledovat vytíženost jednotlivých obchodů nebo regálů se zbožím a podle toho vytvářet lepší logistiku pro obchod. Kamery mohou snímat přechody pro chodce a určovat, kolik lidí přecházelo na červenou nebo kolik lidí celkově prošlo po ulici. Detekce

lidí nachází využití rovněž v bezpečnostních aplikacích pro zabezpečení objektů.

Bezpečnostní software by tak mohl zachytávat pouze lidské narušitele a nebylo by potřeba zaměstnávat někoho, kdo bude sledovat obraz na kamerách.

V dnešní době existují metody pro rozpoznávání osob s úspěšností až 96%.

Cílem této práce je podrobně prozkoumat problematiku detekce a rozpoznání objektů, konkrétně lidských postav, ve video sekvenci. K detekci a rozpoznávání osob se dá přistoupit hned z několika stran a ne všechny řešení jsou vždy úspěšná. Prozkoumání problematiky znamená, že se musíme na dané téma dívat právě z více stran a musíme zvážit použití i alternativních metod. Nejčastější metody a postupy použitelné pro návrh aplikace jsou zpracovány v rámci teoretických kapitol 2 a 3.

Součástí diplomové práce je rovněž návrh a implementace aplikace, která bude vhodně demonstrovat danou problematiku.

Principiálně je aplikace navržena jako kombinace několika různých metod. Nejprve je tu metoda pro detekci obecného pohybu, která zaznamená jakýkoliv pohyb ve videu. Mějme na paměti, že nás zajímají především lidské postavy, takže tato metoda není dostačující. K ní se připojí metoda pro rozpoznání detekovaných objektů. Pomocí ní budeme schopni klasifikovat zda se jedná o člověka. Pokud rozpoznání skončí úspěšně, je vhodné sledovat takovou osobu po celou dobu jejího výskytu v obraze. O to se stará metoda trackingu. Konkrétní a podrobný návrh je popsán v kapitole 4, implementace pak v kapitole 5.

Kapitola 6 pak popisuje výsledky experimentálních testů implementované aplikace.

Kapitola 2

Detekce pohybu v obraze

2.1 Předzpracování

Některé metody detekce pohybu nejsou schopné pracovat správně, pokud nemají vstupní data v určitém tvaru. Správným tvarem dat pro ně může být například obraz převedený do stupňů šedi nebo jinak upravený.

Pro zlepšení detekce pohybu je též vhodné odstranění šumu ze všech snímků videa.

2.1.1 Odstranění šumu

Šum ve videu vzniká již na snímacím čipu kamery a je v každém snímku odlišný. Tím pádem, pokud se pokusíme detekovat pohyb ve videu odečítáním dvou snímků, získáme hodně nežádoucích informací v podobě pixelů šumu.

K odstranění šumu lze použít mnoho technik

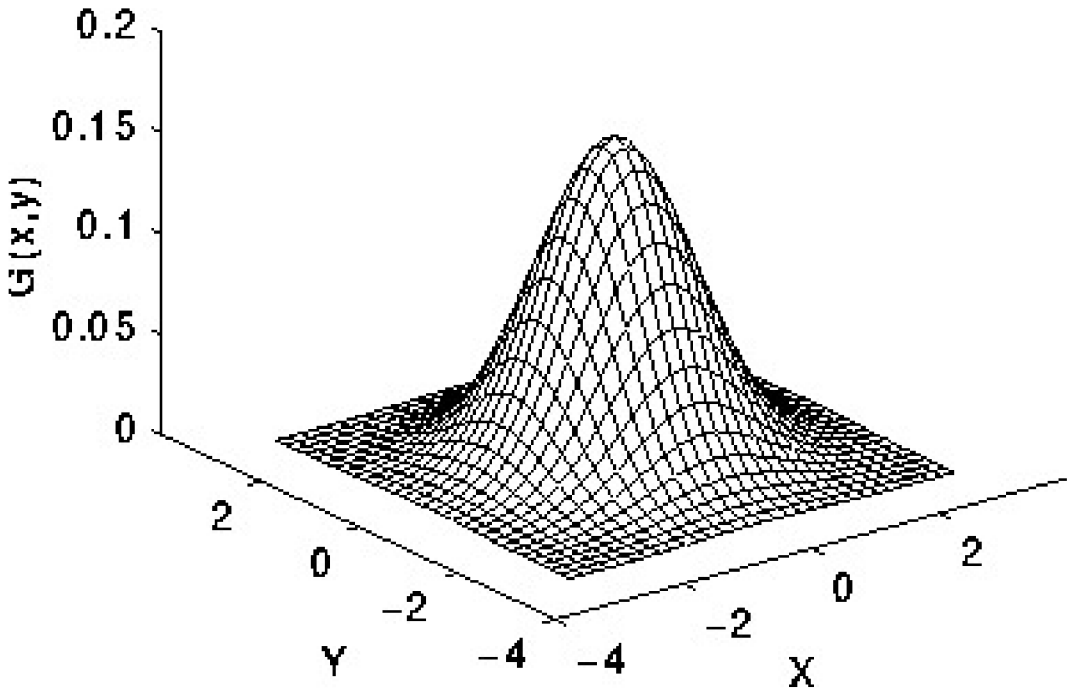
- Lineární filtry
  - Gaussův filtr,
  - Průměrování,
  - Dolní propust,
- Nelineární filtry
  - Mediánový filtr,
  - Konzervativní vyhlazení,
  - Prahování vlnkových koeficientů.

Gaussův filtr

Gaussův filtr je lineární konvoluční filtr. Využívá konvoluce pomocí masky, která je tvořena Gaussovou funkcí

$$G(x,y) = \frac{1}{2\pi\sigma^2} \; e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{2.1}$$

Konvolucí vstupního obrazu a Gaussovy masky získáme sice trochu rozmazaný obraz, ale výrazně ubyde šumu.



Obrázek 2.1: Znázornění Gaussovy funkce pro bod $(0, 0)$ a $\sigma = 2$.

.

2.1.2 Převod do HSV

Převodem do jiného barevného prostoru je v mnoha případech možné zvýšit přesnost detekce některých metod. Například odečítání aktuálního snímku od pozadí by se neprovádělo na základě intenzity barvy, ale na základě jiných hodnot.

HSV (Hue, Saturation, Value) je barevný model vytvořený v roce 1972. Je uváděn jako nejbližší model lidskému chápaní barev.

Skládá se ze tří složek (odtud název):

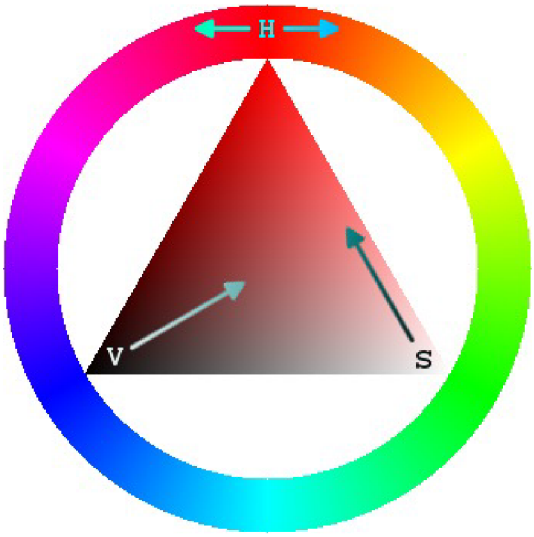
- Hue ... barevný tón, odstín, převládající složka.
- Saturation ... sytost barvy, množství šedi v poměru k odstínu.
- Value ... hodnota jasu, množství bílého světla.

Převod z RGB do HSV

$$H = \begin{cases} \text{nedefinováno}, & \text{jestliže } max = min \\ 60° \cdot \frac{G-B}{max-min} + 0°, & \text{jestliže } max = R \ \& \ G \geq B \\ 60° \cdot \frac{B-R}{max-min} + 120°, & \text{jestliže } max = G \\ 60° \cdot \frac{R-G}{max-min} + 240°, & \text{jestliže } max = B \\ 60° \cdot \frac{G-B}{max-min} + 360°, & \text{jestliže } max = R \ \& \ G < B, \end{cases} \tag{2.2}$$

$$S = \begin{cases} 0, & \text{jestliže } max = 0 \\ 1 - \frac{min}{max}, & \text{jinak}, \end{cases} \tag{2.3}$$

$$V = max. \tag{2.4}$$



Obrázek 2.2: HSV diagram, převzato z [13]

.

Složky $R, G, B \in \mathbb{R}$ jsou v rozmezí $\langle 0; 1 \rangle$. Hodnota max resp. min je maximální resp. minimální hodnotou ze složek R, G, B.

2.2 Základní metody detekce pohybu

Cílem obecné detekce pohybu je zachytit pohybující se objekt. Ačkoliv člověku toto zadání nepřijde nijak obtížné, v dnešní době pravděpodobně neexistuje metoda, která by dosahovala stoprocentní jistoty v určování.

Člověk na videu nemá problém rozpoznat třeba letící míč, projíždějící auto nebo chodce jdoucího po přechodu. Počítač na druhé straně vnímá pouze rychle se měnící matici pixelů.

Z toho už můžeme i selským rozumem odhadovat, jak mohou fungovat ty nejzákladnější metody detekce pohybu. Jelikož se video skládá z několika snímků pouštěných rychle za sebou, můžeme takto sledovat, zda nedochází k výrazné změně mezi dvěma po sobě jdoucími obrázky. V takovém případě ovšem zachytíme pouze jedná-li se o pohyb. Získáme boolovskou hodnotu (true / false). Nebudeme schopni rozlišit pohybující se objekt od stínu mraku nebo jen prostého setmění oblohy.

Aspektů, které negativně ovlivňují správnou detekci pohybu, je mnoho. Namátkou vyberme ty nejznámější: změny osvětlení scény, stíny, odrazy světla (například od skleněné výlohy v záběru), šum v obraze kamery, atd.

Nyní se pokusím najít a popsat několik základních metod pro úspěšnou detekci pohybu.

2.2.1 Odečtení pozadí

Nejjednodušší metoda pro detekci pohybu ve videu je odečtení pozadí od aktuálního snímku. Navazuje na text ze začátku této kapitoly.

Porovnává rozdíly v intenzitě jasové složky mezi jednotlivými pixely aktuálního snímku a snímku pozadí (background). Když zjistí, že se intenzita liší více něž je stanovený práh, označí pixel jako pohybující.

Práh lze nastavit pro každý kanál obrazu a potom využít například Euklidovskou vzdálenost pro vypočtení celkového rozdílu ve všech kanálech obrazu.

Výhodou této metody je zcela zřetelně její rychlost a možnost určit polohu pohybujících se objektů. Nevýhodou pak bude reakce na jakoukoliv změnu ve videu. Metoda reaguje na všechny nežádoucí aspekty popsané v úvodu této kapitoly.

Některé chyby se dají odstranit pomocí aktualizace pozadí neboli modelu pozadí (background model).

2.2.2 Rozdíl jasových histogramů

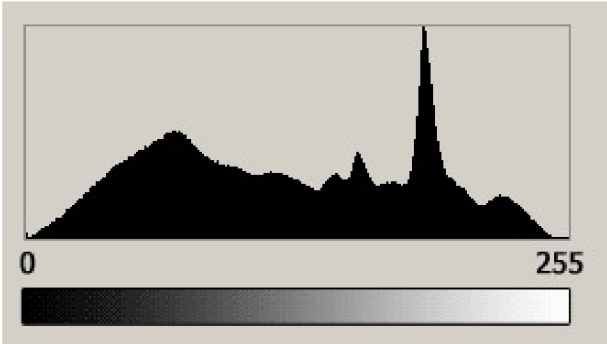
Jedná se o velmi podobnou metodu té předchozí. Namísto porovnávání jednotlivých pixelů mezi sebou je možné porovnávat histogramy celých snímků.

Histogram je statistické znázornění výskytu jednotlivých barev v obrazu. Pokud budeme brát zřetel pouze na jasovou složku obrazu, získáme histogram s 256 sloupci. Každý sloupec odpovídá právě jedné intenzitě. Výška sloupce potom udává zastoupení dané intenzity v celém obraze, viz obrázek 2.3.

Metoda porovnává vždy dva histogramy, jeden pro pozadí a jeden pro aktuální snímek. Pro porovnávání histogramů existuje několik vhodných metod. Uvedu zde dvě nejpoužívanější.

Průnik histogramů

Metoda počítá průnik dvou normalizovaných histogramů podle vzorce 2.5 a zjišťuje, zda se výsledek neliší více než zadaný práh.



Obrázek 2.3: Příklad histogramu. Převzato z [3]

$$H(h_1, h_2) = \sum_i \min(h_{1i}, h_{2i}) \tag{2.5}$$

Metoda nejmenších čtverců
Jedná se o matematicko-statistickou metodu umožňující nalézt zjednodušující model pro složitější struktury dat. Na histogramy ji lze použít ve tvaru 2.6. Počítá průnik dvou normalizovaných histogramů výrazně přesněji než předchozí metoda, což je ovšem vykoupeno vyšší časovou náročností.

$$H(h_1, h_2) = \sum_i \frac{(h_{1i} - h_{2i})^2}{h_{1i} + h_{2i}} \tag{2.6}$$

2.2.3 Local Binary Patterns

Jednou z pokročilejších metod detekce je LBP čili Local Binary Patterns. Metoda původně navržená na klasifikaci textur se hojně používá v mnoha odvětvích počítačového vidění. S úspěchem ji lze použít například na detekci obličejů v obraze, rozpoznávání otisků prstů, klasifikaci textur nebo právě k detekci pohybu.

Detekce pohybu využívající LBP je ve skutečnosti velmi podobná metodě porovnávání jasových histogramů. Rozdíl spočívá v histogramu samotném. V tomto případě nepoužíváme histogram jasových hodnot pro každý pixel, ale konstruujeme úplně nový histogram za pomocí tzv. LBP příznaků.

Jednotlivé příznaky se počítají pro malé oblasti (například 16x16 pixelů) a až dohromady tvoří histogram. Pokud budeme mít histogram pozadí a aktuálního snímku, můžeme je navzájem porovnat a zjistíme došlo-li v obrazu k pohybu či nikoliv.

Díky rozdělení obrazu na menší celky je tato metoda odolná vůči změně globálního osvětlení. Zároveň je imunní vůči detekci menších objektů jako jsou například listy na stromech, ptáci, stébla trávy atd. Nevýhodou zůstává neschopnost určit, kde v obrazu došlo k pohybu. To je způsobené tím, že se porovnávají histogramy pro celé snímky.

Extrahování LBP příznaků je popsáno v kapitole 3.

2.2.4 Optický tok

Metoda optického toku (optical flow) patří mezi výpočetně nejnáročnější metody v oblasti detekce pohybu. Její myšlenka spočívá v tom, že jsme schopni pro každý pixel ve snímku spočítat vektor pohybu a také jeho rychlost.

Mimo jiné z toho vyplývá, že metoda je velmi vhodná například pro předpovídání pohybu objektů. Díky časové složitosti je ovšem téměř nemožné pokrýt každý pixel ve videu a přitom zachovat reálnou rychlost videa.

Rychlost lze zvýšit snížením počtu sledovaných pixelů. Lze vytvořit jakousi řídkou síť pixelů, pro které budeme optický tok počítat.

2.2.5 Aktualizace modelu pozadí

Aktualizace snímku pozadí se hodí téměř v každé metodě detekce pohybu. Ať už jde o metodu, která porovnává jednotlivé pixely (viz 2.2.1) nebo histogramy celých snímků, je nutné je vždy porovnávat se snímkem pozadí.

Pokud bychom tento snímek nechali neustále stejný, riskujeme to, že detekce bude závislá například na změnách globálního osvětlení. Pokud bychom ale snímek postupně v čase aktualizovali na nové hodnoty, mohli bychom se těmto vadám vyhnout.

Základní aktualizace pozadí

Nejjednodušší a nejrychlejší funkční metodou, jak aktualizovat pozadí, je zachytit snímek, ve kterém nedochází k žádnému pohybu a nahradit jím současný snímek pozadí.

Metoda Gaussova průměru

Metoda Gaussova průměru (Running Gaussian Average) přistupuje k aktualizaci pozadí trochu jinak. Vytváří takzvaný model pozadí, který se aktualizuje pro každý pixel jednotlivě. Model se snaží co nejvíce přizpůsobit funkci hustoty Gaussova rozdělení (Gaussian Probability Density Function) pro posledních n snímků.

Aby se razantně nezvyšovala paměťová náročnost aplikace uchováváním posledních n hodnot pro každý pixel, můžeme funkci pro snímek t napsat kumulativně jako

$$\mu_t = \begin{cases} \alpha I_t + (1-\alpha)\mu_{t-1}, & t > 0 \\ I_t, & t = 0 \end{cases}, \tag{2.7}$$

kde μ_t je nová hodnota pozadí, μ_{t-1} je hodnota pozadí v předchozím snímku, I_t je intenzita pixelu ve snímku t a α je koeficient, který určuje rychlost reakce na nová naměřené data a je v rozsahu $0 < \alpha < 1$.

Směrodatná odchylka σ se spočítá vztahem

$$\sigma_t = \sqrt{\alpha(I_t - \mu_{t-1})^2 + (1-\alpha)\sigma_{t-1}^2} \tag{2.8}$$

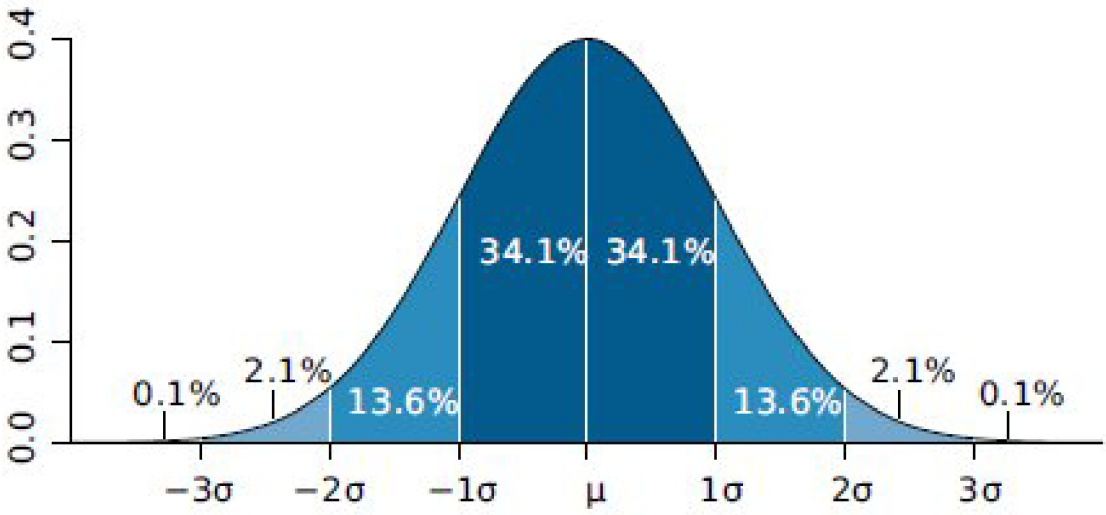
Pro určení pohybujícího se objektu použijeme podmínku

$$|\mu_{t-1}(x,y) - I_t(x,y)| > \theta, \tag{2.9}$$

kde

$$\theta = (c \cdot \sigma_t). \tag{2.10}$$

Konstanta c je zvolena vzhledem ke tvaru Gaussova rozdělení (obrázek 2.4) na hodnotu $c = 3$.



Obrázek 2.4: Hustota Gaussova rozdělení pravděpodobnosti. Převzato z [14]

2.3 Sledování objektů

Sledováním objektů se rozumí děj, u kterého známe pozici objektu v čase t a chceme se dovědět jeho pozici v čase $t+1$. Lidský mozek tuto situaci vyhodnocuje zcela instinktině za pomoci odhadů rychlosti, vzdálenosti a času. Když oko zaznamená pohyb míče, mozek je schopen odhadnout, kam míč dopadne a za jak dlouho. Nejsou to sice žádná přesná data, ale dokáží sloužit alespoň jako odhad pro vyhodnocení situace.

Počítač musí tento děj pouze simulovat, protože postrádá jakékoliv vědění o pohybujícím se objektu. Objekt je pro něj jen sada pixelů.

Existuje několik technik, které je možné použít pro sledování objektu ve videu. Ty mají pro nasazení do aplikace často podobné předpoklady.

Prvním krokem ke správnému trackingu je nalezení tzv. význačných bodů (angl. feature points). Tyto body jsou nějak charakteristické pro daný objekt, který je v našem zájmu a který chceme sledovat.

Vyhledání význačného bodu do velké míry závisí na použité metodě. Obecně lze říci, že se bude jednat o nějak významný bod v obraze. Může to být například charakteristické světlé nebo naopak tmavé místo, hrana mezi dvěma různobarevnými plochami, vrchol nějakého geometrického obrazce.

Pokud máme nalezeny význačné body v čase t, pak jsme schopni pomocí vhodné metody nalézt tyto body i v čase $t+1$.

2.3.1 Lucas-Kanadeho metoda

Jedná se o široce používanou diferenční metodou vyvinutou Brucem D. Lucasem a Takeo Kanadeem (odtud název) v roce 1981. Princip spočívá ve využívání pouze malých lokálních oblastí okolo význačných bodů, díky čemuž dosahuje dobrých rychlostí a není tak citlivá na šum v obraze (viz například [6]).

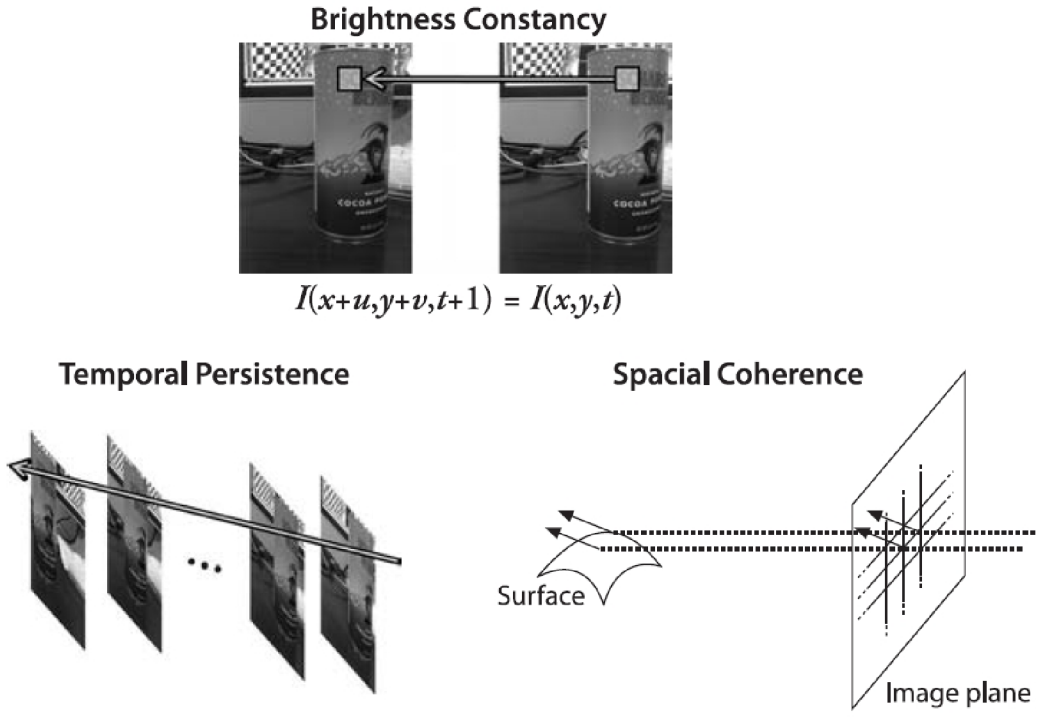
Na druhou stranu, tím, že se zabývá pouze malými lokálními oblastmi, může se lehce stát, že nezachytí rychlejší nebo prudší pohyb. Důvodem je, že bod, který hledá už bude za hranicí oblasti, v níž metoda pracuje.

Tento problém je naštěstí možné odstranit s použitím takzvaného "pyramidového"Lucas-Kanadeho algoritmu, který pracuje tak, že začne trackovat od nejvyššího patra pyramidy (nejnižší detail v obraze) a postupně se propracovává přes všechny patra v pyramidě až do nejnižších pater (největších detailů obrazu). To umožní metodě odhalit i větší pohyb a zachytit jej pomocí lokálních oken.

Předpoklady

Lucas-Kanade algoritmus je navržen tak, že vstupní hodnoty musí splňovat tyto parametry (znázorněny jsou také na obrázku 2.5)

- Jasová stálost ... Předpokládá se, že bod, který sledujeme, bude mít stejnou intenzitu v čase t i $t + 1$.
- Časová stálost ... Předpokládá se, že posun bodu mezi framy bude malý a přibližně konstantní.
- Prostorová spojitost ... Sousední body ve scéně náleží stejnému povrchu na reálném objektu.



Obrázek 2.5: Předpoklady pro Lucas-Kanade metodu. 1. Jasová stálost, 2. Časová stálost, 3. Prostorová spojitost. Převzato z [6]

Princip

Princip algoritmu vychází z dodržení všech předpokladů stanovených výše. Nejjednodušší bude ukázat princip metody pro jednu dimenzi. Pokud tedy platí první předpoklad, tak se pro námi trackovaný pixel x intenzita $f(x)$ v čase t nemění

$$\frac{df(x)}{dt} = 0.$$

Podle rovnice stálosti jasu je intenzita bodu x přímo závislá na čase t, tedy ve tvaru $I(x(t), t)$. Po aplikování řetízkového pravidla pro parciální derivace získáme rovnici

$$\left.\frac{dI}{dx}\right|_t \left(\frac{dx}{dt}\right) + \left.\frac{dI}{dt}\right|_{x(t)} = 0,$$

což je v podstatě totéž jako

$$I_x v + I_t = 0,$$

kde v je jednodimenzionální vektor posunu, který hledáme, I_x je prostorová parciální derivace přes první snímek a I_t je parciální derivace mezi snímky v závislosti na čase. Tato rovnice ovšem platí pouze pro jednu dimenzi. Pro dvě dimenze musíme pozměnit rovnici do tvaru:

$$I_x u + I_y v + I_t = 0,$$

kde u a v jsou složky dvou dimenzionálního vektoru posunu, který hledáme. Z toho ovšem vyplývá, že máme jednu rovnici pro jeden bod a v ní dvě neznámé. Zde přichází jedno omezení a to, že Lucas-Kanade algoritmus nefunguje pouze pro jeden bod, ale vyžaduje význačných bodů více.

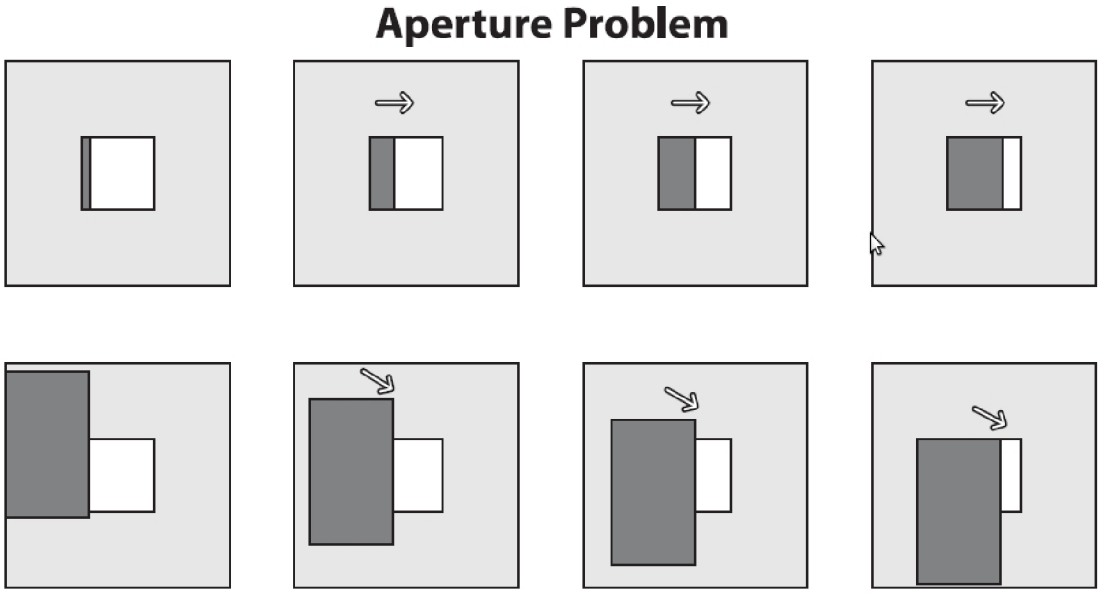
Pro každý význačný bod $q_1, \ldots, q_n$ jsme schopni poskládat podobnou rovnici, a tím pádem můžeme získat soustavu rovnic

$$\begin{aligned} I_x(q_1)u + I_y(q_1)v &= -I_t(q_1) \\ I_x(q_2)u + I_y(q_2)v &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)u + I_y(q_n)v &= -I_t(q_n). \end{aligned} \tag{2.11}$$

Po vyřešení získáme výsledný vektor optického toku. Rovnice se dají vyřešit některou z numerických metod například metodou nejmenších čtverců.

Proč potřebujeme více bodů než jen jeden je též možné odvodit pomocí tzn. "aperture problemu"(obrázek 2.6). Na něm je vidět, že pokud máme malé detekční okno, což Lucas-Kanade algoritmus má, nemůžeme bezpečně určit směr pohybu ve chvíli, kdy je detekovaný objekt výrazně větší než naše detekční okno. Nejčastěji z objektu uvidíme pouze jednu jeho hranu a podle ní, jak ukazuje obrázek, není možné korektně určit směr pohybu.

Více informací lze nalézt například v [6].



Obrázek 2.6: Pokud máme malé detekční okno přes které se pohybuje mnohem větší objekt, nejčastěji z něj vidíme pouze jednu jeho hranu. Z té však nejsme schopni zjistit, jakým směrem se objekt pohybuje. Převzato z [6]

Kapitola 3

Rozpoznávání lidských postav v obraze

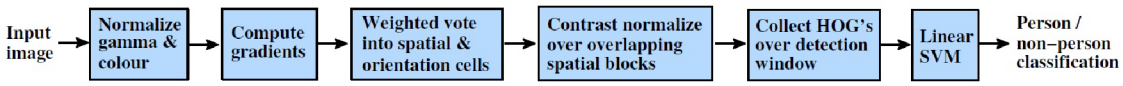
Předešlá kapitola byla o tom, jak správně detekovat jakýkoliv pohyb ve videu. Pokud nám informace o obecném pohybu nestačí, je načase naučit počítač rozpoznat, jaký objekt se to ve video sekvenci vlastně pohybuje. A k tomu přesně slouží tato kapitola. Snaží se ukázat dvě používané metody pro extrahování příznaků z obrazu a jejich rozdíly. V závěru kapitoly je obecná sekce o klasifikaci jednotlivých objektů podle právě extrahovaných příznaků.

3.1 Histogram orientovaných gradientů

Histogram orientovaných gradientů je v současnosti pravděpodobně nejlepší metodou na detekci lidí v obraze. Byl navržen v roce 2005 Navneetem Dalalem a Billem Triggsem uveřejněním jejich článku [7] na vědecké konferenci o počítačovém vidění ve Francii.

Z obecného pohledu je ke správnému rozpoznání objektu nutné získat vhodné příznaky a ty potom předat klasifikátoru. Příznaky jsou v metodě orientovaných gradientů zastoupeny tzv. HOG deskriptory (Histogram Orientovaných Gradientů).

Dekriptor se skládá z většího množství histogramů, které se počítají na základě gradientů spočtených pro každý pixel v obraze. Detailní popis bude následovat níže v této kapitole. Histogramy jsou určovány lokálními buňkami v obraze o dané velikosti. Jsou tak velmi málo náchylné například na změny globálního osvětlení.



Obrázek 3.1: Postup extrahování příznaků a detekce objektu v obraze pomocí HOG deskriptorů. Převzato z [7]

3.1.1 Postup rozpoznání

Na obrázku 3.1 je vidět postup detekce člověka na statickém obraze.

- Výpočet gradientů pro všechny pixely v obraze.
- Rozdělení obrazu do mřížky disjunktních buněk dané velikosti.

- Výpočet váženého histogramu orientací gradientů pro každou buňku.
- Rozdělení obrazu na překrývající se bloky dané velikosti obsahující daný počet buněk.
- Normalizace histogramů v rámci každého bloku.
- Kombinace těchto histogramů je poslána lineárnímu SVM klasifikátoru, který rozhodne, zda se jedná či nejedná o člověka.

3.1.2 Výpočet gradientu

Detekce hrany spočívá v nalezení výrazných lokálních změn v intenzitě sousedních pixelů v obraze. Tato změna je měřitelná pomocí gradientu obrazu.

Obraz můžeme definovat jako funkci $f(x, y)$, jejíž hodnotou je intenzita pixelu o souřadnicích x a y. Jelikož je funkce f dvoudimenzionální, je gradient G dvousložkovým vektorem

$$G = \begin{pmatrix} G_x \\ \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{df}{dx} \\ \\ \frac{df}{dy} \end{pmatrix}. \tag{3.1}$$

Velikost gradientu G je definována vztahem

$$|G| = \sqrt{G_x^2 + G_y^2}. \tag{3.2}$$

Směr gradientu G v bodě $[x, y]$ je definován jako

$$\theta(x, y) = \arctan\left(\frac{G_x}{G_y}\right). \tag{3.3}$$

Složka gradientu G_x se v bodě $[x, y]$ zjistí pomocí diskrétní konvoluce funkce obrazu $f(x, y)$ a funkce $h(x, y)$, které odpovídá konvoluční maska $[-1, 0, 1]$, tj.

$$G_x = -1 * f(x - 1, y) + 0 * f(x, y) + 1 * f(x + 1, y). \tag{3.4}$$

Pro výpočet složky gradientu G_y využijeme masky $[-1, 0, 1]^T$ tj. vzorce

$$G_y = -1 * f(x, y - 1) + 0 * f(x, y) + 1 * f(x, y + 1). \tag{3.5}$$

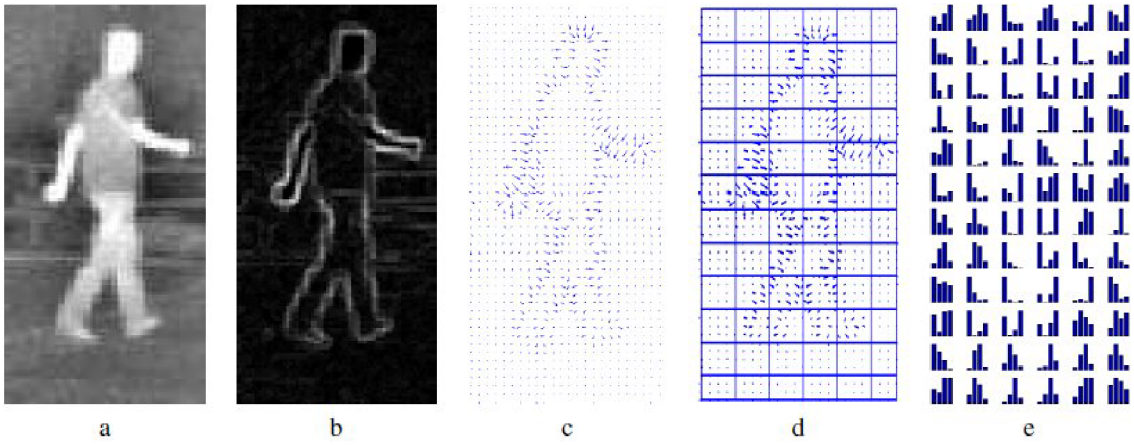
3.1.3 Určení lokálního histogramu

Nejprve je nutné obraz rozdělit hustou mřížkou na buňky o určité velikosti. Dalal s Triggsem v [7] ukázali, že nejlepších výsledků dosahuje algoritmus pro velikost buňky 8×8 pixelů.

Nyní řekněme, že máme rozdělený obraz na buňky o velikosti 8×8 a zároveň máme spočítaný gradient pro každý pixel v obraze.

Histogram je nutné určit pro každou buňku zvlášť. Jednotlivé kanály histogramu jsou určeny orientací (směrem) gradientů uvnitř buňky. Počet kanálů histogramu je libovolný, ale Dalal s Triggsem opět zjistili, že nejlepší rozdělení je do devíti kanálů v rozmezí buďto $0° - 180°$ nebo $0° - 360°$.

Velikost kanálů histogramu je určena velikostí (délkou) příslušných gradientů. Konkrétně se tedy prochází všechny gradienty v buňce a podle orientace se zařazují do příslušného kanálu. Velikost kanálu je pak navýšena o velikost daného gradientu, případně o hodnotu nějaké funkce závislé na velikosti gradientu (například druhá mocnina nebo odmocnina velikosti gradientu).



Obrázek 3.2: Postup rozpoznání člověka pomocí HOG deskriptorů. Převzato z [5]

3.1.4 Normalizace bloků

Abychom se co nejvíce vyhnuli vlivu globálních podmínek na výsledné rozhodování, je výhodné obraz znovu rozdělit do tzv. bloků dané velikosti a ty potom normalizovat.

Velikost bloku je opět libovolná, ale testy ukázaly, že výhodné je používat velikost 2×2 nebo 3×3 buněk (16×16 nebo 24×24 pixelů).

V rámci těchto bloků se bude provádět normalizace histogramů jednotlivých buněk. Je výhodné, aby se bloky vzájemně překrývali. Každá buňka pak bude přispívat do více než jednoho bloku.

Pro normalizaci je možné použít několik metod. Například

$$f = \frac{v}{\sqrt{||v||_2^2 + e^2}}, \tag{3.6}$$

kde v je nenormalizovaný vektor obsahující všechny histogramy v daném bloku, a proměnná e je nějaká malá konstanta.

3.1.5 Deskriptor

Vektor normalizovaných histogramů pro jeden blok nazýváme deskriptorem. Z každého bloku tak vyjde jeden deskriptor.

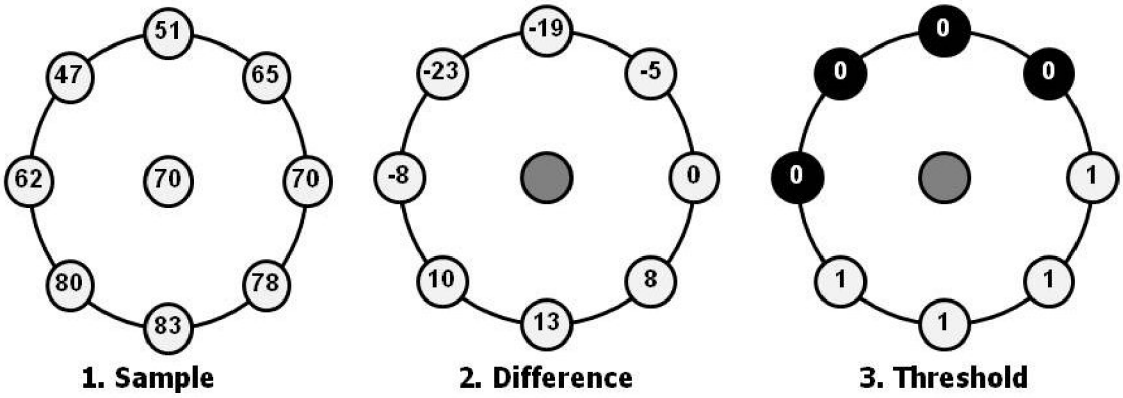
Posledním krokem v rozpoznávání je předat sadu deskriptorů nějakému klasifikátoru. Pro více informací o histogramech orientovaných gradientů bych rád odkázal na původní práci Dalala a Trigse [7].

3.2 Local Binary Patterns

LBP je jednoduchý a přesto účinný operátor vhodný pro extrahování příznaků z obrazu viz [12]. Pracuje tak, že prochází pixel po pixelu obrázek a prahuje jeho okolí na velikost prahu středového pixelu. Výsledek potom násobí konkrétní mocninou dvojky.

Na obrázku 3.3 je vidět postup zpracování okolí jednoho bodu.

1. Načte se pixel a jeho osmi-okolí.
2. Od každého pixelu z okolí se odečte hodnota středního pixelu.
3. Hodnoty, které jsou záporné, jsou nastaveny na 0. Kladné hodnoty na 1.



Obrázek 3.3: Postup zpracování okolí jednoho bodu metodou LBP. Převzato z [12]

Okolí bodu, ze kterého se LBP počítá, není fixní, ale je možné ho zvolit libovolně. Výpočet konečné hodnoty LBP pro intenzitu g_c pixelu $[x_c, y_c]$ pak vypadá následovně

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p, \tag{3.7}$$

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{3.8}$$

kde g_p je intenzita příslušného bodu v okolí, P je počet pixelů v okolí středového bodu a R je rádius okolí.

Konkrétně pro obrázek 3.3 by vypadal výpočet 3.7 takto

$$1 * 1 + 1 * 2 + 1 * 4 + 1 * 8 + 0 * 16 + 0 * 32 + 0 * 64 + 0 * 128 = 15.$$

Local binary patterns se dokonce dají úspěšně spojit dohromady s histogramy orientovaných gradientů (popsaných výše). O takovém spojení pojednává například [15].

3.3 Klasifikace

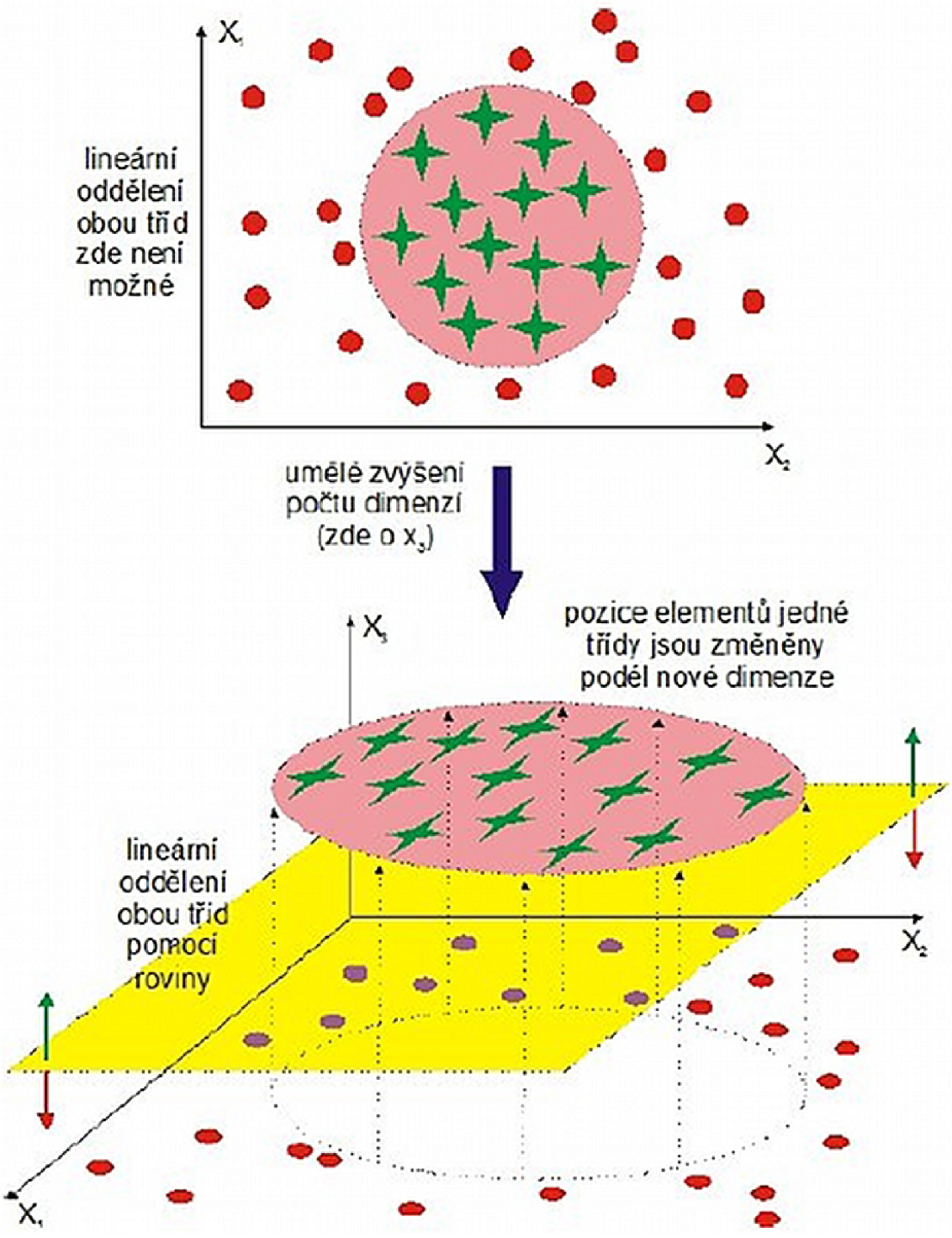
Klasifikace znamená doslova třídění. Je to postup, jak velké množství dat zařadit do různých tříd. Pokud bychom jako příklad uvedli rozpoznávání číslic, budeme mít přesně deset klasifikačních tříd, do kterých budeme obrázky číslic třídit.

V případě této práce se nám jedná především o rozhodnoutí, zda se jedná nebo nejedná o člověka. V tom případě nám stačí pouze dvě klasifikační třídy (Je člověk, Není člověk).

3.3.1 Lineární SVM

V případě, že máme pouze dvě třídy pro klasifikaci, nabízí se použití metody SVM (Support Vector Machines). Ta je ve své základní verzi binární, tedy rozděluje data do dvou množin. V prostoru všech příznaků extrahovaných příslušnou metodou hledá nadrovinu, která optimálně rozděluje trénovací data. Tato nadrovina je lineární funkcí v prostoru příznaků.

Linearita funkce pro nadrovinu je zachována i ve vícerozměrném prostoru příznaků díky tzv. jádrové funkci. Ta umožňuje nahradit původně lineárně neseparovatelnou úlohu za lineárně separovatelnou.



Obrázek 3.4: Princip vzniku možnosti lineárního oddělení dvou tříd s nelineárními hranicemi pomocí přidané dimenze. Převzato z [2]

Kapitola 4

Návrh aplikace pro detekci osob ve video sekvenci

Pro efektivní návrh aplikace je nejprve nutné definovat si cíle, které má aplikace splňovat. Cílem práce je detekce lidských postav ve videu a tak bych rád, aby aplikace splňovala tyto body.

- Rychlá a pokud možno přesná detekce lidí ze stacionární kamery
- Minimální závislost detekce na vlivu změny osvětlení a stínů.
- Možnost zvýraznění trajektorie pohybu každého člověka v záběru
- Zobrazení přesnější kontury člověka
- Nezávislost aplikace na systémové platformě

4.1 Obecná idea průběhu detekce

Existuje několik možností, jak navrhnout aplikaci pro sledování osob ve video sekvenci.

Nejlepší cestou pro detekci lidí ve statickém obraze je v dnešní době metoda využívající histogramy orientovaných gradientů, viz výše.

Ve videu už tato oblast tak jistě pokrytá není. Jedním z prvních návrhů by samozřejmě mohlo být postavit detektor na výše zmíněných HOG deskriptorech a tuto metodu aplikovat na každý obrázek ve video sekvenci. V tom případě bychom téměř jistě narazili na výkonostní problém. I když je metoda orientovaných gradientů jednou z rychlejších, implementovat ji tak, aby zvládala 25 snímků za vteřinu u libovolně velkého videa by bylo prakticky nereálné. Dalalova a Triggsova metoda ([7]) zvládá video o rozměrech 320×240 při 1 FPS (frame per second).

Hlavně z tohoto důvodu se nabízí jiný pohled na design aplikace. Bylo by možné použít některou z metod rychlé detekce pohybujících se objektů a až na zachycené objekty pustit klasifikátor využívající HOG deskriptory. Tím bychom se vyhnuli nutnosti aplikovat rozpoznávání na celý snímek videa. A v zároveň by nebylo nutné spouštět rozpoznávání na každý snímek videa. V krajní možnosti by se mohlo rozpoznání spustit pouze v době, kdy objekt vstoupí do obrazu. Pomocí metody pro sledování pohybu by byl objekt označen, dokud by neopustil scénu.

Tento "high level"postup lze shrnout do několika kroků, které bude nutné dodržet.

- Vytvoření modelu pozadí
- Detekce pohybu a zjištění oblastí pohybu (ROIs)
- Rozpoznání nalezeného objektu pomocí HOG deskriptorů
- Sledování rozpoznaného objektu (lidské postavy)
- Zanamenávání trajektorie
- Vykreslení výsledků

4.2 Konkrétní návrh detekce

Pro konkrétní návrh aplikace je nutné si vybrat specifické metody a vědět, jak se dají využít v praxi. Nezbytnou součástí návrhu by mělo být, jak bude program fungovat a kdy se které metody budou volat. Navrhované řešení lze shrnout do těchto bodů

- Načtení videa
- Inicializace detektoru, trackeru a background modelu
- Pro každý snímek videa
  - Upgrade backgroung modelu
  - Získání oblastí s pohybem
  - Rozpoznání lidských postav v těchto oblastech
  - Přidání nových rozpoznaných postav do trackeru
  - Update současných osob v trackeru
  - Sledování nalezených osob
  - Vykreslení výsledků

4.2.1 Načtení videa

Video je potřeba dekódovat a získat z něj jednotlivé snímky tak, aby bylo možné s nimi pracovat. Nabízí se dva možné přístupy. Prvním by bylo dekódovat video hned při načtení a snímky si uložit do paměti. Ovšem při větší velikosti videa a vetším počtu snímků bychom rychle zaplnili celou paměť. Je tedy nutné video načítat postupně snímek po snímku. V paměti se tedy bude uchovávat vždy jenom aktuální a předchozí snímek. Na aktuálním snímku se bude provádět detekce a rozpoznávání. Předchozí snímek bude zapotřebí k trackování nalezených osob.

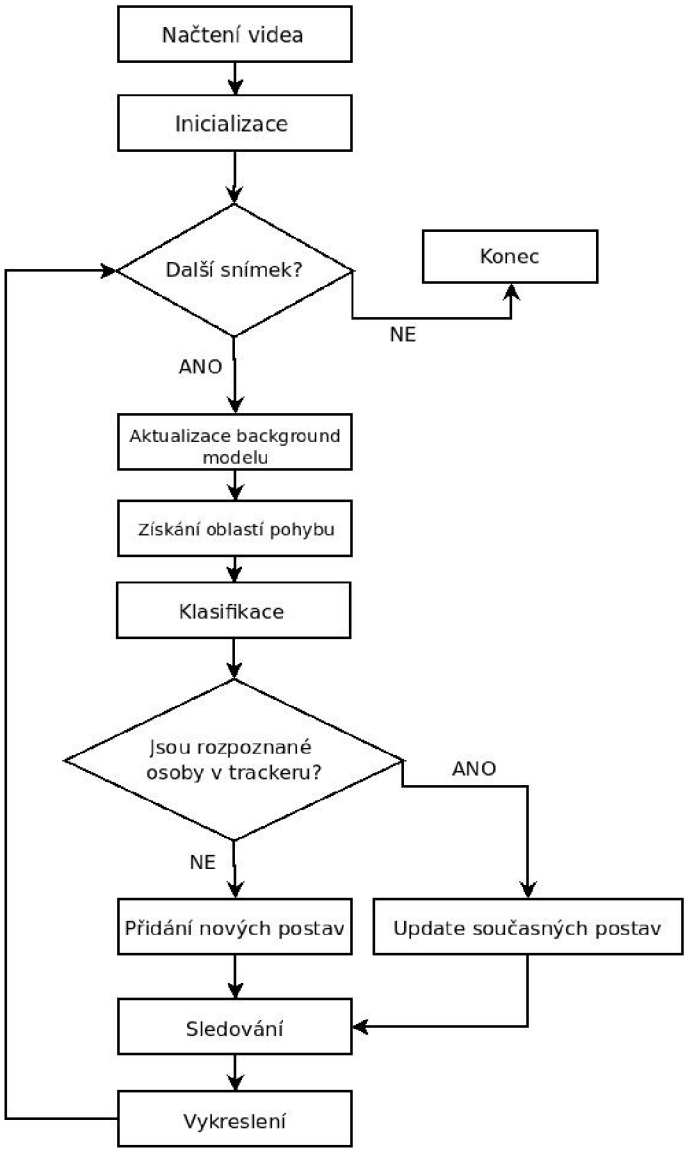
4.2.2 Vytvoření detektoru, trackeru a background modelu

Obecný pohyb se bude detekovat pomocí modelu pozadí (background modelu). Oblasti s aktivním pohybem vzniknou tak, že se od aktuálního snímku odečte snímek pozadí. Pozadí se v modelu musí neustále aktualizovat, čímž se částečně vyhneme reakcím na změnu vlivu globálního osvětlení.

Konkrétní model pozadí bude využívat metodu Gaussova průměru (viz 2.2.5), protože vytváří dostatečný model a zároveň je velmi rychlá.

Model pozadí se musí inicializovat pomocí prvního snímku videa. Další snímky ho pak budou postupně aktualizovat.

Stejně tak detektor a tracker se musí inicializovat hodnotami získanými z videa, především velikostí framu. Většina výpočetně náročných operací se neprovádí na celých snímcích, ale jenom na jejich výřezech. Je tedy nutné, aby jak detektor, tak tracker znaly rozměry původního videa a uměly zpětně přepočítat souřadnice nalezených resp. sledovaných objektů vzhledem v celému snímku (viz níže v rozboru detekce pro jednotlivé snímky).



Obrázek 4.1: Návrh aplikace pro detekci osob ve video sekvenci. Program flow.

4.2.3 Pro každý snímek videa

Následující kroky je nutné provést s každým snímkem videa.

Upgrade backgroung modelu

Model pozadí se počítá pomocí Gaussovy funkce z několika předchozích snímků. Ty si můžeme představit jako pole nebo zásobník. Při aktualizaci prostě vložíme nový snímek na vrchol zásobníku.

Získání oblastí s pohybem

Po odečtení aktuálního snímku a modelu pozadí získáme aktivní oblasti ve videu. Získáme vlastně masku, která určuje, kde na snímku došlo k pohybu. Z této masky odstraníme šum pomocí Gaussova filtru a vyprahujeme ji tak, abychom získali binární obraz.

Nyní máme masku znázorňující pohybující se objekty. Klíčové bude podle získané binární masky rozdělit vstupní obraz na několik menších tak, aby se jednotlivé obrazy daly poslat rozpoznávací funkci. Nejjednodušším způsobem je nalézt ohraničující obdélníky (bounding boxy) okolo všech kontur, které jsou v binární masce. Pomocí těchto obdélníků následně rozřežeme původní snímek a výsledné malé obrazy pošleme rozpoznávací funkci. Jelikož nás zajimají ucelené oblasti pohybu, bude nutné jednotlivé bounding boxy sjednotit, pokud někde dojde k jejich překryvu. Docílíme tím toho, že budeme předávat rozpoznávací funkci vždy celý objekt a ne jenom jeho část.

Rozpoznání lidských postav v těchto oblastech

Nejlepší metodou pro rozpoznání lidských postav se jeví histogramy orientovaných gradientů (viz 3.1). Jejím hlavním úkolem bude nalézt v jednotlivých obrazech lidské postavy a potvrdit tak, že pohybující se oblasti nalezené pomocí background modelu jsou skutečně lidé.

Přidání nově rozpoznaných postav do trackeru

Tracker si v sobě musí udržovat seznam všech lidí, které je potřeba sledovat. Každá nově rozpoznaná lidská postava musí být přidána do sledovací databáze. To ovšem není tak jednoduché, jak by se mohlo na první pohled zdát. Není možné přidávat do seznamu všechny nalezené postavy z důvodu duplicity. Pokud by například rozpoznávací funkce na pěti framech videa našla stále stejnou osobu, byla by v trackeru hned pětkrát. To je nežádoucí a je proto nutné zajistit, aby daná osoba byla v systému právě jednou.

Postavy jsou sledovány pomocí význačných bodů nalezených na jejich těle (features). Ve chvíli kdy je nalezena potenciální nová osoba, je nutné projít celý seznam již sledovaných osob a zjistit zda nově nalezený bounding box neobsahuje řekněme 80% význačných bodů ostatních lidí. Pokud tomu tak je, pak se s nejvetší pravděpodobností jedná o již sledovanou osobu a není nutné ji do systému přidávat.

Update současných osob v trackeru

Pokud se rozpoznaná osoba shoduje s nějakou již sledovanou, měla by být sledovaná osoba aktualizována podle nově rozpoznaného bounding boxu. Zároveň by jí měly být vygenerovány nové sledovací body.

Sledování nalezených osob

Při sledování (trackování) je důležitá správná volba trackovacího algoritmu. V tomto případě padla volba na Lucas-Kanade Tracker (viz 2.3.1). Ten je schopen najít v současném snímku body vyznačené v předchozím snímku. Čili pokud v předchozím snímku naleznu osobu a její význačné body, Lucas-Kanade tracker je schopen mi říct, kde se tyto body nacházejí ve snímku současném. Toho můžeme využít právě při sledování pohybujích se objektů. Kdy Lucas-Kanade tracker bude v podstatě aktualizovat a posouvat (sledovat) význačné body jednotlivých lidských postav ve videu.

V nejjednodušším případě můžeme postavy sledovat v rámci celého framu. Takové řešení je ovšem velmi náročné na výkon a v podstatě zbytečné. Jako forma optimalizace a urychlení procesu sledování se nabízí stejná možnost jako v případě rozpoznávání. Tedy nechat význačné body sledovat pouze v menším vyříznutém okně. Takové okno může být sestrojeno v závislosti na bounding boxu sledované osoby. Samozřejmě je nutné přidat k oknu rozumně velký okraj, protože daná osoba nezůstane na místě a může se pohnout kterýmkoliv směrem. Velikost přidaného okraje závisí na rychlosti sledované osoby, ale vzhledem k rychlosti videa dvacet pět snímků za vteřinu se dá předpokládat, že se objekt z předchozího framu nepřemístí na současný o více než o svoji čtvrtinu. V tomto ohledu by se dalo zajisté dále optimalizovat a odhadovat například směr rozšíření sledovacího okna v souvislosti s předchozím pohybem, ale to je jen možnost optimalizace výkonu, která nemá s funkčností nic společného.

Ve chvíli, kdy známe nové souřadnice všech význačných bodů pro jednu osobu, je nutné ještě aktualizovat její bounding box. Jeho velikost zůstane stejná, protože nedošlo k žádné detekci nebo rozpoznání, ale jeho pozice se musí posunout, stejně jako se posunuly body uvnitř. Jednou možností by bylo vypočítat, o kolik a jakým směrem se posunul každý bod a z průměru všech bodů vypočítat, kam by se měl posunout bounding box. Toto řešení by se dalo použít například pro sledování automobilů, protože tam se pohybuje objekt jako celek. Ovšem v případě lidí tomu tak není. Různé části těla se pohybují různými směry, a tak se taková metoda nedá považovat za spolehlivou. Lepší možností je využít metodu na hledání středu shluku bodů a do tohoto středu umístit následně střed bounding boxu. Takovou metodou je například K-Means.

Poslední vlastností, kterou zajišťuje tracker, je zaznamenávání trajektorie pohybu dané osoby. Trajektorie bude spojnice bodů, kterými osoba prošla za celou dobu, kdy byla sledována. Pokaždé když tracker posune bounding box dané postavy, zaznamená si střed tohoto obdélníku a z těchto bodů nakonec zrekonstruuje výslednou trajektorii.

Vykreslení výsledků

Vykreslit bude potřeba obdélníky kolem jednotlivých postav a trajektorii jejich pohybu. Každá postava by měla být vykreslena jinou barvou, aby bylo na první pohled vidět, kudy se pohybuje.

Kapitola 5

Implementace

Kapitola o implementaci obsahuje zachycení skutečného stavu aplikace. Měla by obsahovat informace o tom, jak byl reálně zpracován návrh popsaný výše. Jestli nebylo nutné se někde ochýlit od cesty navrhované v předešlé kapitole a zároveň jestli byly splněny všechny body návrhu či nikoliv.

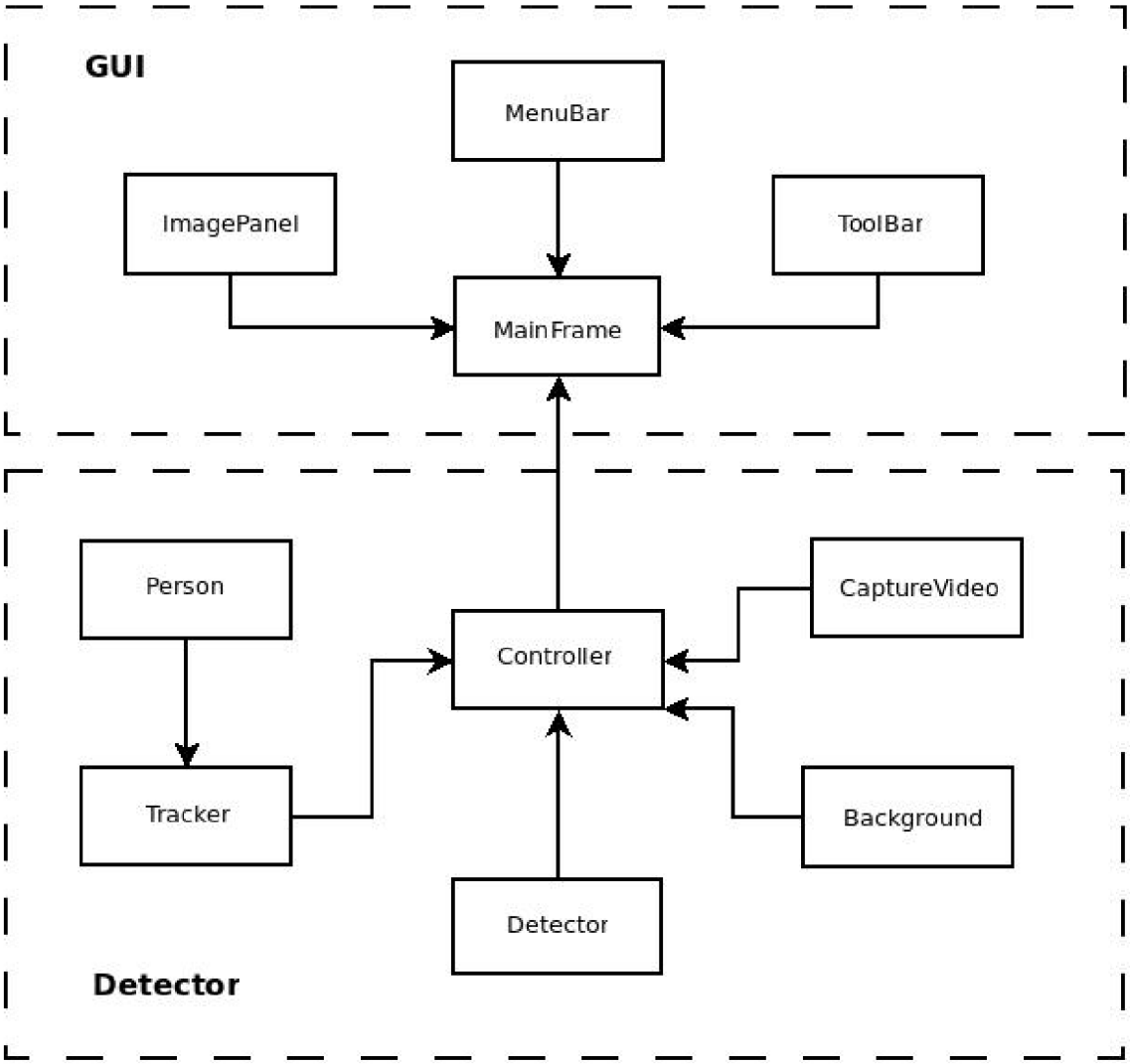
5.1 Použité technologie

Nejprve bych rád shrnul všechny implementační a vývojářské technologie, které jsem použil.

- C++ 4.5
  Výběr jazyka byl jasný a daný už v zadání diplomové práce. Nicméně jsem se v aplikaci snažil co nejvíce využívat potenciál objektově orientovaného C++.
- OpenCV 2.2
  Abych nemusel všechny v návrhu popisované algoritmy implementovat sám, používám knihovnu OpenCV verze 2.2, která obsahuje všechny stěžejní algoritmy pro moji práci.
- WxWidgets 2.8
  WxWidgets používám pro vytvoření GUI aplikace. Volba na wxWidgets padla především kvůli nezávislosti na platformě. Čili i když vývoj probíhal kompletně na Linuxu, je možné aplikace přeložit i pod operačním systémem Windows.
- Eclipse 3.6
  Vývojové prostředí Eclipse je dle mého názoru nejlepší vývojové prostředí pro Javu a C++. Nabízí například možnost automatického generování Makefilu a debugování kódu.
- Git
  Verzování kódu je nutností a myslím, že je vhodné jej tu minimálně zmínit. Rozhodnutí mezi subversion a gitem bylo jednoduché. Git nabízí o mnoho více možností a je lokální. Nepotřebuje žádný server, dokáže verzovat jakékoliv soubory ve složce.

5.2 Třídy aplikace

Rozdělení do jednotlivých tříd ukazuje názorně obrázek 5.1.



Obrázek 5.1: Diagram komponent

5.2.1 Modul detektoru

Detektorem se rozumí pomyslné jádro aplikace - její funkční jádro. Je složené z několika tříd, které se o funkcionalitu starají. Srdcem detektoru je třída Controller. Na ní navazují ostatní moduly a třídy.

Controller

Controller se stará o hlavní funkcionalitu. Řídí ostatní moduly a předává hodnoty mezi nimi. Uvnitř controlleru běží hlavní smyčka detekce lidských postav. Umožňuje zastavení a spuštění videa.

Objekt controlleru je vytvořen při startu aplikace a čeká, dokud uživatel nevybere video, ve kterém chce spustit detekci. Video je načteno pomocí třídy CaptureVideo. V tu chvíli se zároveň vytvoří objekty tříd Tracker, Detector a Background a inicializují se pomocí prvního snímku videa. První snímek je přitom vykreslen na obrazovku.

Následně se opět čeká na uživatele až celou akci odstartuje. Jakmile se tak stane, spustí

se uvnitř controlleru cyklus, který probíhá pro každý snímek videa přesně podle návrhu 4.2.3.

Background

Třída background dělá v podstatě prostředníka mezi controllerem a background modelem z OpenCV. Využívá se třídy CvBGStatModel, která je vytvořena funkcí cvCreateGaussianBGModel a prvním snímkem videa. Další snímky pak slouží k aktualizaci modelu pomocí funkce cvUpdateBGStatModel(...)

Model si v sobě udržuje aktuální snímky pozadí i popředí. Takže pokud potřebujeme vědět, co se ve snímku pohnulo, potřebujeme snímek popředí.

Kvůli odstranění šumu a vytvoření binární masky se ve třídě background provádí ještě gaussovo filtrování a prahování výstupu z background modelu.

CaptureVideo

Načítání videa náleží třídě CaptureVideo. Jejím úkolem je udržovat v sobě načtený video soubor a na vyžádání controlleru vrátit jeho správný snímek. K tomu využívá třídu CvCapture a metodu cvCaptureFromAVI(...). Stará se taky o správné převedení snímku do odstínů šedi. Udržuje v sobě současný a předchozí snímek videa. Při načítání videa si z něj vytáhne rovněž FPS (počet snímků za vteřinu) a šířku s výškou videozáznamu.

Detector

Detektor pracuje na principu histogramu orientovaných gradientů. Primárně využívá třídu HOGDescriptor naimplementovanou v OpenCV.

Na začátku dostane frame videa a sadu obdélníků vzešlých z background modelu a z obecné detekce pohybu. V těchto obdélnících dochází k nějakému pohybu a na detektoru je, aby rozpoznal lidské postavy v těchto oblastech.

Pro každý obdélník na vstupu provede detektor oříznutí původního framu. Následně vzniklý obraz pošle funkci detectMultiScale(...) ze třídy HOGDescriptor. Pokud se v konkrétní části framu vyskytovaly lidské postavy, vrátí funkce sadu obdélníků kolem těchto postav.

Nová sada detekovaných obdélníků putuje do Trackeru.

Tracker

Nejdůležitějším posláním Trackeru je sledování lidí. K tomuto účelu si v sobě udržuje aktualizovanou databázi resp. vektor všech lidí, kteří jsou určeni pro trackování (třída Person). Tracker obsahuje funkce pro vyhledávání a přidávání lidí do vektoru. Jeho hlavní funkcí je ovšem funkce, která předpovídá výskyt dané osoby v aktuálním snímku. Tato hlavní trackovací funkce je postavena na Lucas-Kanadeho trackeru implementovaného v OpenCV metodou cv::calcOpticalFlowPyrLK(...).

OpenCV zároveň nabízí funkci cv::goodFeaturesToTrack(...), které je využito ve chvíli, kdy je nový člověk přidáván do vektoru. V tu chvíli se najdou vhodné body k trackování a uloží se do třídy Person.

Person

Třída Person působí jako kontejner pro informace o sledované osobě. Obsahuje jak samotný bounding box okolo postavy, tak i všechny význačné body, které jsou sledovány v Trackeru. Ve chvíli, kdy detektor najde lidskou postavu a tracker usoudí, že takovou osobu ještě nesleduje, vytvoří novou instanci třídy Person. Uloží do ní bounding box, který našel detektor, a vytvoří význačné body.

Ve třídě Person jsou zároveň informace o tom, jak dlouho je objekt sledován a jaká je jeho trajektorie.

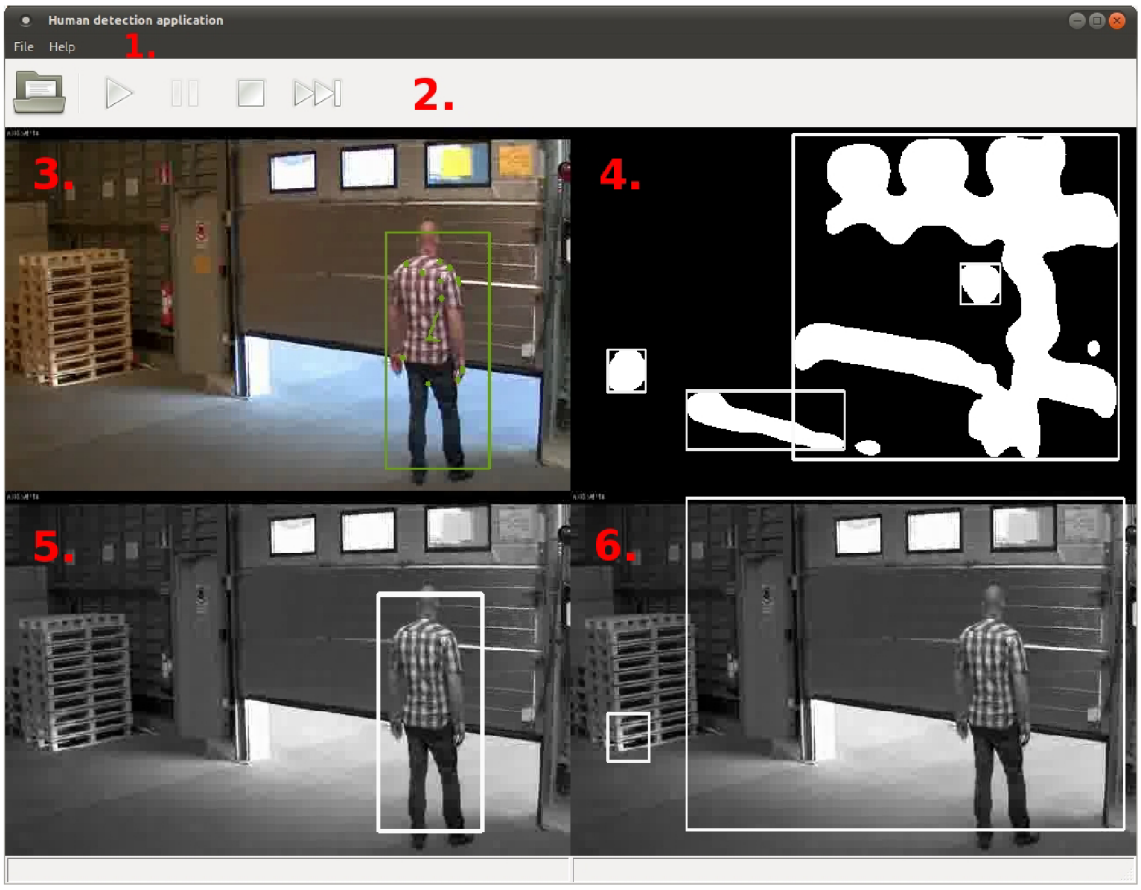
Kromě dat je ve třídě Person ukryta funkce pro aktualizaci bounding boxu, která posouvá střed obdélníku do středu shluku význačných bodů. Ten je zjištěn pomocí algoritmu K-means, tedy funkce cv::kmeans(...)

5.2.2 Modul uživatelského rozhraní

Uživatelské rozhraní bylo navrženo s ohledem na maximalní jednoduchost. Skládá se ze čtyř tříd

- MainFrame ... Hlavní okno, které obsahuje všechny ostatní prvky.
- ImagePanel ... Panel pro zobrazení videa.
- ToolBar ... Panel s tlačítky pro ovládání.
- MenuBar ... Hlavní menu aplikace.

Na obrázku 5.2 jsou všechny elementy uživatelského rozhraní vidět. ImagePanel je zde umístěn hned čtyřikrát, protože v každém se zobrazuje jiný obraz.



Obrázek 5.2: GUI aplikace. 1. MenuBar, 2. ToolBar, 3. ImagePanel s výsledkem Trackování, 4. ImagePanel s výstupem z Background modelu, 5. ImagePanel s výstupem z HOG Detektoru, 6. ImagePanel s obecnou detekcí pohybu

Kapitola 6

Testy a experimentální výsledky

6.1 Úspěšnost aplikace v závislosti na počtu sledovaných osob

Aby byly vidět i dílčí výsledky, rozhodl jsem se nejprve otestovat jednotlivé metody zvlášť a až potom otestovat výsledek jako takový. Mým cílem je ukázat, že jednotlivě tyto metody nemusí dávat nijak oslnivé výsledky, ale propojené dohromady dokáží vytvořit funkční aplikaci.

Nejprve ukáži testování klasifikační metody, potom sledovací metody a nakonec přijdou celkové výsledky programu.

V jednotlivých testech se snažím přikládat co nejvíce obrázků s děním ve videu, ale není možné zde uvést vše. Proto bych rád odkázal na přiložené CD, kde je možné nalézt většinu testovacích videí v elektronické podobě.

Testy byly prováděny experimentálně převážně na videu s následujícími parametry

Délka	00:03:46
Počet framů	5650
FPS	25
Rozlišení	1280×720
Kodek	ffmpeg2
Celkový počet osob	58
Výskyty osob na všech snímcích	7893

Tabulka 6.1: Vlastnosti testovacího videa

6.1.1 Výsledky rozpoznávání

Testování rozpoznávací metody bylo prováděno tak, že se rozpoznávalo na každém framu ve videu. Na každém snímku se spustila rozpoznávací metoda tzn. HOG deskriptory. Její výstup byl porovnán s anotovanými daty a vyhodnocena úspěšnost detekce.

Anotovanými daty se rozumí informace o tom, kde se přesně nachází lidské postavy na každém snímku videa. V mém případě se jednalo o textový soubor, který na řádku obsahuje číslo framu a souřadnice bounding boxů okolo všech lidských postav v daném snímku videa.

Úspěšnost rozpoznání μ se spočítá jako

$$\mu = \frac{\sum_{n=1}^{N} H_n}{\sum_{n=1}^{N} R_n}, \tag{6.1}$$

kde N je celkový počet snímků, H_n je počet správně rozpoznaných osob na n-tém snímku a R_n skutečný počet osob na daném snímku.

Nastavení HOG deskriptorů jsem nechal tak, jak doporučuje Dalal a Triggs v [7], tj.

Velikost bloku (block size)	16×16 pixelů
Velikost buňky (cell size)	8×8 pixelů
Detekční okno (detection window size)	64×128 pixelů
Překrytí (stride)	8×8 pixelů

Tabulka 6.2: Nastavení HOG deskriptorů

Počet nerozpoznaných osob (false negative) se určí jako rozdíl celkového počtu osob a správně rozpoznaného počtu osob.

Počet špatně detekovaných osob (false positive) je nutné spočítat podobným způsobem jako počet správně detekovaných osob. Pokud HOGy najdou osobu, která nepatří do anotovaných dat, pak se jedná o špatné rozpoznání.

Výsledky rozpoznávání jsou patrné z následující tabulky

	Celkem osob	Rozpoznáno	False negative	False positive	Úspěšnost
1 osoba	406	173	233	9	42,61%
2-3 osoby	369	154	215	2	41,73%
Skupiny	3381	1309	2072	239	38,72%
Celkem	4156	1636	2520	250	39,36%

Tabulka 6.3: Počet nalezených postav a celková úspěšnost rozpoznání.

Z tabulky 6.3 jasně vyplývá, že nejlepších výsledků je dosaženo pro jednotlivé osoby. Dvojice osob mají nižší úspěšnost převážně kvůli tomu, že ve videu jdou většinu své cesty v zákrytu. Tímpádem buď nejsou rozpoznány vůbec nebo jsou rozpoznány jako jedna osoba.

Dle dokumentace k OpenCV je implementace metody histogramu orientovaných gradientů stejná jako v původní práci Dalala a Trigse [7]. Pro bližší testy této metody tedy odkážu na jejich původní práci.

Obrázek 6.1: Vlevo: Správně rozpoznaná postava, Vpravo: Vzdálenější postava nerozpoznána.

6.1.2 Výsledky sledování

Při testování rozpoznávání bylo možné počítat jednotlivé snímky, na kterých se vyskytovaly osoby. U sledování to již není tak triviální.

Základní schopností trackeru je, že dokáže sledovat jakýkoliv objekt, v našem případě lidské postavy, po celou dobu jejich výskytu ve videu. Tedy od jejich úspěšného rozpoznání až do té doby, dokud neopustí scénu.

Pokud bychom chtěli otestovat čistě jenom tracker, musíme testovat právě tuto jeho vlastnost.

Rozpoznávání bude sloužit pouze k počátečnímu nalezení osoby. Zbytek bude mít na starosti tracker. Výsledek se bude určovat podle toho, kolik osob nalezených ve videu dokáže tracker správně "odsledovat"až do okamžiku, kdy opustí scénu.

Pro jednu osobu může tracker skončit třemi možnými způsoby

- Sledování bylo úspěšné a skončilo ve chvíli, kdy osoba opustila scénu.
- Sledování nebylo úspěšné. Tracker ztratil význačné body sledované osoby (viz obrázek 6.3).
- Sledování nebylo úspěšné. Tracker pro novou osobu použil význačné body jiné osoby.

Výsledky sledování jsou patrné z následující tabulky

	Úspěch	**Ztraceno**	**Převzetí**	**Úspěšnost**
1 osoba	10	15	0	40%
2 osoby	4	11	0	26,66%
3-8 osob	3	10	0	23,08%
Celkem	17	36	0	32,08%

Tabulka 6.4: Počet správně a špatně sledovaných osob

Obrázek 6.2: Vlevo dole špatně rozpoznaná lampa místo člověka, vpravo nerozpoznané dvě postavy.

6.1.3 Celkové výsledky

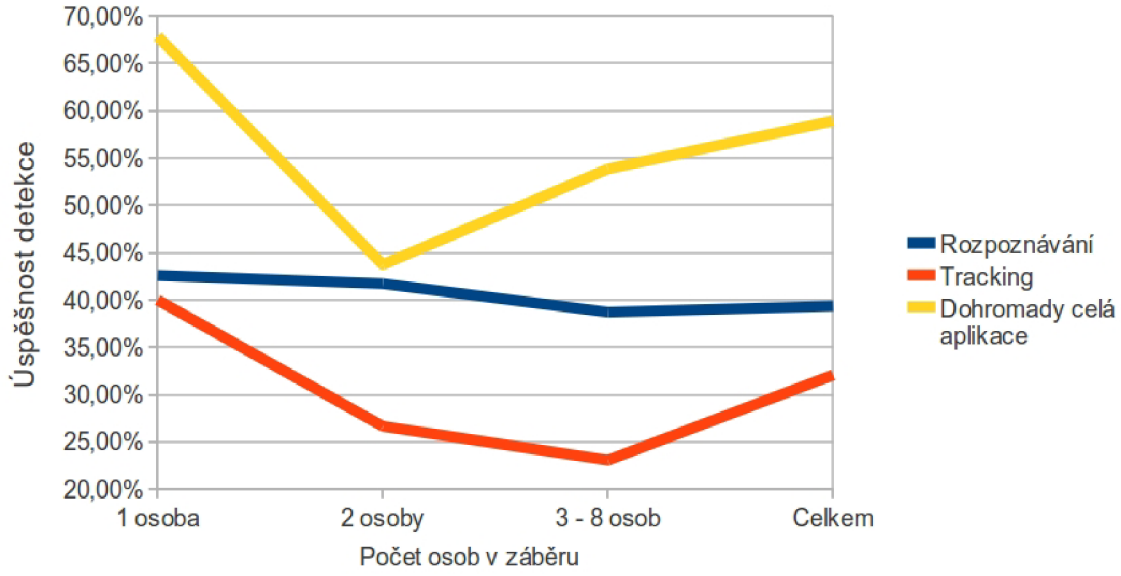
Celkové výsledky spojují detekci s trackováním a ukazují, že s použitím aktualizace trackeru v průběhu sledování, lze docílit výrazně lepších výsledků než s použitím jednotlivých metod zvlášť.

Výsledky sledování jsou patrné z následující tabulky

	Úspěch	**Ztraceno**	**Převzetí**	**Úspěšnost**
1 osoba	19	8	1	67,86%
2 osoby	7	7	1	43,75%
3-8 osob	7	6	0	53,85%
Celkem	33	21	2	58,93%

Tabulka 6.5: Počet správně a špatně sledovaných osob

Obrázek 6.3: Ztráta význačných bodů. Chyba trackeru.



Obrázek 6.4: Závislost úspěšnosti sledování na počtu osob

6.2 Výpočetní náročnost v závislosti na velikosti vstupního videa

Testování výkonu aplikace se dá měřit pomocí snímků za sekundu - FPS (frames per second). Výkon může ovlivnit několik faktorů

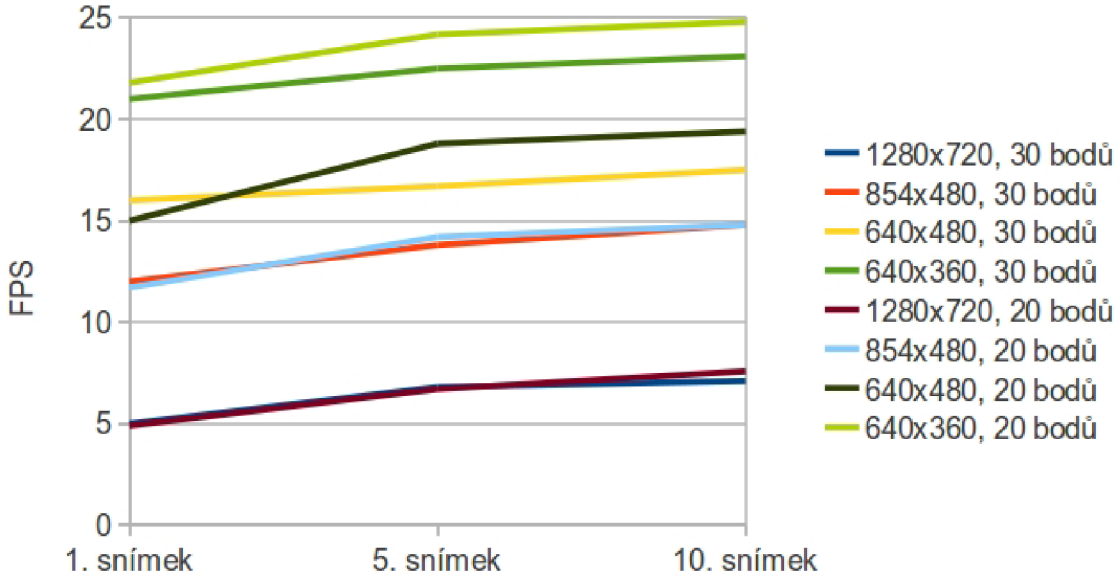
- Velikost vstupního videa resp. rozměry snímku videa.
- Vnitřní nastavení programu, tj. po kolika snímcích se bude pouštět detekce, kolik význačných bodů se bude sledovat.
- V neposlední řadě výkon počítače.

Výkon byl testován na stoji v konfiguraci Intel Core2Duo E8400 3,0GHz, 4 GB RAM, OS Linux 64-bit.

Následující tabulka uvádí výsledky pro různé velikosti videa a různá nastavení aplikace

HOG každý	**1. snímek**	**5. snímek**	**10. snímek**
1280 × 720 **30 bodů**	5	6,8	7,1
1280 × 720 **20 bodů**	4,9	6,7	7,58
854 × 480 **30 bodů**	12	13,8	14,8
854 × 480 **20 bodů**	11,7	14,2	14,8
640 × 480 **30 bodů**	16	16,7	17,5
640 × 480 **20 bodů**	15	18,8	19,4
640 × 360 **30 bodů**	21	22,5	23,1
640 × 360 **20 bodů**	21,8	24,18	24,8

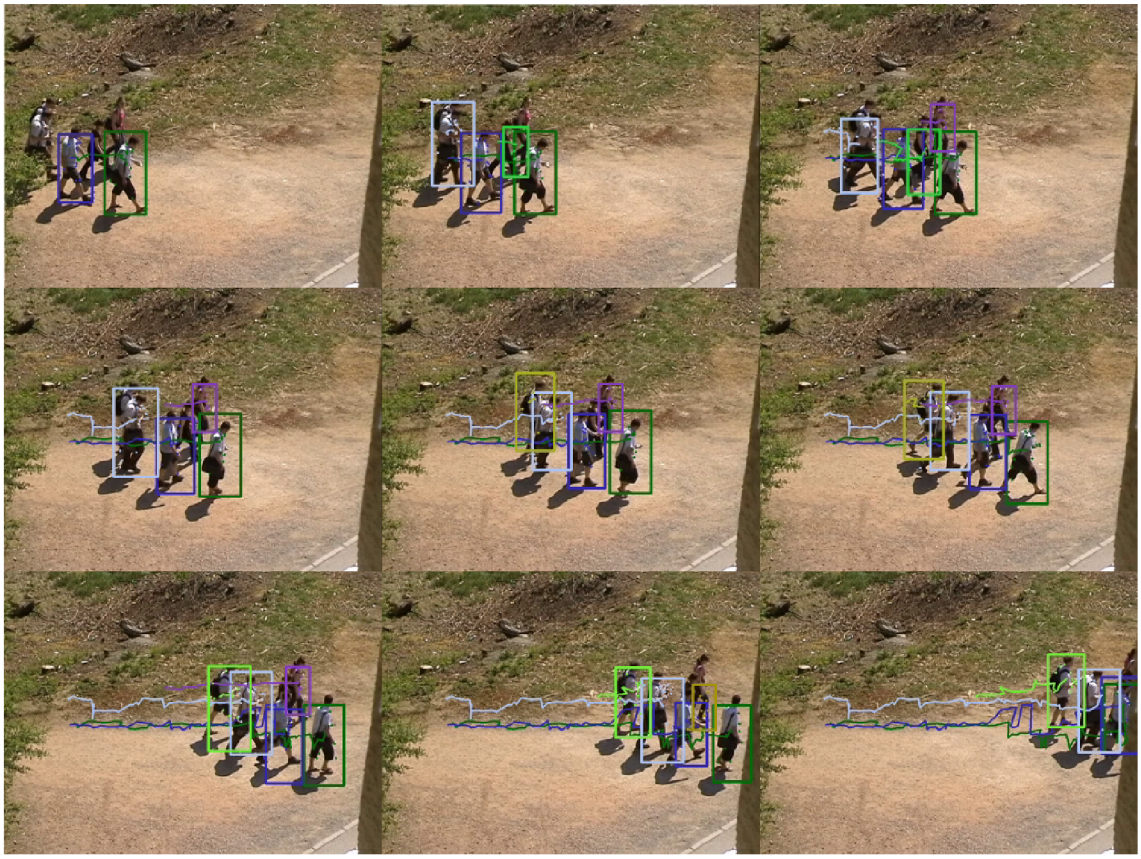
Tabulka 6.6: Náročnost aplikace [FPS] v závislosti na velikosti videa. Sloupce určují na kolikátém snímku bylo spuštěno rozpoznávání. Řádky se od sebe liší počtem význačných bodů (20 nebo 30) sledovaných pomocí trackeru



Obrázek 6.5: Celková náročnost aplikace [FPS]

6.3 Příklady sledování

Obrázek 6.6: Příklad sledování dvou osob



Obrázek 6.7: Příklad sledování skupiny lidí (konkrétně šesti)

Obrázek 6.8: Sledování jednotlivce proti pohybujícímu se pozadí

Kapitola 7

Závěr

Cílem diplomové práce bylo prozkoumat problematiku detekce pohybujících se objektů ve video sekvenci a navrhnout aplikaci, která využije nejlepších nalezených metod.

Konkrétním zaměřením byla detekce a rozpoznání lidských postav, a proto i výběr metod byl situovaný tímto směrem. Při hledání podkladů pro tuto práci se mi opakovaně potvrdilo, že nejlepší metodou pro klasifikaci lidských postav ve statickém obraze jsou histogramy orientovaných gradientů. Na této metodě je také postavena navržená aplikace.

Z testů však jasně vyplývá, že pokud chceme funkční aplikaci na sledování osob, nemůžeme se spolehnout pouze na detekční a rozpoznávací metody. Potřebujeme trackovací metodu, které předáme rozpoznaný obraz a budeme sledovat jeho pohyb v obraze. Výběr sledovací metody padl na Lucas-Kanadeho algoritmus, převážně kvůli rychlosti a možnosti sledovat složitější objekty.

Výsledky rozpoznávací metody, konkrétně histogramu orientovaných gradientů, nejsou nijak oslnivé, což přisuzuji především nedostatečné trénovací sadě. Pokud bych v práci pokračoval, byla by to první věc, která by se musela vylepšit. Jako trénovací sadu jsem používal tu, která je již naimplementována do knihovny OpenCV a ta je podle všech mých výsledků nedostatečná. Vychází totiž z databáze MIT [10] a ta není příliš obsáhlá. Pro další vývoj bych rozhodně volil rozsáhlejší databázi trénovacích dat, například sadu INRIA ([1]).

Rychlost aplikace je taktéž naznačena v testech (konkrétně v 6.2). Z grafů vyplývá, že se podařilo dosáhnout zpracování v reálném čase pouze pro nejmenší videa s rozlišením 640×360. Naopak u HD videa s rozlišením 1280×720 je rychlost na prahu pěti snímků za sekundu.

Možností dalšího vývoje je mnoho. První je definitivně již zmíněná lepší trénovací sada. Další možností je určitě urychlení zpracování. Nejlepší volbou by byla určitě paralelizace nejlépe na GPU. Sledování videa je sice sekvenční proces, ale jednotlivé metody pracující se statickým obrazem (tedy s velkým polem pixelů), by se určitě daly implementovat tak, aby využívaly co nejvíce vláken.

Literatura

[1] INRIA Person Dataset. http://pascal.inrialpes.fr/data/human/.

[2] Support vector machines (SVM) (Algoritmy podpůrných vektorů). http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf.

[3] Vše o světle – 13. Histogram. http://www.fotografovani.cz/art/fozak_df/rom_1_13_histogram.html.

[4] Acharya, T.; Ray, A.: *Image processing: principles and applications*. Wiley interscience, John Wiley, 2005, ISBN 9780471719984.

[5] Bertozzi, M.; Broggi, A.; Del Rose, M.; aj.: A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. 30 2007-oct. 3 2007, doi:10.1109/ITSC.2007.4357692.

[6] Bradski, D. G. R.; Kaehler, A.: *Learning opencv, 1st edition*. O'Reilly Media, Inc., 2008, ISBN 9780596516130.

[7] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. http://lear.inrialpes.fr/pubs/2005/DT05, June 2005.

[8] Dalal, N.; Triggs, B.; Schmid, C.: Human Detection Using Oriented Histograms of Flow and Appearance. http://dx.doi.org/10.1007/11744047_33, 2006, 10.1007/11744047_33.

[9] Hochman, Z.: Detekce pohybujících se objektů ve videosekvenci. 2010.

[10] MIT: MIT Pedestrian Data set. http://cbcl.mit.edu/cbcl/software-datasets/PedestrianData.html.

[11] Piccardi, M.: Background subtraction techniques: a review. 2004, doi:10.1109/ICSMC.2004.1400815.

[12] Pietikäinen, M.: Local Binary Patterns. http://www.scholarpedia.org/article/Local_Binary_Patterns, 2010.

[13] Wikipedia: HSV — Wikipedia, The Free Encyclopedia. http://cs.wikipedia.org/wiki/HSV, 2004, [Online; accessed 22-July-2004].

[14] Wikipedia: Normal distribution — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Normal_distribution&oldid=429917975, 2011, [Online; accessed 23-May-2011].

[15] Xiaoyu Wang, S. Y., Tony X. Han: An HOG-LBP Human Detector with Partial Occlusion Handling. 2009.

[16] Yilmaz, A.; Javed, O.; Shah, M.: Object tracking: A survey. http://doi.acm.org/10.1145/1177352.1177355, December 2006.

[17] Zeng, C.; Ma, H.; Ming, A.: Fast human detection using mi-sVM and a cascade of HOG-LBP features. 2010, doi:10.1109/ICIP.2010.5654100.

[18] Zhu, Q.; Yeh, M.-C.; Cheng, K.-T.; aj.: Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. 2006, doi:10.1109/CVPR.2006.119.

Příloha A

Obsah CD

A.1 Diplomová práce ve formátu PDF

A.2 Zdrojové kódy aplikace pro detekci a rozpoznání lidských postav ve video sekvenci

A.3 Návod pro přeložení aplikace na systému Linux

A.4 Sada originálních videí použitých pro testování aplikace

A.5 Sada otestovaných videí s vykreslenými výsledky