VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

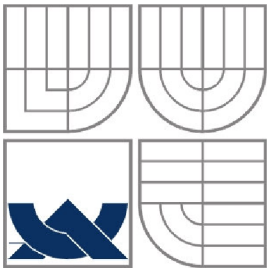
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SLEDOVÁNÍ POHYBU OSOB VE VIDEO SEKVENCI

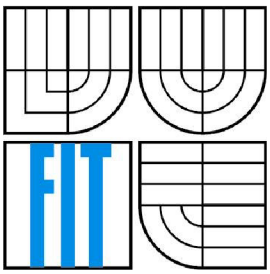
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ZDENĚK HOCHMAN
AUTHOR

BRNO 2008



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

SLEDOVÁNÍ POHYBU OSOB VE VIDEO SEKVENCI

MOVING PERSON SURVEILLANCE SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — ZDENĚK HOCHMAN
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠPANĚL
SUPERVISOR

BRNO 2008

Abstrakt

Práce se věnuje detekci pohybujících se osob ve video sekvenci. Zabývá se hlavně problematikou detekce pohybu a určením místa pohybu. V práci je představena poměrně nová metoda detekce pohybu Local Binary Pattern, která umožňuje rychlou a přesnou detekci pohybu. Součástí práce je také aplikace pro detekci osob. Práce podrobně zveřejňuje testy této aplikace a uvádí klady a zápory různých nastavení aplikace, které může posléze čtenář využít při práci s aplikací.

Klíčová slova

Detekce pohybujících se osob, Local Binary Patterns, detektor, histogram, C++, video sekvence, aktualizace pozadí.

Abstract

This thesis is devoted to detect moving person in video sequence. It is dealed with detect motion primarily and it is determinated by place of motion. The thesis introduces relatively new method of detection motion Local Binary Patterns, which allow quick and exact detect of motion. Inhere of this thesis is also application for detect person. The thesis publishes tests of this application detailed and it mentiones positives and negatives of different settings application which reader can finally use at work with application.

Keywords

Detect moving person, Local Binary Patterns, detector, histogram, C++, video sequence, background update.

Citace

Hochman Zdeněk: Sledování pohybu osob ve video sekvenci. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Sledování pohybu osob ve video sekvenci

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Španěla
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Zdeněk Hochman
23.3.2008

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Michalu Španělovi za cenné rady, doporučení a připomínky. Dále bych chtěl poděkovat svojí mamince za podporu, účinkování v některých testovacích video sekvencích a za korekturu mé práce. Děkuji také dalším členům své rodiny a svým přátelům za poskytnutou pomoc při tvorbě této práce.

Obsah

Úvod

Dříve než se člověk naučil mluvit a psát, pozoroval svoje okolí. V tomto okolí se kromě rostlin a zvířat nacházeli jiní lidé. Člověk dokáže naprosto intuitivně odlišit lidskou bytost od dalších objektů. Tato vlastnost mu byla dána v průběhu staletích evoluce, protože byla nezbytnou pro přežití. V dnešní době se dají počítače použít pro mnoho činností, od prohlížení fotografií až po řízení letů do vesmíru. Počítač umí částečně rozpoznat lidské písmo i hlas. Uměl by rozpoznat i osoby? Na tuto otázku by měla částečně odpovědět i tato práce.

Další otázka, která člověka možná napadá, je, jestli má rozpoznání pohybujících se osob nějaké praktické uplatnění. Určitě ano, například při ostraze objektu. Pokud počítač zjistí, že ve střežené zóně nastal pohyb, může rozhodnout, zda se jedná pouze o zatoulanou kočku nebo zloděje a případně spustit poplach. Další uplatnění může najít například v robotice. Robot zjistí, že se s ním snaží komunikovat člověk a pomocí dalších algoritmů může zjistit, o koho se jedná.

Téma bakalářské práce jsem si zvolil z několika důvodů: obor zabývající se zpracováním obrazu je velmi perspektivní, ve druhém ročníku mě zaujal předmět základy počítačové grafiky a rád programuji v jazyce C nebo C++.

V rámci této bakalářské práce se pokusím ve videu, snímaném v reálném čase, identifikovat pohyb. Dále se pokusím určit místo, kde k pohybu dochází. Tímto získám body, kde k pohybu dochází. Tyto body poté seskupím do jednotlivých objektů. Nakonec budu tyto výsledky v názorné formě prezentovat uživateli.

V první kapitole vysvětluji, co je to obraz, jeho zpracování a uložení v paměti počítače. Kapitola druhá se zabývá metodami pro detekci pohybu. Tyto metody zde podrobně rozebírám. V kapitole třetí popisuji, jak jsem postupoval při návrhu aplikace. Kapitola čtvrtá se věnuje implementaci. V páté kapitole prezentuji výsledky testování mojí aplikace. Shrnutím celé práce se zbývám v závěru.

Tato práce navazuje na semestrální projekt, při jehož tvorbě jsem získal znalosti o detekci pohybu. Doufám, že tato práce bude přínosem v oboru sledování pohybu osob ve video sekvenci.

1 Zpracování obrazu

Zpracování obrazu je velmi komplexní činnost, která se skládá z několika kroků.

Prvním krokem je samotné získání obrazu a jeho uložení v paměti počítače. K získání obrazu se nejčastěji používá videokamera nebo webkamera. Aby mohl být výstup kamery uložen v paměti počítače, je potřeba analogovou informaci o obraze převést na digitální formu. Proces převodu analogové informace na digitální se nazývá digitalizace.

Další krokem může být předzpracování obrazu. Při předzpracování obrazu je snaha o odstranění vad, které vznikly v důsledku nedokonalosti snímacího zařízení a při digitalizaci.

Třetím krokem bývá samotné zpracování. Podle toho, co je mým cílem, se snažím v obraze identifikovat pomocí algoritmů významné body nebo objekty. A dále s identifikovanými body pracovat. Pokud je cílem detekovat pohybující se objekty, budu se snažit najít objekty, které v různých snímcích obrazu změnily svoji polohu. Avšak pokud budu chtít zjišťovat velikost objektů, budu se snažit určit poměr velikosti zkoumaného objektu k velikosti objektu, jehož rozměry znám. Jak jsem v uvedených příkladech naznačil, samotné zpracování obrazu závisí na tom, co je mým cílem.

Posledním krokem je reprezentace výsledků. Reprezentace výsledků mi nejen slouží k ověření, že mnou prováděné zpracování poskytuje žádané výsledky, ale může dále sloužit jako vstup pro algoritmy, které se zabývají dalším zpracováním.

1.1 Reprezentace obrazu a digitalizace

Pokud se zabývám zpracováním obrazu, měl bych si nejdříve definovat samotný pojem obraz. Bohužel naprosto jednoznačná definice neexistuje. V mém případě budu pojem obraz chápat jako optický dvourozměrný obraz.

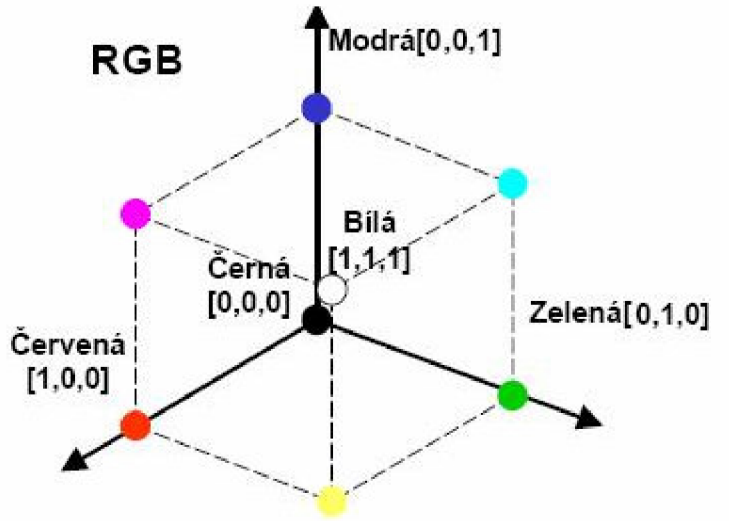
K matematické definici obrazu [1], se používá *obrazová funkce*, což je spojitá funkce dvou proměnných *f(x,y)*, kde x a y jsou reálná čísla, která představují souřadnice konkrétního bodu v obrazu. Hodnoty, kterých může nabývat x a y jsou omezeny rozměry obrazu (šířkou a výškou), z toho vyplývá, že definiční obor D(f) obrazové funkce je omezený. Obor hodnot H(f) obrazové funkce může být jedno nebo více čísel v závislosti na použitém barevném modelu.

Obrazová funkce může být spojitá nebo diskrétní. Spojitá obrazová funkce má definiční obor i obor funkčních hodnot spojitý. Diskrétní obrazová funkce má definiční obor i obor funkčních hodnot diskrétní. V reálném světě je obrazová funkce spojitá. Avšak protože pracuji na počítači, který je schopen ukládat pouze diskrétní hodnoty, je nutné převést spojitou obrazovou funkci na diskrétní. Tento převod se nazývá *digitalizace*. Diskrétní obrazová funkce se nazývá raster. Raster je matice

složená z bodů, které se nazývají pixely. Pixel poté nese hodnotu diskrétní obrazové funkce. Tato hodnota, jak již bylo řečeno, závisí na použitém barevném modelu.

Barevný model popisuje základní barvy a model míchání těchto základních barev do výsledné barvy. Barva je v přírodě dána směsí světla různých vlnových délek a různé barevné modely se snaží napodobit barvu co nejvěrněji. V praxi se používají modely, u kterých je zvolen vhodný kompromis mezi přesností podání barevného dojmu a složitostí konkrétního modelu [3]. Mezi nejčastěji používané barevné modely patří RGB, CMY, HSV, HLS, YUV.

Nejpoužívanějším barevným model je model RGB. Využívá aditivního míchání 3 barev – červené, zelené a modré. RGB model je reprezentován jednotkovou krychlí (Obrázek 1). Výsledná barva je dána tím, kolik každé složky obsahuje. Uplatnění složky v barvě je dáno číselným rozsahem od 0 do maximální hodnoty. Nula znamená, že složka se neuplatňuje, při maximu se uplatňuje celá intenzita složky.



(Obrázek 1) Reprezentace RGB modelu pomocí jednotkové krychle. Převzato z [5].

V RGB modelu je barva reprezentována pomocí tří barevných složek. Avšak v určitých případech postačí (někdy je dokonce vyžadováno), když bude barva reprezentována pouze jednou složkou. Tato složka se nazývá intenzita.

Převod z RGB modelu na stupně šedi (achromatický obraz) způsobuje ztrátu dat. Ztráta dat je však nahrazena menším paměťovým prostorem potřebným k uložení obrazu. Většinou se obraz převádí do 256 stupňů šedi. Z toho vyplývá, že pro uložení informace o jednom pixelu je potřeba 1 byte. Další zrychlení pak nastává při práci s obrazem, protože mám informaci o pixelu uloženu pouze v 1 bytu. Intenzitu vypočtu pomocí následujícího vzorce:

$$I = 0{,}299R + O{,}587G + 0{,}144B$$

Každá složka má při převodu na intenzitu jiný koeficient, jímž je zastoupena v intenzitě. Koeficienty jsou rozdílné z důvodu, jak lidské oko vnímá barvy. Lidské oko nejlépe vnímá zelenou

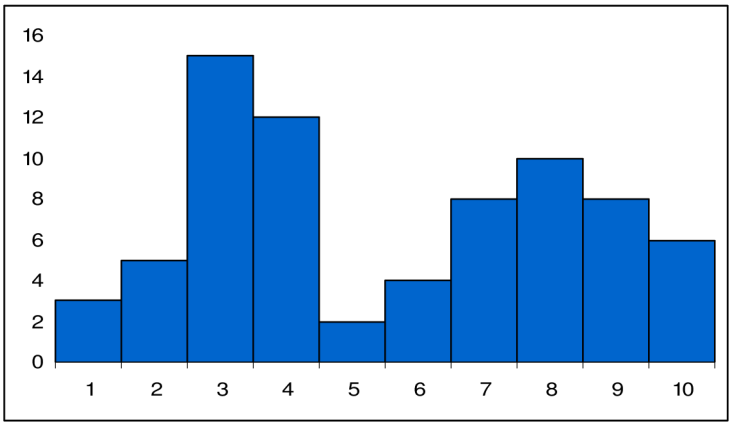
barvu a nejhůře modrou barvu. Pokud jsou koeficienty nastaveny na uvedené hodnoty, tak se lidskému oku jeví šedý obraz jako původní (Obrázek 2).

(Obrázek 2) Převod RGB obrazu do 256 stupňů šedi

1.2 Histogram

Při detekci pohybu v obrazu často využívám histogramů. Tudíž zde vysvětlím, co histogram je a jaké metody porovnávání histogramů existují.

Histogram je využíván ve statistice, k reprezentaci četnosti výskytu hodnot v tabulce. Na ose x jsou jednotlivé třídy histogramu. Každá třída je specifikována na určitém intervalu, který se nepřekrývá s intervaly ostatních tříd. Velikost intervalu bývá obvykle pro všechny třídy stejná. Osa y udává počet výsledků, které patří do příslušné třídy. Na obrázku (Obrázek 3), je znázorněn histogram, který má 10 tříd, každá třída má velikost intervalu 1 a například do třídy 4, patří 12 výsledků.



(Obrázek 3) Ukázka histogramu.

Dalším problémem, kterým jsem se byl nucen zabývat, bylo zjistit, zda jsou si dva histogramy podobné nebo zda jsou zcela odlišné. K zjištění podobnosti existuje celé řada metod. Některé z nich zde uvedu.

Průnik histogramů (histogram intersection):

$$H(x_1, x_2) = \sum_i \min(x_{1,i}, x_{2,i})$$

Porovnání se provádí na normalizovaných histogramech x_1 a x_2. Metoda prochází všechny třídy histogramů a zjišťuje, v kterém ze dvou histogramů má aktuální třída menší hodnotu. Tuto hodnotu pak započte do výsledku.

Nejmenší čtverce (Chi-square):

$$H(x_1, x_2) = \sum_i \frac{(x_{1,i} - x_{2,i})^2}{x_{1,i} + x_{2,i}}$$

Metoda nejmenších čtverců je daleko přesnější při porovnávání histogramů než metoda průnik histogramů, avšak je časově náročnější. Tuto metodu jsem implementoval a používám ji pro porovnávání histogramů.

Logaritmická pravděpodobnost (Log-likelihood):

$$H(x_1, x_2) = 2\sum_i \left[x_1 \ln x_1 - x_1 \ln x_2\right]$$

Další funkce, kterou jsem zahrnul do své implementace. Je časově náročnější než předešlé, protože je nutné vyčíslit logaritmus. Tato metoda pracuje s normalizovanými histogramy.

Kromě těchto vyjmenovaných funkcí se dá nalézt ještě další velké množství funkcí určených pro statistické testy. Jednou z nich je například korelace (Correlation).

2 Detekce pohybujících se objektů

Abych detekoval pohyblivý objekt, musím nejdříve zjistit, jestli se v obraze něco pohybuje. Poté musím určit místo, kde k pohybu dochází. Nakonec se pokusím body, které jsem označil za pohybující se, seskupit do objektů.

2.1 Metody detekce pohybu

Detekce pohybu hraje v mnoha aplikacích klíčovou roli. Uplatňuje se často v průmyslových aplikacích, například ve sledování dopravy, ostrahy objektů nebo při detekci nebezpečných předmětů v kolejišti. Detekce pohybu nachází uplatnění i ve vojenských aplikacích. Například při automatickém zaměřování. Protože je detekce pohybu velmi využívaná, snaží se vědci o nalezení nových rychlejších a přesnějších algoritmů pro detekci nebo o vylepšení stávajících.

Metody detekce pohybu, kterým se zde budu věnovat jsou:

- Porovnání jasových histogramů
- Rozdílné body mezi snímky
- Local Binary Patterns
- Optický tok

Při detekci pohybu často využívám porovnávání snímků. Jeden snímek beru jako referenční, označím si ho jako pozadí. Obdržím další snímek, který si označím jako popředí. Porovnám popředí s pozadím. Pokud k žádné změně nedošlo, není v popředí pohyb. Pokud nastaly v popředí změny, detekoval jsem pohyb. Pro člověka není takovéto porovnání dvou snímků žádným problémem. Lidské vnímání je uzpůsobeno tak, že se zaměřuje pouze na podstatné změny. Nezkoumá každý bod obrazu. Na rozdíl od lidského vidění počítač zkoumá každý bod snímku. Takže v oblastech dvou snímků, které by člověk označil za stejné, počítač najde změny (Obrázek 4). Tyto změny jsou zapříčiněny několika různými aspekty. Například šumem, který je způsoben nedokonalostí kamery, změnou osvětlení, drobným pohybem, který nepovažujeme za relevantní (pohyb listů, trávy).

Po detektorech pohybu je vyžadováno, aby detekovaly pouze významné změny. Této vlastnosti lze docílit volbou vhodného detektoru, nastavením určité tolerance (prahu) a průběžnou aktualizací pozadí. Volbou vhodného detektoru se zabývám v této kapitole. Nastavením tolerance se zabývám v kapitole 5.3. A průběžnou aktualizací pozadí se zabývám v kapitole 2.2.

(Obrázek 4) Porovnání jasu pixelů dvou snímků

2.1.1 Porovnání jasových histogramů

V jasovém histogramu představují třídy histogramu intenzitu pixelu. Pokud pracuji se snímkem ve stupních šedi, má jasový histogram 256 tříd. Jasový histogram vytvořím tak, že zjistím intenzitu pixelu a zvýším o 1 počet výskytů v třídě, která odpovídá intenzitě příslušného pixelu.

Metoda je založena na porovnání histogramu pozadí s histogramem aktuálního snímku. Pokud jsou histogramy odlišné (odlišné o větší hodnotu než je nastaven práh), můžu konstatovat, že ve scéně došlo k pohybu (Obrázek 5). Bohužel nevím, kde přesně ve scéně došlo k pohybu, protože porovnáním histogramů pouze zjistím, že došlo ke změně jasové charakteristiky snímku. Zjištěním oblasti pohybu se zabývám v kapitole 3.2.

(Obrázek 5) Porovnání jasových histogramů. Červeně vyznačeny významné odchylky.

Jedná se o jednoduchou a rychlou metodu. Je náchylná ke změně světelných podmínek (změní se jas pixelů a tím i celý histogram). Nedokáže určit, kde ve snímku ke změně došlo. Úspěšnost detekce také závisí na způsobu, jakým aktualizuji pozadí.

Tuto metodu jsem implementoval ve své aplikaci. Avšak není stěžejní metodou pro detekci pohybu. Její využití spočívá v porovnání s metodou Local Binary Patterns (kapitola 2.1.3).

2.1.2 Rozdílné body mezi snímky

Metoda pracuje tak, že porovná každý pixel pozadí s korespondujícím pixelem aktuálního snímku. Pokud od sebe pixely odečtu a ukládám je do nového snímku, získám tzv. rozdílový snímek. Pixely, které jsou shodné, mají v rozdílovém snímku hodnotu 0 (černá barva). Naopak pixely, které by byly naprosto odlišné, by měly hodnotu 255 (bílá barva). Z rozdílového snímku pak můžu určit, v kterých oblastech je snímek neměnný a kde naopak došlo ke změně.

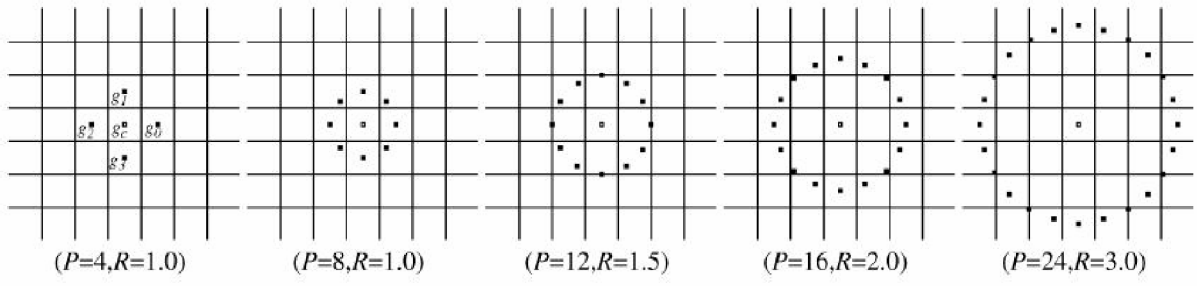
Metoda může pracovat jak s RGB snímkem, tak se snímkem ve stupních šedi. U stupňů šedi sice ztratím barevnou informaci, ale na oplátku dosáhnu zrychlení algoritmu.

Metoda patří mezi výpočetně nenáročné metody, ale dokáže určit, kde ve snímku došlo k pohybu. Na toto místo by se posléze dal použít náročnější algoritmus pro detekci typu pohybujícího se objektu. Metoda je stejně jako metoda jasových histogramů náchylná na změnu světelných podmínek.

2.1.3 Local Binary Patterns

Local Binary Patterns (dále jen LBP) je medoda původně vyvinutá pro klasifikaci textur. Byla vyvinuta na universitě Oulu, která se nachází ve Finsku. Kromě klasifikace textur našla uplatnění i v segmentaci obrazu, detekci obličeje a detekci pohybu. Tuto metodu jsem si zvolil jako hlavní metodu pro detekci pohybu ve své aplikaci. Proto v této kapitole kromě obecného popisu metody uvedu, jak je tato metoda implementována v mé aplikaci.

Metoda pracuje s obrazem, převedeným do stupňů šedi. LBP se nezabývá pouze hodnotou pixelu, ale při svém výpočtu uvažuje i hodnoty okolních pixelů. Protože se LBP operátor zabývá nejen zkoumaným pixelem, ale i jeho okolím, je nutno určit, v jakém okolí bude pracovat. LBP operátor $LBP_{P,R}^{riu2}$ je určen počtem zkoumaných sousedních pixelu P a vzdáleností těchto pixelů od zkoumaného pixelu R (tzv. radius). Na obrázku (Obrázek 6) je znázorněno, jakými sousedními pixely se bude zabývat LBP operátor pro různá nastavení parametrů P a R.



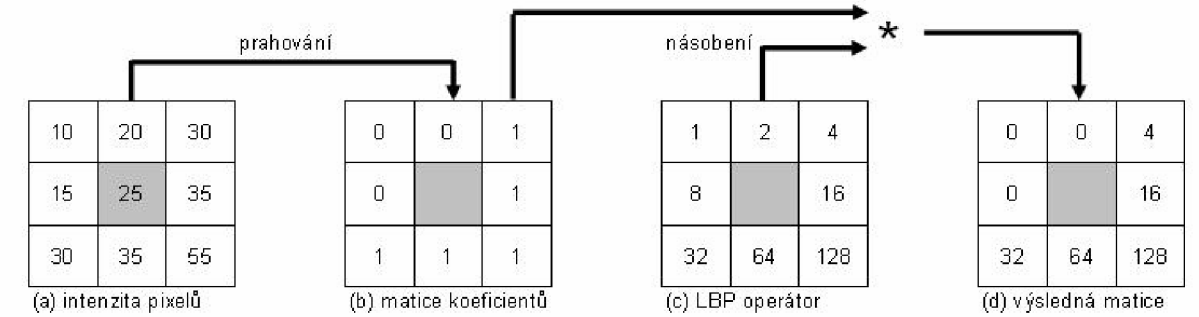
(Obrázek 6) Převzato z [6]. Ukazuje různá nastavení parametrů P a R LBP operátoru.

Po určení sousedních pixelů, s kterými bude LBP operátor pracovat, je dále nutné určit váhu sousedních pixelů, s jakou se budou započítávat do výsledné LBP hodnoty. Ve své aplikaci používám LPB operátor P=8, R=1.0. Operátor s příslušnými váhami je uveden na obrázku (Obrázek 7c)

Výpočet LBP hodnoty pixelu probíhá následovně: Nejdříve je nutné barevný obrázek převést do stupňů šedi, abych mohl pracovat s intenzitou jednotlivých pixelů. Poté pomocí prahování hodnoty středového pixelu získám matici koeficientů (Obrázek 7b). Prahování provádím podle vzorce:

$$f(x) = \begin{cases} 0 & x > y \\ 1 & x \leq y \end{cases}$$

Kde x je hodnota středového pixelu a y je hodnota sousedního pixelu. Poté vynásobím matici koeficientů (Obrázek 7b) s LBP operátorem (obrázek 7c) (pozn. Nejedná se o násobení matic. Pouze mezi sebou násobím prvky matic se stejnými indexy). Vynásobením dostanu výslednou matici (Obrázek 7d). LBP hodnotu pro středový pixel získám součtem prvků výsledné matice. LBP hodnota středového pixelu v příkladu by byla 4+16+32+64+128=244.



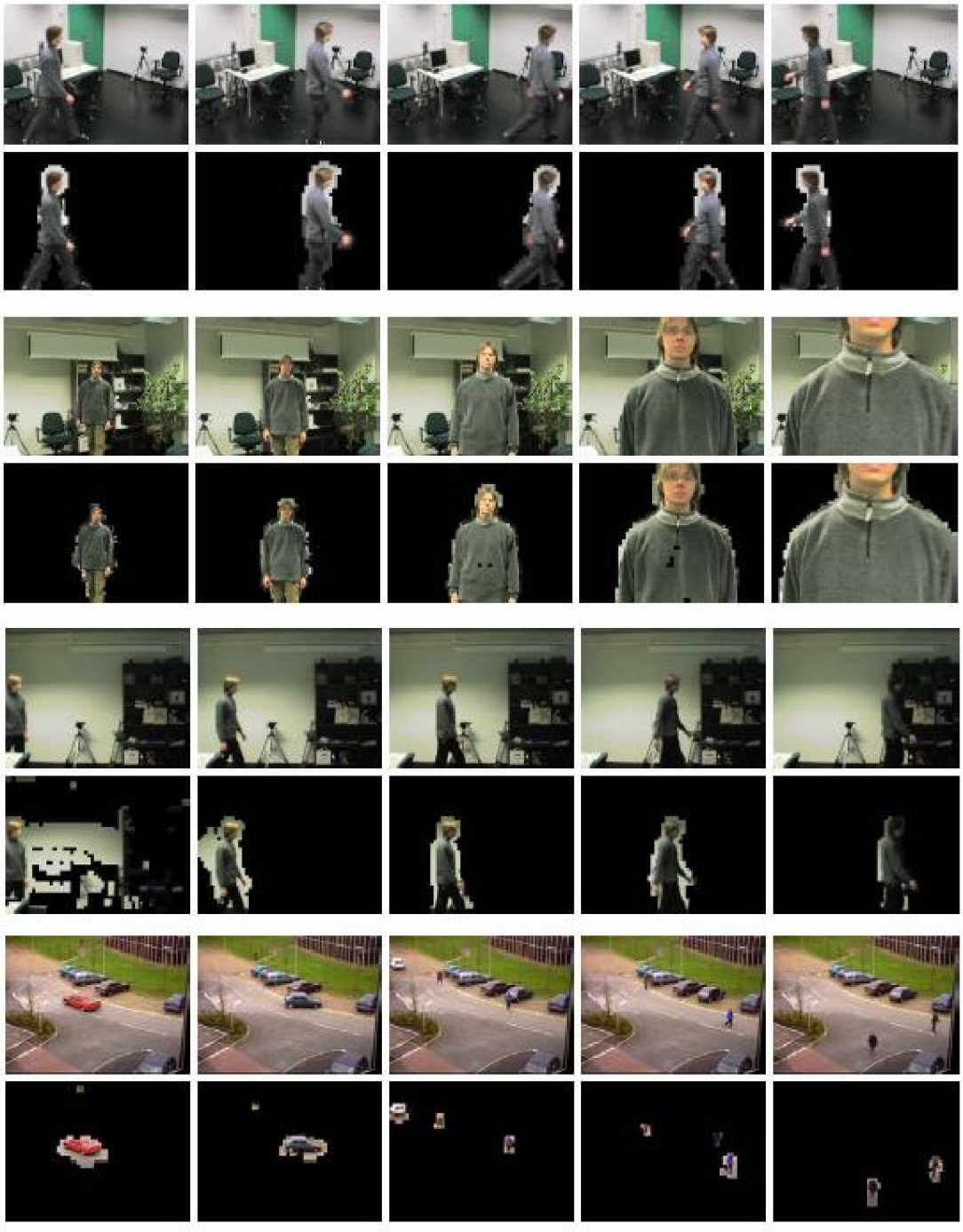
(Obrázek 7) Výpočet LBP

Ještě musím vyřešit, jak budu postupovat, pokud je středovým pixelem krajní bod obrazu (nemá úplné okolí). Buď nebudu LBP hodnotu pro krajní body počítat, nebo neexistující body v matici koeficientů nahradím nulou.

Třídy LBP histogramu odpovídají hodnotám, kterých může nabývat LBP hodnota pixelu. V mém případě má LBP histogram 256 tříd. Rozsah tříd je 0 až 255. Po vypočtení LBP hodnoty pixelu, navýším počet výskytů v příslušné třídě LBP histogramu o 1.

Detekci pohybu pak provádím na základě porovnání LBP histogramu pozadí s LBP histogramem aktuálního snímku. Pokud jsou histogramy podobné, mohu konstatovat, že nedochází k pohybu. Naopak pokud je rozdíl histogramů dostatečný (větší než práh), mohu konstatovat, že došlo k pohybu (Obrázek 8).

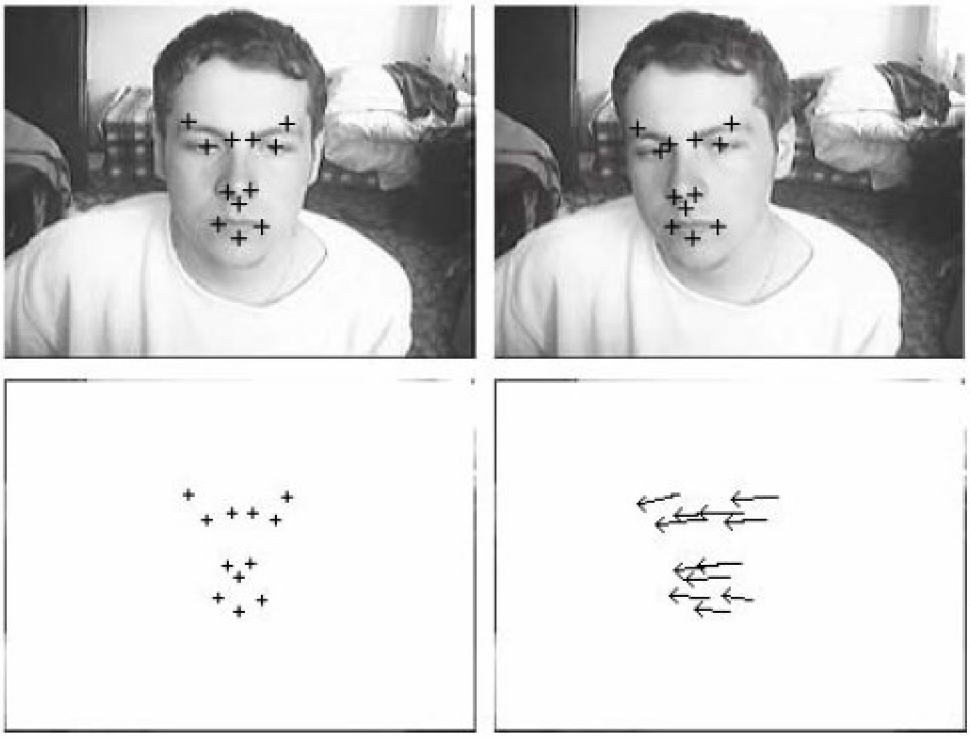
Metoda se dá zařadit mezi rychlé metody. Výsledek není ovlivňován změnou světelných podmínek. Další výhodou je, že drobné pohyby (jako pohyb listů nebo trávy) neovlivňují výsledek. LBP jsou invariantní vůči rotaci. Této vlastnosti se využívá při klasifikaci textur. Jedinou podstatnou nevýhodou při realizaci detektoru pohybu je, že LBP nejsou schopny určit místo pohybu v obraze. Odstraněním této vady se zabývám v kapitole 3.2.



(Obrázek 8) Výsledky detekce pohybu metodou LBP. Převzato z [6].

2.1.4 Optický tok

Zachycuji změny obrazu v čase. Každý bod obrazu má přiřazen vektor, který obsahuje směr a rychlost pohybu v daném bodě. Jsem schopen popsat změny obrazu uvnitř regionu, který je dán parametrickými funkcemi souřadnic (Obrázek 9).



(Obrázek 9) Detekce pohybu pomocí optického toku. Převzato z [2].

Metoda dokáže určit místo, kde k pohybu došlo. Ale výpočetně je velmi náročná. Pokud by v obraze nedocházelo k pohybu, zbytečně bych plýtval výkonem. Tento algoritmus je vhodnější použít jako dodatečnou detekci, například k predikci pohybu. Další nevýhodou je, že potřebuje konstantní světelné podmínky.

Detailní popis a použití této metody lze nalézt v [2].

2.2 Aktualizace pozadí

Mezi velmi důležité, ale často opomíjené činnosti patří aktualizace pozadí. I když zvolím sebelepší detektor, bez správné aktualizace pozadí nebude poskytovat tak dobré výsledky, jak by mohl. Aktualizace pozadí je v podstatě započtení aktuálního snímku do pozadí. Tudíž pokud do záběru vstoupí objekt a zůstane na stejném místě určitou dobu (určitý počet snímků), nebude považován již za pohybující se objekt, ale za pozadí. Volba správné aktualizace se také odvíjí od účelu aplikace. Pokud budu sledovat parkoviště, tak auto, které zaparkuje na parkovacím místě, mohu do 2 minut přestat považovat za pohyblivý objekt, ale již ho považovat za pozadí. Avšak pokud se budu snažit detekovat překážky v kolejišti, byla by chyba, aby předmět, který představuje ohrožení, se stal pozadím.

Aktualizace pozadí se dá dále využít při změně osvětlení. Detektor může hlásit chvíli pohyb, ale po určité době se sám z nastalými podmínkami vyrovná a již pracuje správně. Podobný problém jako změna osvětlení můžu způsobovat i posunutím kamery.

Jak jsem naznačil, je aktualizace pozadí velmi rozsáhlým oborem, tudíž zde uvedu jenom několik metod aktualizace pozadí.

Průměr předchozích snímků je velmi jednoduchá metoda. Vezmu několik předchozích snímků a pro každý pixel vypočítám průměrnou hodnotu. Průměrnou hodnotu poté považuji za pozadí.

$$x = \frac{\sum_i y_i}{i}$$

X je průměrná hodnota pixelu, i počet předchozích snímků a y hodnota pixelu předchozího snímku.

Váha předchozích snímků je podobná předchozí metodě. Jediný rozdíl je, že předchozí snímky se do výsledku nezapočítávají se stejnou vahou.

Započtení aktuálního snímku vahou do pozadí. Tato metoda započte hodnoty, které obsahuje aktuální snímek x_a s určitou vahou α do pozadí x_b.

$$x_{b+1} = \alpha x_a + (1-\alpha) x_b$$

Tyto vyjmenované metody se kromě aktualizace pixelů pozadí dají využít i k aktualizaci histogramů. Tudíž nepotřebuji v paměti uchovávat snímek pozadí, ale stačí mně již vypočtený LBP histogram, který aktualizuji.

Ve své aplikaci využívám *započtení histogramu do pozadí s určitou vahou*. Váhu označuji jako učicí koeficient. Čím vyšší hodnotu má učicí faktor, tím rychleji se objekt stane součástí pozadí. Algoritmus postupně prochází všechny třídy histogramu (i).

$$x_{b+1}[i] = \alpha x_a[i] + (1-\alpha) x_b[i]$$

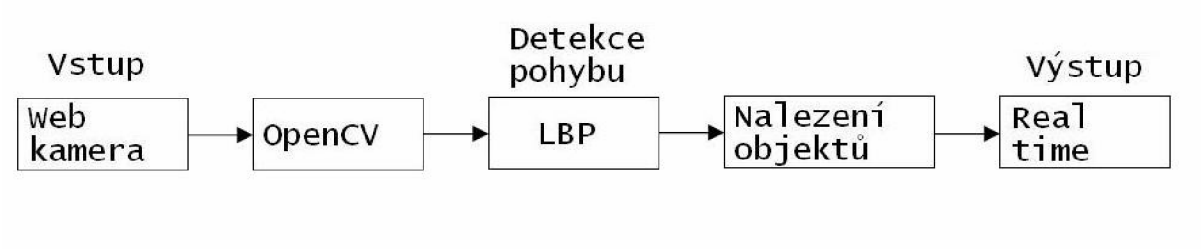
Další metoda pro aktualizaci pozadí je uvedena v [7]. Tato metoda využívá několika histogramů pozadí a pro aktualizaci a porovnání s popředním volí nejvhodnější ze sady uložených histogramů.

3 Návrh aplikace

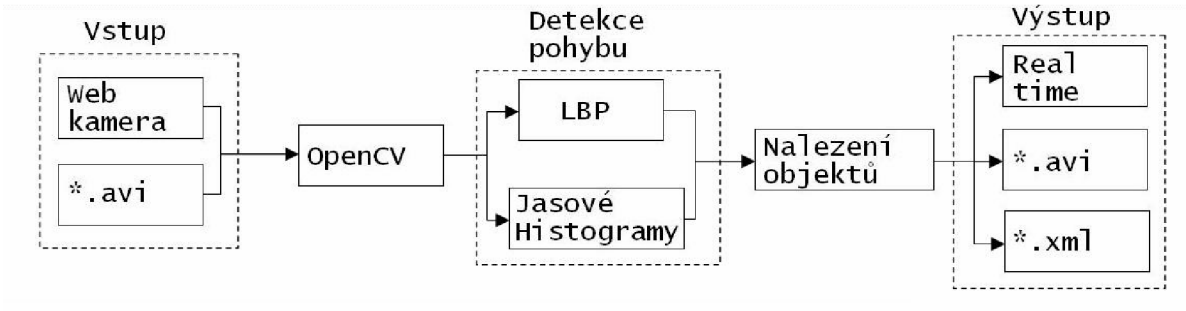
Většina aplikací se zabývá buď sledováním pohybu, nebo detekcí objektů v obraze. Mým cílem je tyto dvě akce vzájemně provázat.

3.1 Struktura aplikace

Prvotní návrh aplikace byl velmi jednoduchý (Obrázek 10), avšak v průběhu vývoje, jsem původní návrh dále rozšiřoval (Obrázek 11).



(Obrázek 10) Prvotní návrh aplikace.



(Obrázek 11) Konečná podoba aplikace.

Prvním krokem byla volba vstupu. Mým cílem je detekovat pohybující se objekty v reálném čase. Proto jsem zvolil jako vstup webkameru. V dnešní době poskytují webkamery obraz s dostatečným rozlišením, jsou cenově dostupné a někdy bývají i součástí notebooků. Avšak v průběhu dalšího vývoje jsem potřeboval, srovnávat různá nastavení detektorů pohybu. Bohužel vstup v reálném čase neposkytuje relevantní srovnání. Proto jsem do aplikace zavedl, kromě vstupu z webkamery i vstup z video souboru. Aplikace podporuje videa ve formátu avi.

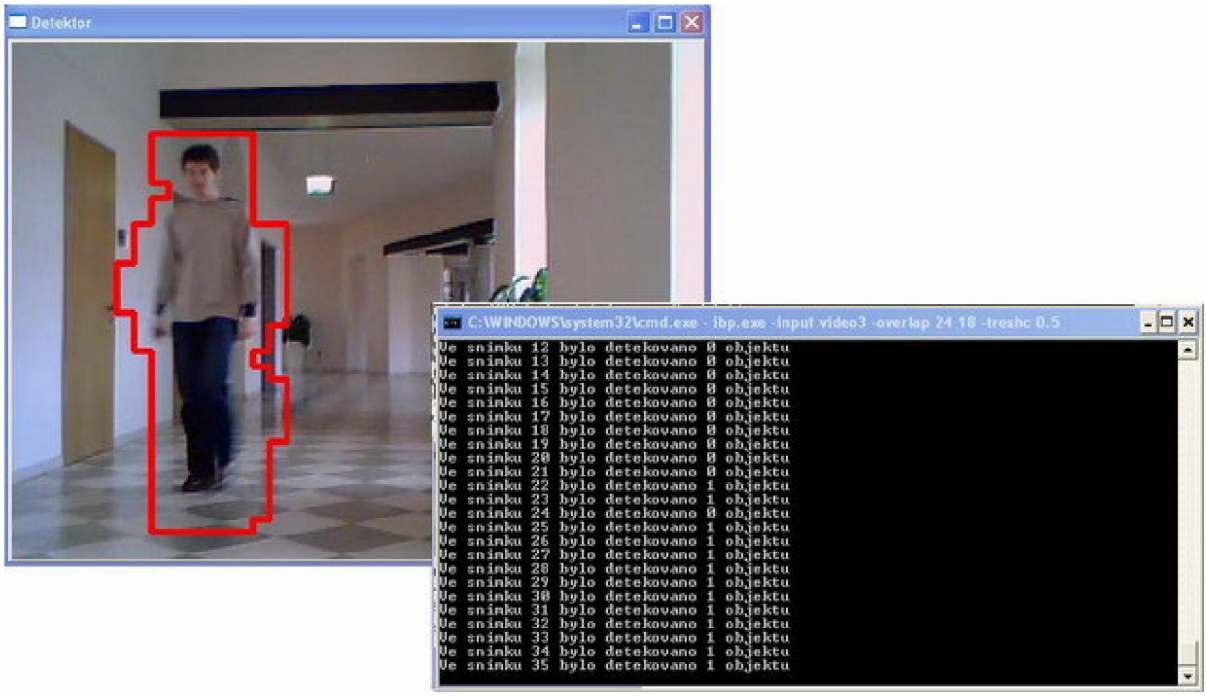
Surový vstup, je potřeba převést do reprezentace, která se dá uložit v počítači a dále s ní jednoduše pracovat. K převodu jsem využil knihovny OpenCV. Jedná se o open source knihovnu, vyvíjenou společností Intel. Umožňuje jednoduše získávat z webkamery i video souboru snímky, které uloží do objektu, který je uzpůsoben pro práci se snímky a obrázky. Tento objekt pak slouží jako vstup pro detekci pohybu.

Většina detektorů pohybu využívá metod srovnávání jasových histogramů nebo rozdílných bodů mezi snímky. Protože se snažím nejen o vytvoření kvalitní aplikace, ale i rozšíření znalostí

v oboru počítačového vidění, rozhodl jsem se pro detekci pohybujících objektů metodu LBP. V konečné podobě jsem ještě implementoval metodu srovnávání jasových histogramů, abych mohl porovnat výsledky poskytované LBP. Kromě samotné detekce pohybu je detektor upraven způsobem uvedeným v kapitole 3.2 tak, aby byl schopen určit i místa, kde k pohybu dochází. Jak je detektor pohybu implementován, popisuji v kapitole 4.2.

Další část aplikace představuje nalezení jednotlivých objektů ve snímku. K nalezení objektů využívám metody detekce kontur (popsáno v kapitole 3.3).

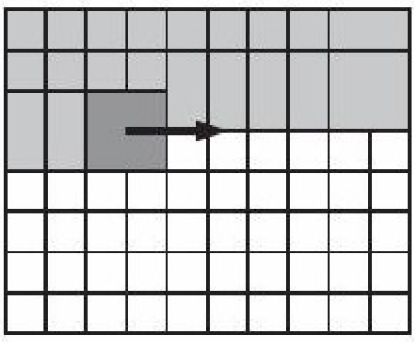
Poslední částí aplikace je reprezentace výsledků. Nejlepší reprezentací pro člověka je obraz, ve kterém jsou pohybující se objekty vyznačeny. Aplikace vždy zobrazuje zpracované snímky ve svém výstupním okně a informace o běhu aplikace do konzole (Obrázek 12). Další možností je ukládat výsledné snímky s vyznačenými objekty do video souboru formátu avi. Poslední možností je zaznamenávat výstup do xml souboru.



(Obrázek 12) Výstup aplikace.

3.2 Určení místa pohybu v obraze

K určení místa pohybu používám velmi jednoduchou metodu. Obraz rozdělím do menších oblastí, které označuji jako bloky. Každý blok má svůj vlastní LBP histogram pozadí. A při detekci pohybu vytvářím histogram pro každý blok. Abych dosáhl přesnější detekce pohybu, využívám částečného překrytu bloků (Obrázek 13).



(Obrázek 13) Částečně překrývající se bloky. Převzato z [7].

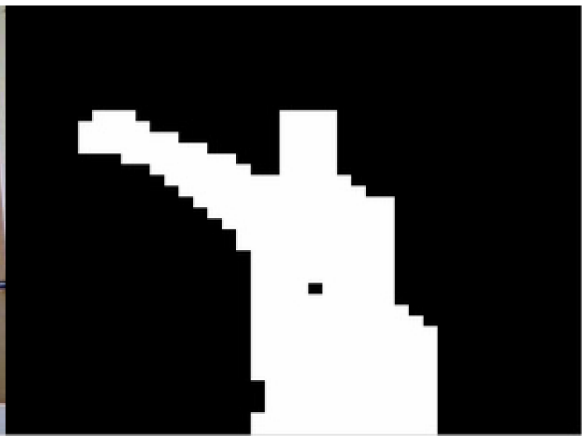
Velikost bloků a velikost jejich vzájemného překrytí, kromě přesnosti detekce místa pohybu, také významně ovlivňuje dobu vyhodnocení snímku. O těchto závislostech se lze více dozvědět v kapitole věnované testování (kapitola 5.1 a 5.2).

3.3 Detekce objektů

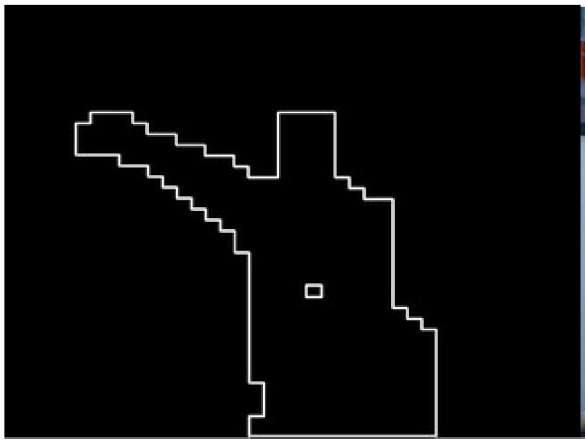
Většina mnou nalezených metod pro detekci objektů pracuje přímo s obrazem. Ve své aplikaci jsem se pro detekci jednotlivých pohybujících se objektů rozhodl použít metodu založenou na detekci kontur (hranic objektu).

Metoda detekce kontur v mém případě pracuje s monochromatickým (černobílým) obrazem. Nejdříve vytvořím nový monochromatický obraz, ve kterém jsou pixely náležející blokům, ve kterých byl detekován pohyb, označeny bílou barvou (Obrázek 15). Pixely náležející blokům, kde nebyl detekován pohyb zůstávají černé (Obrázek 15). Monochromatický obraz slouží jako vstup funkce pro detekci kontur (funkce je vestavěna v OpenCV). Funkce rozliší jednotlivé objekty a uloží jejich hranice do struktury (Obrázek 16). Nakonec vykreslím hranice objektu do výstupního snímku (Obrázek 17).

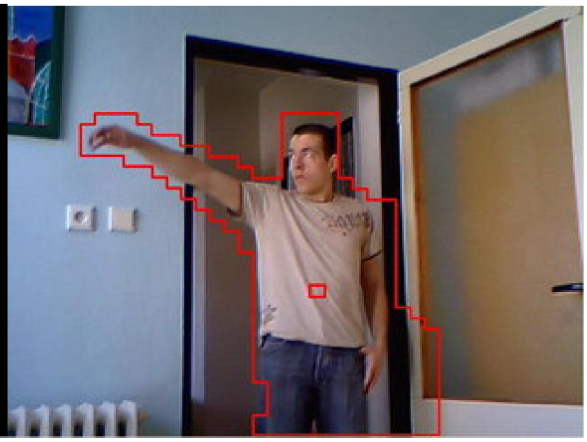
(Obrázek 14) Vstup do z webkamery



(Obrázek 15) Monochromatický obraz



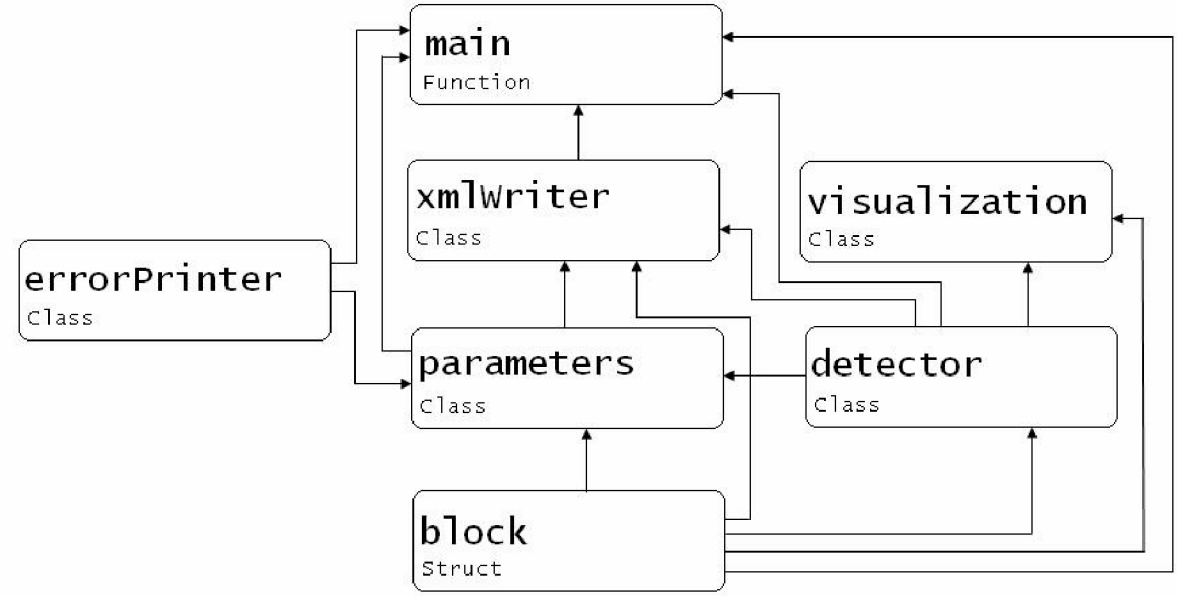
(Obrázek 16) Kontury objektu



(Obrázek 17) Výstupní snímek

3.4 Návrh tříd aplikace

Aplikaci jsem rozdělil do několika tříd (Obrázek 18). Snažil jsem se, aby každá třída zapouzdřovala jednu část aplikace. V části věnované implementaci (Kapitola 4) popisuji významné rysy jednotlivých tříd. Všechny metody, jejich vstupní a výstupní parametry jsou poté popsány v programové dokumentaci, která je součástí elektronické přílohy. Detaily implementace lze poté najít ve zdrojových textech, které jsou také součástí elektronické přílohy.



(Obrázek 18) Třídy a jejich vztahy (šipky reprezentují, jak se třídy vzájemně využívají, nikoliv dědičnost)

4 Implementace

Testovací aplikace je implementována v jazyce C++. Jako vývojové prostředí jsem použil Microsoft Visual Studio 2005. Dále používám funkce z knihovny OpenCV [9] pro převod zpracování obrazu a detekce kontur. Pro vytváření výstupního xml souboru používám funkce knihovny TinyXML [10].V následujících kapitolách popíši jednotlivé třídy, jejich podstatné metody a vzájemné vztahy tříd (Obrázek 18). Nebudu se zabývat detaily implementace. Ovládání aplikace se provádí pomocí příkazového řádku. Ovládání aplikace popisuji v příloze.

4.1 Struktura block

Tato představuje blok snímku. Strukturu využívám jak k uchování informací o pozadí, tak k uložení informací o popředí. Poté vzájemným srovnáním detekuji pohyb. Struktura uchovává souřadnici bloku ve snímku. Z těchto souřadnic se dá odvodit i velikost bloku. Dále uchovává číslo bloku. Číslo bloku využívám při zápisu do xml souboru a při ladění aplikace. Nejpodstatnější informací, kterou uchovává, je histogram daného bloku (podle metody, jaká se používá pro detekci se jedná buď o jasový histogram, nebo o LBP histogram). Histogram jsem implementoval jako pole hodnot typu int.

4.2 Třída detector

Tato třída slouží k detekci pohybu a určení místa, kde k pohybu dochází. Nejdůležitějšími atributy jsou seznamy bloků pozadí a popředí. Seznamy obsahují všechny bloky daného snímku a jsou implementovány pomocí šablony list z knihovny STL. Další atributy této třídy uchovávají rozměry snímku, rozměry bloků, vzájemné překrytí bloků, učicí faktor, metodu pro porovnání histogramů a její práh.

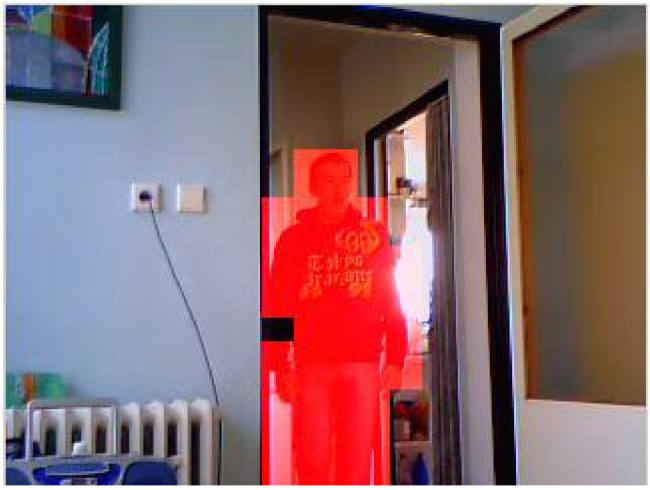
V této třídě jsou implementovány detektory pohybu na základě LBP a jasových histogramů. Práce s třídou je pro obě metody stejná, liší se pouze názvy volaných metod. Nejdříve je potřeba nastavit detektor (velikost bloků, porovnávací metodu, práh atd.). Poté co jsem nastavil detektor, je nutné vytvořit seznam s referenčními bloky. Metoda pro vytvoření seznamu referenčních bloků vypočítá pro každý blok histogram a uloží jej do struktury. Na vstup metody se zadá ukazatel na snímek, který se má považovat za referenční (LBPSetReferer pro LBP nebo brightnessSetReferer pro jasové histogramy).

Když mám nastaven referenční snímek, mohu již detekovat pohyb. Prvním krokem k detekci je naplnit seznam popředí. Seznam opět naplním voláním metody, jejímž vstupním parametrem je

ukazatel na snímek (LBPCountFrame pro LBP nebo brightnessCountFrame pro jasové histogramy). Metoda vypočte histogramy pro všechny bloky popředí. Zároveň aktualizuje pozadí započtením histogramů popředí do histogramů pozadí. Míra, jakou se uplatní histogramy popředí, určuje učicí algoritmus (více v kapitole 2.2). Druhým krokem v detekci pohybu je zavolání metody (returnMotionBlock), která porovnává histogramy jednotlivých bloků pozadí a popředí (vypočte rozdíl histogramů). Metoda vrací seznam bloků, jejichž rozdíl histogramů je větší než nastavený práh porovnávací metody.

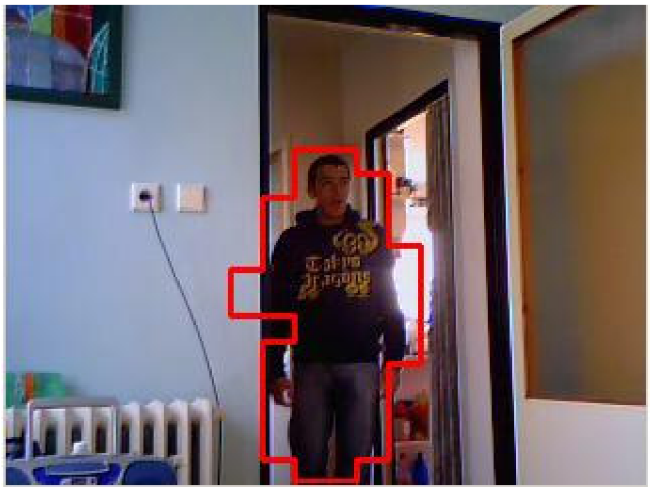
4.3 Třída visualization

Třídu visualization používám k detekci objektů a zobrazení výsledků. Třída má pouze dvě metody. První metoda ve výstupním snímku označí bloky (každý pixel), v kterých byl detekován pohyb. Označení provádí tak, že hodnotu červené barevné složky nastaví na maximální hodnotu (maximální intenzita). Vstupem metody je seznam s bloky, v nichž byl detekován pohyb. Výstup je ukázán na obrázku (Obrázek 19).



(Obrázek 19) Zobrazení bloků s detekovaným pohybem

Druhá metoda detekuje jednotlivé objekty na základě jejich kontur. Vstupem metody je seznam s bloky, v nichž byl detekován pohyb. Poté vytvoří monochromatický snímek, v kterém nalezne kontury a ty vykreslí do výsledného snímku (detailněji popsáno v kapitole 3.3). Kromě vykreslení kontur do výsledného snímku vrací i počet nalezených objektů. Výstup na obrázku (Obrázek 20).



(Obrázek 20) Zobrazení detekovaných objektů

4.4 Ostatní třídy

V této kapitole popíši třídy, které se okrajově týkají samotné detekce pohybujících se osob. Avšak pro funkčnost celé aplikace, případně některých jejich rysů jsou podstatné.

Třída parameters: Tuto třídu využívám pro zpracování, kontrolu, uložení parametrů, s jakými byla aplikace spuštěna, a nastavení objektu třídy detector.

Třída errorPrinter: Objekt této třídy využívám k tisku chybových hlášení a nápovědy.

Třída xmlWriter: Tato třída slouží pro zápis výsledků do xml souboru. Základním výstupem je obraz, v kterém člověk nejlépe rozpozná výsledky. Avšak pro další počítačové zpracování je tento typ výstupu nevhodný. Proto jsem zvolil jako další možný typ výstupu xml soubor. K vytváření xml souboru používám TinyXML.

Funkce main: Funkce kromě postupného volání metod objektů v sobě implementuje tzv. zatmění. Pokud procento bloků, v kterých byl detekován pohyb, překročí zadanou mez, aplikace prohlásí, že se nejedná o pohyb. Tudíž se pohyb nezobrazuje. Zatmění je volitelnou vlastností, která se dá nastavit v parametrech, s kterými je aplikace spuštěna. Tato vlastnost je použitelná například tehdy, pokud je kamera zakryta nebo v důsledku nárazu do ní se změní snímaný prostor.

5 Testování

Tato kapitola se zabývá, jak různá nastavení aplikace ovlivňují kvalitu detekce osob ve video sekvenci. Testovací počítač měl následující parametry: CPU – AMD Turion 64 1,58 GHz, paměť – 512MB DDR.

5.1 Závislost kvality detekce osob na velikosti bloku

Jako první jsem testoval velikost bloku, abych zjistil, jaká velikost bloku je nejvhodnější pro detekci osob. K testům jsem použil video v rozlišení 320x240 a 640x480. Překryt bloků jsem nastavil na polovinu velikosti bloku. Jako porovnávací metodu histogramů jsem zvolil metodu nejmenších čtverců.

Nejdříve jsem u videa s rozlišením 320x240 nastavil velikost bloku na 16x12. U této velikosti bloku bylo velmi obtížné zvolit vhodný práh při porovnávání histogramů. U nízkých hodnot prahu bylo detekováno velmi mnoho neexistujícího pohybu (Obrázek 21). Důvodem tohoto chování je malá přesnost informací v histogramu. Do histogramu se započítává velmi málo hodnot (pouze 192). Malý počet hodnot v histogramu způsobuje velkou citlivost detektoru na drobné změny, které nechci považovat za pohyb. Pokud jsem hodnotu prahu navýšil, detektor špatně zjišťoval hranice objektu (osoby) (Obrázek 22).

(Obrázek 21) Detekce neexistujícího pohybu

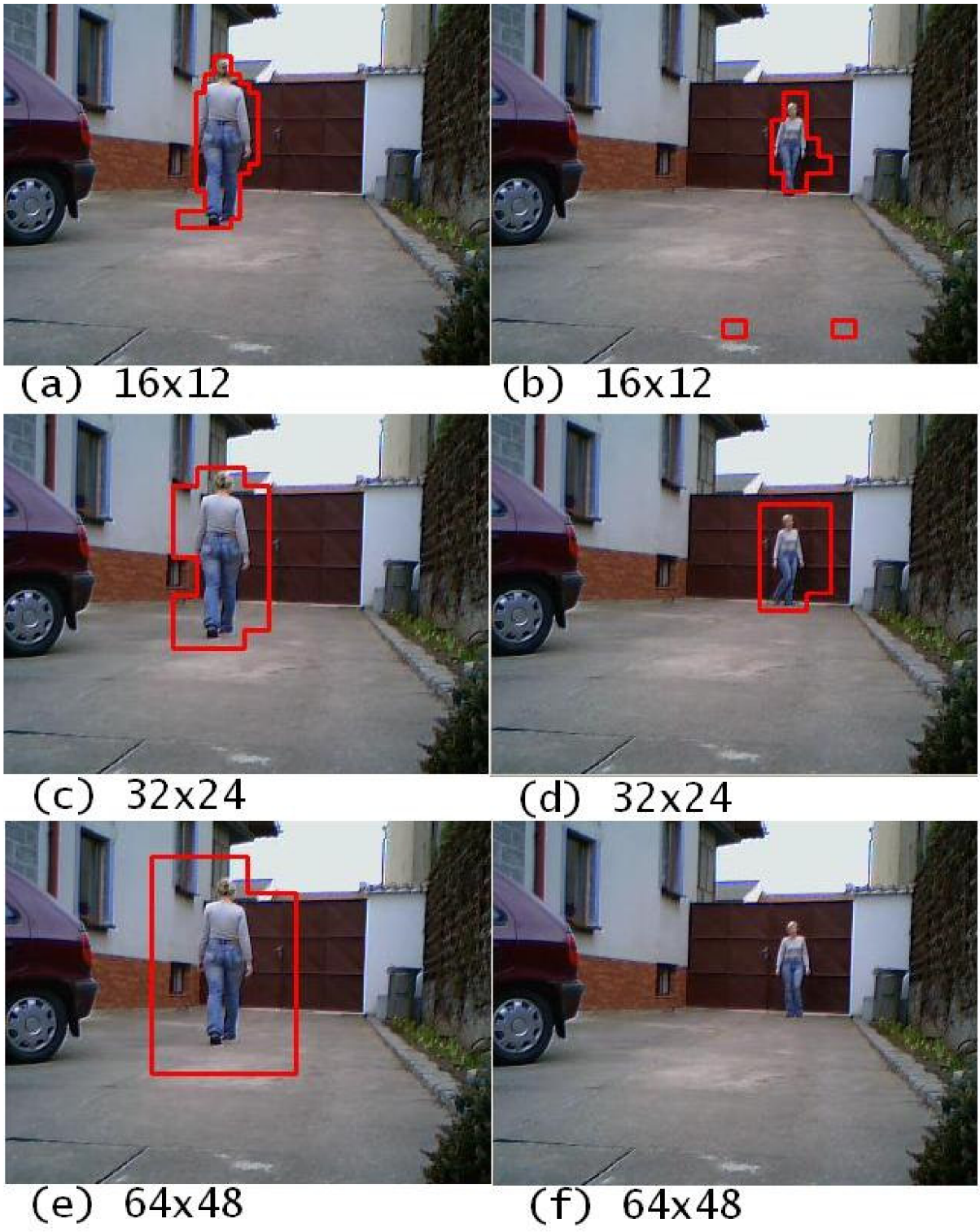
(Obrázek 22) Nepřesná detekce hranic osoby

Avšak pokud se povedlo nastavit hodnotu prahu na ideální hodnotu, byly zjištěny hranice osoby velmi přesně (Obrázek 23a). Při detekci vzdálenějších osob (více než 15 metrů), vykazoval detektor také velmi dobré výsledky (Obrázek 23b). Pokud se budeme zajímat o rychlost

vyhodnocování, je při tomto nastavení vyhodnocení pomalé, tudíž bych ho pro sledování osob v reálném čase nedoporučoval.

Dále jsem zvolil velikost bloku 32x24 (dvakrát větší než u předchozího testu). V tomto případě se hranice pro detekci dala nastavit velmi dobře. Detektor nebyl tak citlivý na změny jako v předcházejícím případě a pohybující se osobu detekoval velmi přesně (Obrázek 23c). Hranice osoby nejsou však tak přesné jako u předcházejícího nastavení. Vzdálenou osobu je detektor také schopen detekovat (Obrázek 23d). Rychlost vyhodnocení byla průměrná. Pro vyhodnocení v reálném čase je tato velikost bloku (s polovičním překrytím) vhodná.

Jako poslední velikost bloku jsem u videa s rozlišením 320x240 zvolil 64x48. Osoby v blízké vzdálenosti jsou detekovány velmi dobře, avšak jejich hranice je velmi nepřesná. Nepřesnost hranice způsobuje příliš velká velikost bloku (Obrázek 23e). Vzdálené osoby detekuje špatně nebo vůbec. Důvodem je, že pohybující se osoba málo změní histogram velkého bloku (Obrázek 23f). Detekce pohybu a vyhodnocení snímku jsou však rychlé, tudíž se toto nastavení hodí pro detekci pohybu osob v reálném čase. Osoby se ale musí pohybovat v blízkosti kamery.



(Obrázek 23) Závislost kvality detekce pohybující se osoby na velikosti bloků u video sekvence s rozlišením 320x240

Výsledky jednotlivých nastavení bloků pro rozlišení videa 320x240 shrnuje Tabulka 1. Z výsledků vyplývá, že optimální velikost bloku je 32x24. Důvodem je dobrá detekce pohybujících se osob na blízké i větší vzdálenosti. Rychlost vyhodnocení je také dobrá. Za čas zpracování celé video sekvence považuji čas od spuštění aplikace po vyhodnocení posledního snímku.

Velikost bloku	Detekce neexistujícího pohybu	Detekce hranic osoby	Detekce vzdálených osob (více než 15 m)	Rychlost detekce (délka videa 26 sekund)
16x12	Velmi často	Výborná	Velmi dobrá	Pomalá – 165 s
32x24	Občas	Dobrá	Dobrá	Dobrá – 60 s
64x48	Velmi málo	Ucházející	Špatná	Velmi rychlá – 30 s

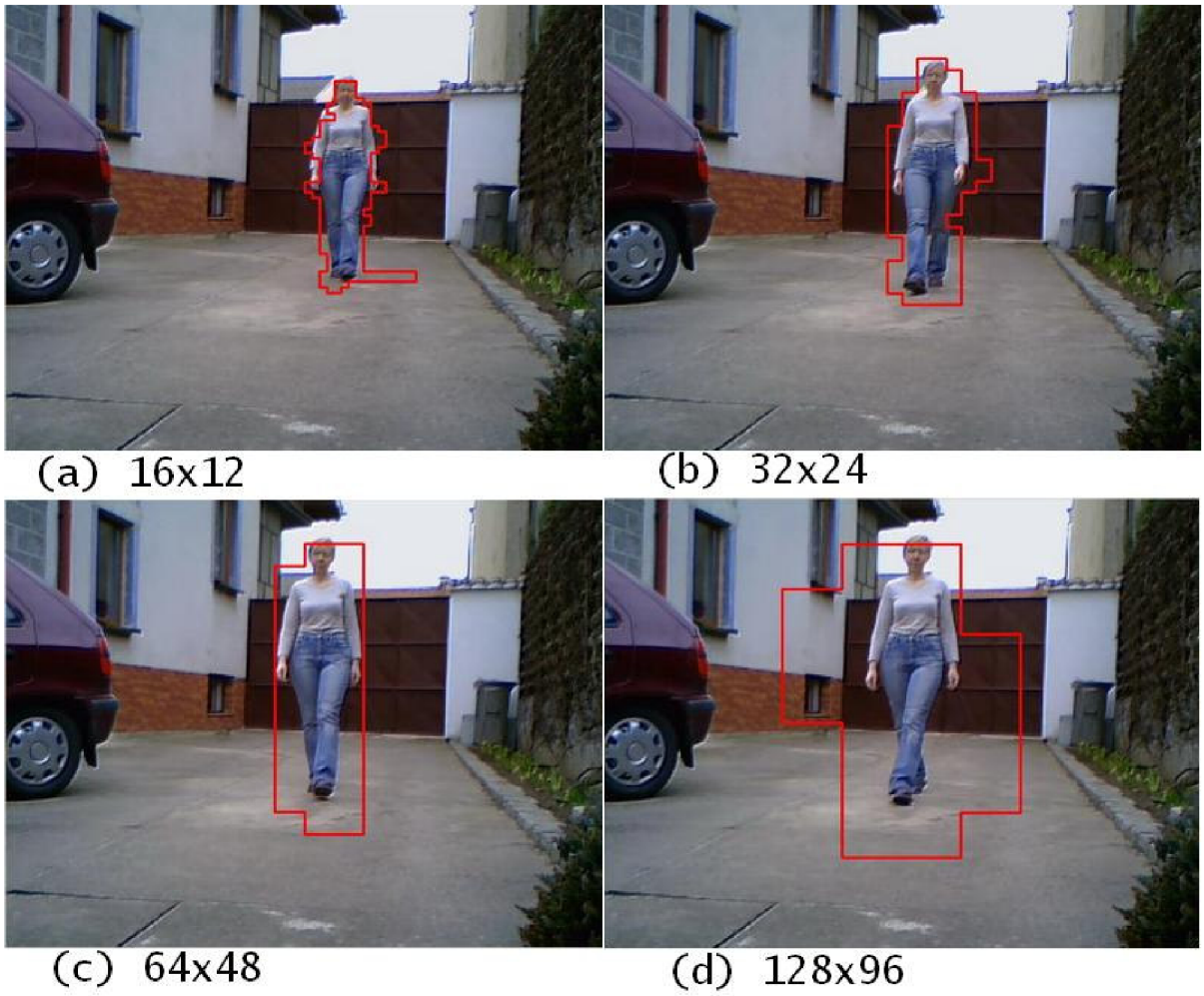
(Tabulka 1) Výsledky nastavení bloků pro video v rozlišení 320x240

Nyní se budu věnovat testování nastavení velikosti bloků u videa s rozlišením 640x480. Nastavení velikosti bloku 16x12 má stejné výhody i nevýhody jako předchozí případ (Obrázek 24a). Ještě více se zde projevuje zpomalení vyhodnocení. Tuto velikost bych pro rozlišení videa 640x480 nedoporučoval.

Pro velikost bloku 32x24 vykazuje detektor osob podobné výsledky jako pro rozlišení 320x240. Hranice osoby jsou určeny celkem přesně, neexistují pohyb je detekován pouze zřídka (Obrázek 24b). Ale protože snímek v rozlišení 640x480 obsahuje čtyřikrát více pixelů než snímek s rozlišením 320x240, je detekce pohybující se osoby zpomalena. Z tohoto důvodu je velikost bloku 32x24 nevhodná pro detekci v reálném čase.

Při rozlišení 320x240 vykazovala velikost bloku 64x48 nepřesné hranice osob a docházelo ke špatné detekci vzdálených osob. U rozlišení 640x480 se dají označit hranice objektu za dobré (Obrázek 24c). Detekce vzdálených osob je také na velmi uspokojivé úrovni. Rychlost zpracování je použitelná pro detekci osob v reálném čase.

Jako poslední jsem otestoval velikost bloku 128x96 (Obrázek 24d). Tato velikost bloku vykazuje stejné vlastnosti jako velikost bloku 64x48 při rozlišení snímku 320x240. Tj. velmi nepřesné hranice objektu, vzdálené osoby detekuje špatně nebo vůbec, velmi rychlé zpracování video sekvence.



(Obrázek 24) Závislost kvality detekce pohybující se osoby na velikosti bloků u video sekvence s rozlišením 640x480

Výsledky jednotlivých nastavení bloků pro rozlišení videa 640x480 shrnuje Tabulka 2. Z výsledků vyplývá, že optimální velikost bloku je 64x48. Důvodem je dobrá detekce pohybujících se osob na blízké i větší vzdálenosti. Rychlost vyhodnocení je také dobrá. Pokud nevyžadujeme detekci v reálném čase, je použitelná i velikost bloku 32x24. S touto velikostí bloku získáme lepší hranici osob a jsou lépe detekovány vzdálené osoby.

Velikost bloku	**Detekce neexistujícího pohybu**	**Detekce hranic osoby**	**Detekce vzdálených osob (více než 15 m)**	**Rychlost detekce (délka videa 15 sekund)**
16x12	Velmi často	Dobrá	Velmi dobrá	Velmi pomalá – 421 s
32x24	Občas	Výborná	Velmi dobrá	Pomalá – 128 s
64x48	Velmi málo	Dobrá	Dobrá	Dobrá – 62 s
128x96	Velmi málo	Ucházející	Špatná	Rychlá – 47 s

(Tabulka 2) Výsledky nastavení bloků pro video v rozlišení 640x480

Z výsledků vyplývá, že velikost bloku podstatně ovlivňuje kvalitu detekce osob a rychlost zpracování video sekvence. Jako ideální velikost bloku se jeví velikost odpovídající jedné desetině velikosti snímku.

5.2 Závislost kvality detekce osob na míře překrytu bloků

V této kapitole se budu zabývat, do jaké míry ovlivňuje velikost překrytu bloků kvalitu detekce osob. Testy budu provádět na videu s rozlišením 320x240, velikost bloků nastavím na 32x24. Ke srovnání histogramů použiji metodu nejmenších čtverců s hodnotou prahu 0,25. Při hodnocení hranic osoby uvažuji dvě kritéria. Prvním je, jak moc vyznačená hranice připomíná siluetu osoby. Druhým kritériem je, jak moc velká část osoby se nachází mimo hranice. Velikost překrytu bloků neovlivňuje detekci vzdálených osob.

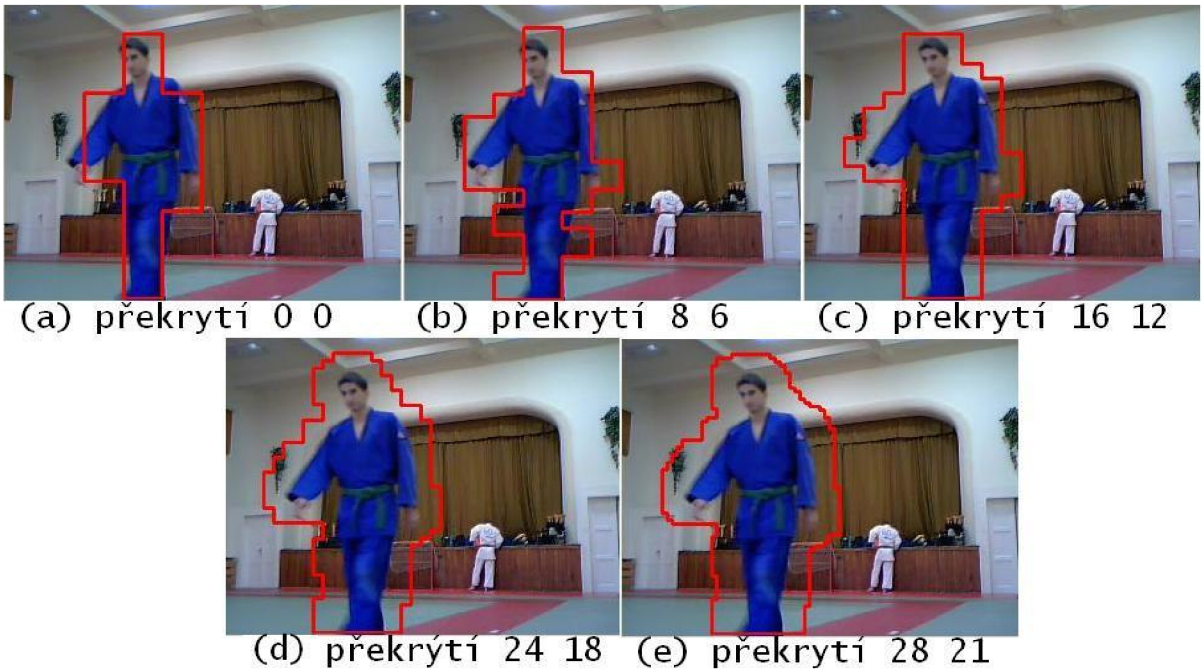
V prvním testu se bloky vůbec nepřekrývají (překryt 0 0). Při tomto nastavení nejsou detekovány okrajové části objektu (Obrázek 25a) a detekce siluety je též špatná. Protože celý snímek je pokryt poměrně malým počtem bloků (100), je vyhodnocení velmi rychlé.

Při nastavení překrytu bloků 8 pixelů na šířku a 6 pixelů na výšku, se kvalita detekce hranic osoby výrazně nezlepšila. Některé hranice jsou již považovány za osobu, ale některé za osobu stále považovány nejsou (Obrázek 25b). Rychlost je stále velmi rychlá (počet bloků je 169).

Další volba překrytu bloků byla polovina velikosti bloku – 16 12. Při tomto překrytu je již celý objekt označen. Silueta objektu je již dobře znatelná (Obrázek 25c). Rychlost vyhodnocení se díky nárůstu počtu bloků (361) zpomalila. Přesto je rychlost vyhodnocení stále dobrá.

Při nastavení překrytu bloků na tři čtvrtiny jejich velkosti (24 18) dochází k výraznějšímu zpomalení vyhodnocení snímku, protože celý snímek je pokryt 1369 bloky. Z toho důvodu je již tato hodnota překrytu bloků špatně použitelná při detekci osob v reálném čase. Silueta osoby je velmi dobře znatelná (Obrázek 25d) a je více detailní než při předchozím nastavení.

Nakonec jsem vyzkoušel hodnotu překrytu 28 21. Počet bloků na jeden snímek je 5329. Vyhodnocení snímku je velmi pomalé. Silueta nevykazuje oproti překrytu 24 18 výraznější zlepšení (Obrázek 25e).



(Obrázek 25) Výsledky detekce s různými překryty bloků

Z testů vyplývá, že pro danou velikost bloků a rozlišení je vhodná velikost překrytu bloků 16 12. Pokud nepožadujeme vyhodnocení v reálném čase nebo vystačíme s menším počtem snímků za sekundu, doporučil bych i hodnotu překrytu 24 18. Výsledky testů shrnuje Tabulka 3.

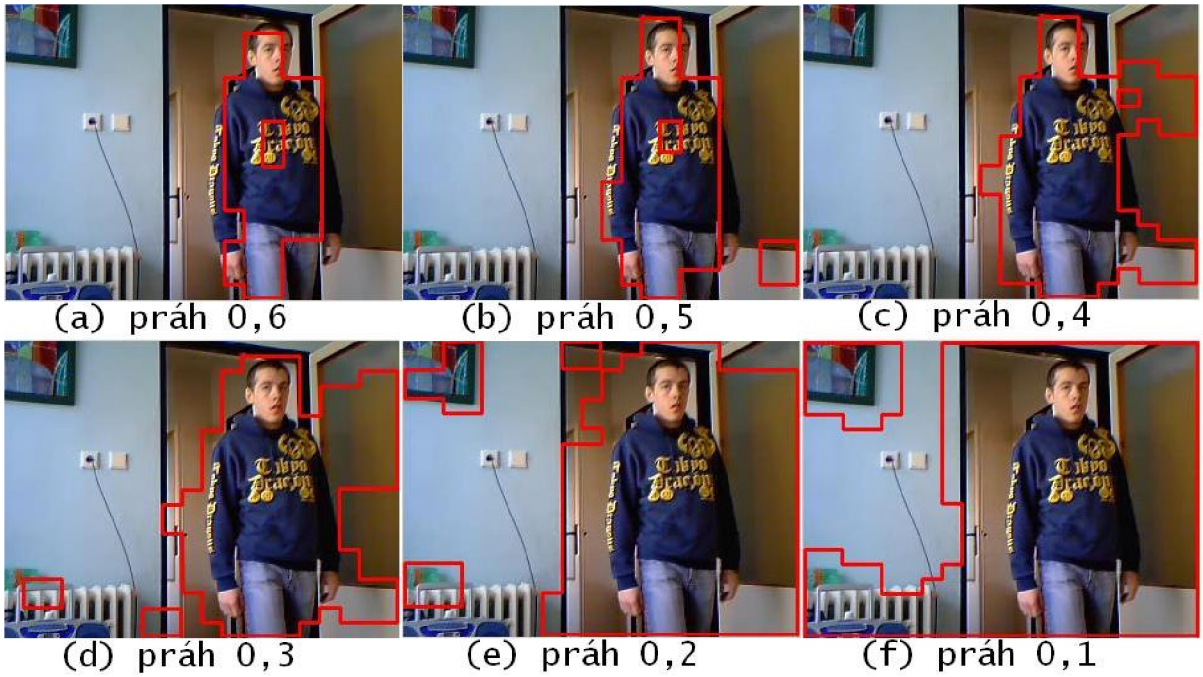
Velikost překrytu bloků	**Nachází se části objektu mimo hranice?**	**Kvalita siluety**	**Rychlost detekce (délka videa 8 sekund)**
0 0	Ano	Špatná	Velmi rychlá – 7 s
8 6	Ano	Ucházející	Rychlá – 10 s
16 12	Ne	Dobrá	Dobrá – 19 s
24 18	Ne	Velmi dobrá	Pomalá – 60 s
28 21	Ne	Velmi dobrá	Velmi Pomalá – 249 s

(Tabulka 3) Výsledky nastavení překrytu bloků pro video v rozlišení 320x240

5.3 Test srovnávacích metod histogramů a hodnoty prahu

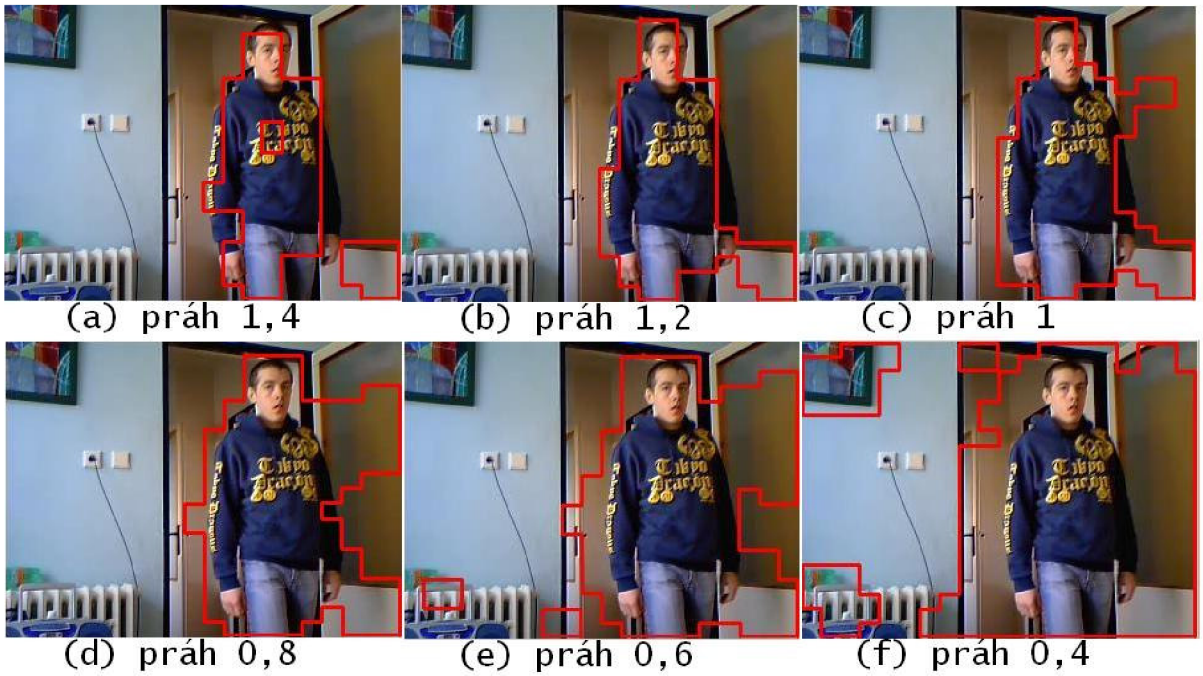
Mezi velmi důležité faktory, které budou ovlivňovat kvalitu detekce, bude bezpochyby patřit hodnota prahu (jak moc se od sebe může histogram pozadí a popředí lišit). Dále v této kapitole otestuji, zda je podstatný rozdíl mezi metodou nejmenších čtverců a metodou logaritmického okolí (popis metod v kapitole 1.2). Testy jsem prováděl s velikostí bloků 32x24 s překrytím 16 12 a hodnotu učicího faktoru jsem nastavil na 0,005.

Nejdříve jsem testoval metodu nejmenších čtverců. Začal jsem s nastavením prahu na hodnotě 0,6 (Obrázek 26a), při této hodnotě je detektor poměrně necitlivý. V některých snímcích špatně detekuje hranice osoby nebo určité části uvnitř hranic osoby považuje za pozadí. Je však málo citlivý ke změnám v okolí. Například nedetekuje odraz osoby ve skleněných dveřích a stíny vrhané osobou (Obrázek 26a). Postupným snižováním prahu se detektor stává citlivějším. Lépe detekuje osobu, ale také detekuje více změn v okolí (Obrázek 26). Pokud hranice prahu klesne pod určitou mez, stává se detektor nepoužitelným, protože stále detekuje pohyb (Obrázek 26f). Nejedná se o skutečný pohyb, ale spíše drobné změny způsobené nedokonalostí snímacího zařízení. Nastavení prahu velmi závisí na okolním prostředí a vzdálenosti sledovaných osob od kamery. Z mých testů vyplývá, že vhodná hodnota prahu se nachází v rozmezí 0,5 až 0,2 (pro velikost bloku 32x24).



(Obrázek 26) Různé hodnoty prahu pro metodu nejmenších čtverců

Jako druhou jsem testoval metodu logaritmické okolí. Začal jsem s nastavením prahu na 1,4 a skončil jsem s nastavením na 0,4. Jak můžete vidět na obrázku (Obrázek 27), výsledky jsou velmi podobné metodě nejmenších čtverců (liší se pouze hodnoty prahu, kdy je daného výsledku dosaženo). U vysokých hodnot prahu je detektor necitlivý (Obrázek 27a). Se snižováním hodnoty prahu jeho citlivost roste, takže lépe detekuje hranice osob, ale detekuje i více změn v okolí. Při poklesu prahu pod určitou mez se opět stává nepoužitelným (Obrázek 27f). Vhodná hodnota prahu pro metodu logaritmického okolí se se pohybuje v rozmezí 1,2 až 0,6 (pro velikost bloku 32x24).



(Obrázek 27) Různé hodnoty prahu pro metodu logaritmické okolí

I když v této práci zveřejňuji výsledky pouze jednoho testu, porovnávací metody a nastavení prahů jsem testoval v několika video sekvencích s různým nastavením velikosti bloků a různými překryty bloků. Z testů jsem vyvodil následující závěry. Volba metody nemá podstatný vliv na chování detektoru. Naproti tomu správné nastavení prahu hraje klíčovou roli. Obě metody jsou přibližně stejně rychlé. Při větším počtu bloků ve snímku (nad 1000) je metoda logaritmické okolí, mírně pomalejší (asi o 2 sekundy, z celkového času vyhodnocení 161 s). Metoda logaritmické okolí má větší rozsah prahu, který mohu nastavit. Nastavení hodnoty prahu závisí také na velikosti bloků (Tabulka 4). Čím je blok menší, tím by měla být hodnota prahu vyšší. Důvodem je, že menší blok obsahuje méně informací v histogramu. Pokud se histogram liší i velmi málo, metoda spočte vysokou hodnotu rozdílu a proto musím nastavit i vyšší práh. U větších bloků je naopak více hodnot obsažených v histogramu. Takže výsledná hodnota rozdílu histogramu, kterou metoda spočte, může být malá, i když jsou histogramy rozdílné. Tudíž je práh vhodné nastavit na nižší hodnotu. Míra překrytu bloků a hodnota prahu se spolu vzájemně neovlivňují.

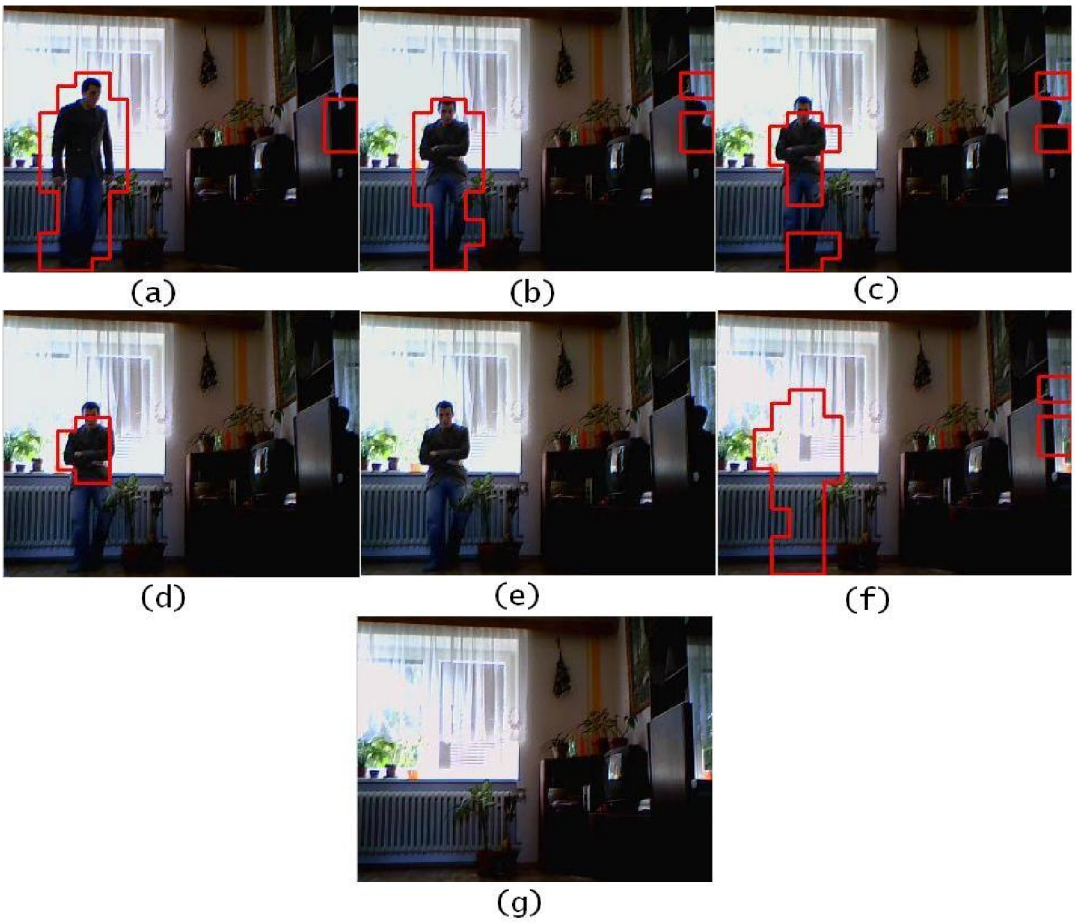
Velikost bloku	Práh nejmenších čtverců	Práh logaritmického okolí
16x12	0,45 – 0,8	1,1 - 2,1
32x24	0,2 – 0,5	0,6 – 1,2
64x48	0,3 – 0,07	0,2 – 0,6

(Tabulka 4) Hodnoty prahů pro nejběžnější velikosti bloků

5.4 Rychlost aktualizace pozadí

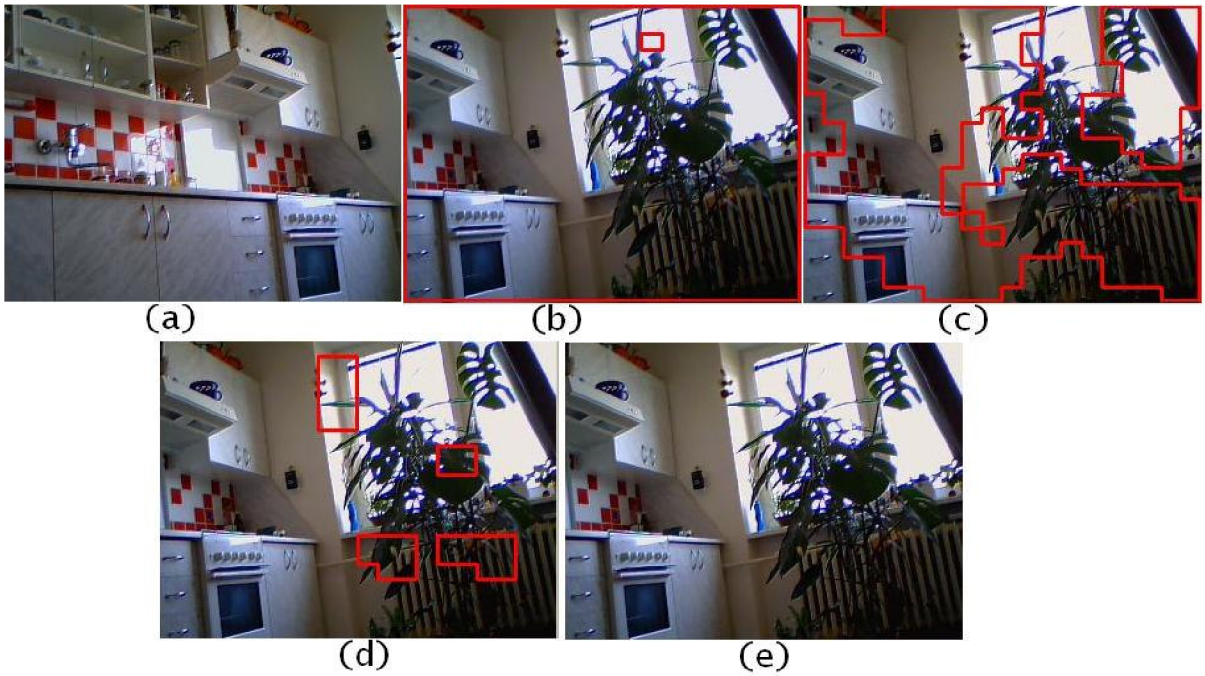
Jak jsem již uvedl v kapitole 2.2, je aktualizace pozadí velmi důležitou součástí pro detekci pohybu. Díky aktualizaci pozadí se dokáže detektor vyrovnat se změnou osvětlení, změnou denní doby, novým statickým objektem i změnou snímaného prostoru. Ve své aplikaci využívám započtení histogramů určitou měrou do pozadí (viz kapitola 2.2), míru započtení nazývám učicí faktor. V této kapitole testuji, jak hodnota učicího faktoru ovlivňuje rychlost aktualizace pozadí.

Rychlost aktualizace testuji pro dvě video sekvence. V první video sekvenci osoba vstoupí do záběru (Obrázek 28a). Opře se o topení a zůstává v záběru nehybná (Obrázek 28b). Postupně se začne stávat součástí pozadí. Nejdříve jsou za pozadí považovány části osoby, jejichž histogramy byly podobné histogramům původního pozadí (Obrázek 28c). Nejdéle se naopak přizpůsobují histogramy, které se od původních histogramů pozadí velmi lišily (Obrázek 28d). Nakonec je celá osoba považována za pozadí (Obrázek 28e). Když se osoba dá opět do pohybu a opustí záběr, dojde opět ke změně pozadí (Obrázek 28f). Osoba, jež byla jeho součástí, je pryč a tudíž to, co se nacházelo za osobou, se považuje za pohybující se objekt. Opět určitou dobu trvá, než se skutečné pozadí přestane považovat za pohybující se osobu (Obrázek 28g).



(Obrázek 28) Osoba se stává statickým objektem

Druhá video sekvence ukazuje, jak se aplikace chová, pokud dojde ke změně snímaného prostoru. Videokamera sleduje určitý prostor (Obrázek 29a). Náhle dojde ke změně snímané scény (náraz do kamery, úmyslné posunutí atd.). Histogramy nového pozadí se liší od histogramů původního pozadí (Obrázek 29b). Opět dochází k postupnému přizpůsobení. Nejdříve se přizpůsobí bloky, které měly velmi podobné pozadí (Obrázek 29c). Nejdéle se budou přizpůsobovat bloky s velmi rozdílnými histogramy (Obrázek 29d). Po určité době, v závislosti na učicím faktoru, se přizpůsobí celé nové pozadí (Obrázek 29e).



(Obrázek 29) Změna snímaného prostoru

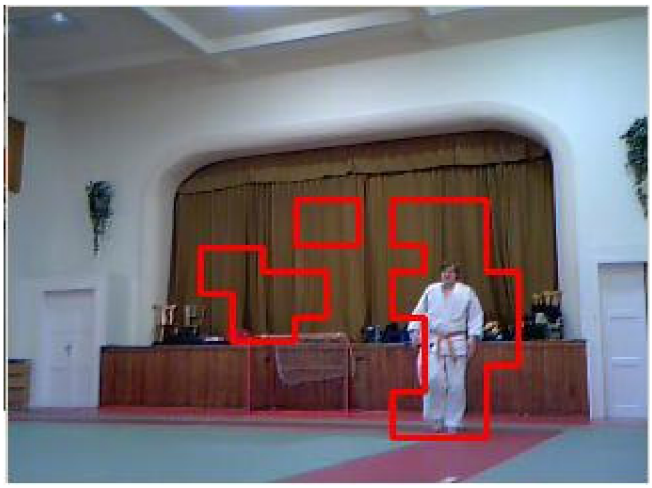
Tabulka 5 uvádí, kolik snímků je potřeba, než se ustálí změny v pozadí. Kromě hodnoty učicího faktoru je potřeba brát i v potaz nastavení prahu pro porovnání histogramů. Čím nižší práh bude nastaven, tím bude detektor citlivější na změny a i počet snímků nutných k přizpůsobení pozadí vzroste. Naopak vyšší hodnota prahu bude vést ke snížení potřebného počtu snímků k ustálení změn v pozadí. V testu využívám u obou video sekvencí pro porovnávání histogramů metodu nejmenších čtverců s hodnotou prahu 0,27. Velikost bloků jsem nastavil na 32x24 s vzájemným překrytím 16 12.

Hodnota učiciho faktoru	Počet snímků k ustálení změn v první video sekvenci	Počet snímků k ustálení změn v první video sekvenci
0,1	11	14
0,05	38	33
0,02	81	75
0,01	170	155
0,005	332	333
0,002	832	786
0,001	1802	1594

(Tabulka 5) Rychlost aktualizace pozadí v závislosti na učicím faktoru

Chtěl bych zdůraznit, že hodnota učiciho faktoru je dvousečná zbraň. Pokud nastavím vysokou hodnotu, budou se histogramy aktuálního snímku vysokou měrou započítávat do histogramu pozadí. To může způsobit, že pomalu se pohybující nebo vzdalující (přibližující) se osoba bude brána za pozadí (Obrázek 30). Výhodou bude, že detektor bude stabilní vůči změnám okolí. Pokud nastavím hodnotu příliš nízkou, stane se detektor citlivým na okolní změny, místo toho, aby tyto změny zahrnul co nejrychleji do pozadí (Obrázek 31). Nedá se přesně říct, jaké rozmezí hodnot učicího faktoru je ideální. Velmi záleží na tom co má aplikace sledovat a v jakých podmínkách ji používám.

(Obrázek 30) Příliš vysoká hodnota učicího faktoru



(Obrázek 31) Příliš nízká hodnota učicího faktoru

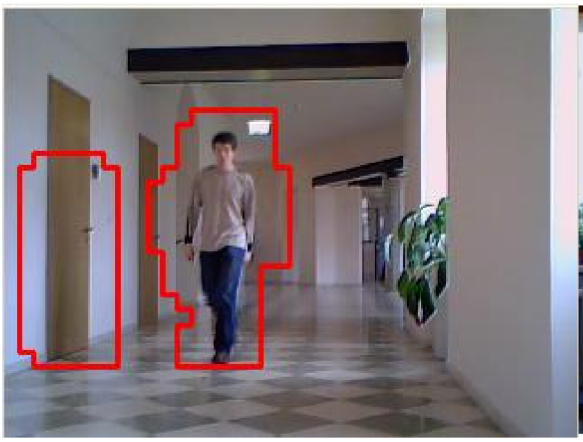
Velikost učicího faktoru má důležitou roli při startu aplikace. Protože se při startu histogramy pozadí počítají pouze z jednoho snímku, trvá určitou dobu, než se celé pozadí ustálí. Čím je hodnota učicího faktoru vyšší, tím se pozadí po startu aplikace rychleji ustálí.

5.5 Speciální testy

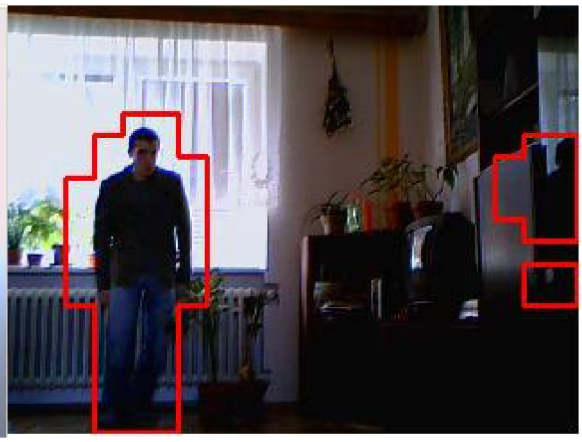
Tato kapitola popisuje, jak se detektor chová v různých speciálních situacích. Například při změně osvětlení, při otřesech kamery atd..

5.5.1 Detekce stínů a odrazů

V této kapitole zjišťuji, jak se detektor vyrovnává se stíny, které vrhají osoby, a s různými odrazy. Detektor bohužel kromě samotných osob detekuje i jejich stíny. Detekci stínů jsem schopen ovlivnit hodnotou prahu. Pokud nastavím vysoký práh, bude detekováno minimum stínů. Naopak při nízkém prahu bude detektor kromě pohybující se osoby detekovat i její stín (Obrázek 32). Odrazy mohou vznikat například od skel, zrcadel nebo naleštěné podlahy (Obrázek 33). Jejich detekci ovlivňuji stejně jako detekci stínů.



(Obrázek 32) Detekován stín



(Obrázek 33) Detekován odraz ve skle

Při testování jsem porovnával metodu detekce pohybu LBP s metodou jasových histogramů. Zjistil jsem, že metoda jasových histogramů je více citlivá na stíny a na odrazy (Obrázek 34). Detekuje i velmi drobné změny stínů a světla. Protože ve své aplikaci považuji detekci stínů a odlesků za nechtěnou, je vhodnější z tohoto hlediska využít metodu LBP.

(Obrázek 34) Ukázka citlivosti metody jasových histogramů

5.5.2 Chování detektoru při změně osvětlení

Při testování citlivosti na změnu detektoru jsem scénu osvětlil dvěma světly. Nejdříve jsem se pohyboval po scéně (Obrázek 35a), když byla obě dvě světla rozsvícena. Poté jsem zhasl jedno ze světel a opět se pohyboval po scéně (zhasl jsem ve snímku 107).

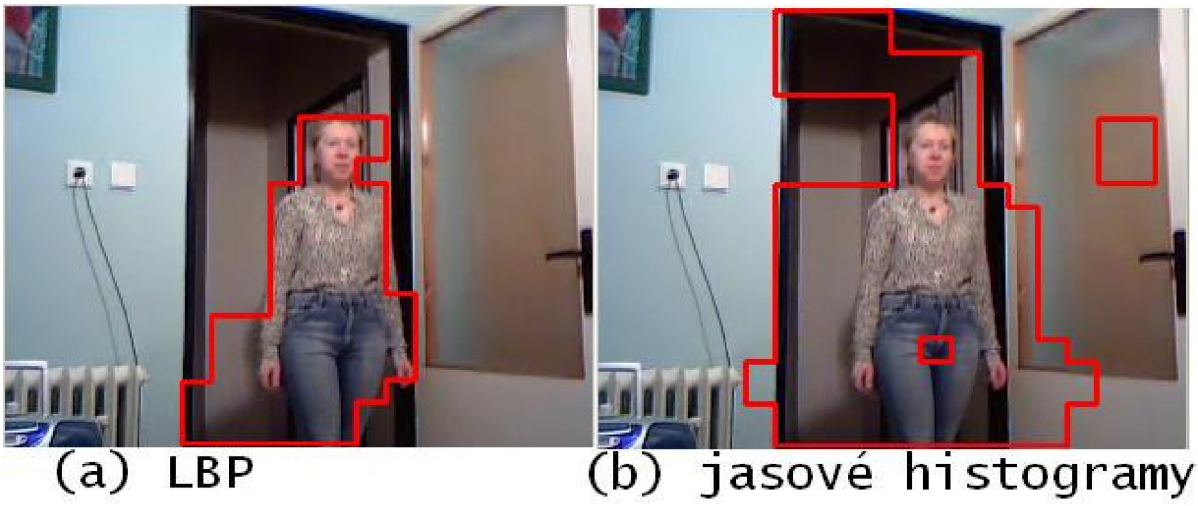
Pokud jsem k detekci pohybující se osoby použil metodu detekce pohybu LBP, tak při zhasnutí světla byl detekován i neexistující pohyb, ale pouze ve velmi malé míře (Obrázek 35b). Za několik snímků se metoda se změnou osvětlení bez problémů vyrovnala (Obrázek 35c a 35d).

Metoda jasových histogramů se se změnou osvětlení vyrovnala mnohem hůře. Detekovala více neexistujícího pohybu (Obrázek 35e) a i přizpůsobení novému osvětlení jí trvalo déle (Obrázek 35f a 35g).

(Obrázek 35) Reakce metody LBP a jasových histogramů na změnu osvětlení (s – snímek, JH – metoda jasových histogramů)

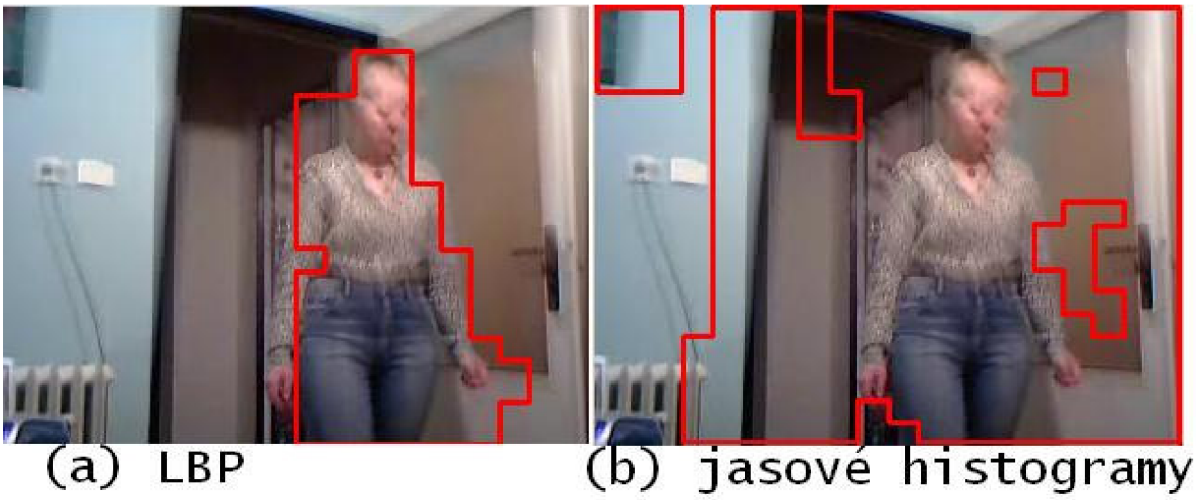
5.5.3 Vliv otřesů na detekci

S drobnými otřesy kamery se můžeme setkat velmi často. Například když je kamera umístěna ve venkovním prostředí, mohou poryvy větru způsobit roztřesení snímaného obrazu. Dále mohou otřesy způsobovat těžká nákladní auta nebo projíždějící vlak. Ve testu sem použil video sekvenci, při jejímž snímání jsem roztřásl kameru. V prvním případě jsem kameru roztřásl mírně (podobné otřesy by mohl způsobit vítr). Detekce pohybujících se osob byla pomocí LBP o poznání lepší než detekce pomocí jasových histogramů (Obrázek 36). I v roztřeseném obraze byla metoda LBP schopna velmi dobře detekovat pohybující se osobu.



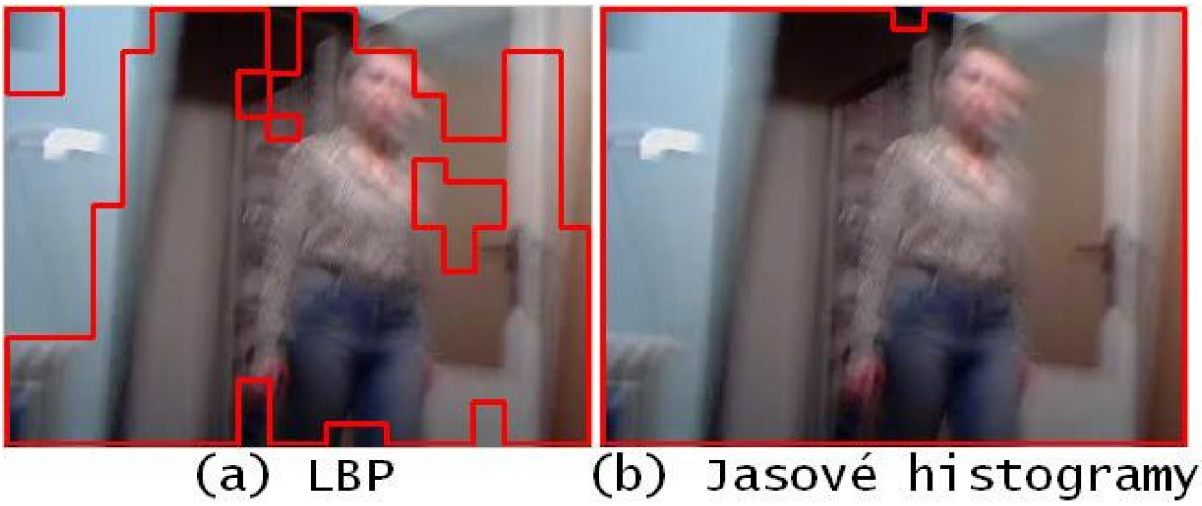
(Obrázek 36) Mírné otřesy kamery

Při druhém testu jsem kameru roztřásl více (otřesy podobného typu by mohl způsobit velmi silný vítr nebo vlak projíždějící v těsné blízkosti). Opět se prokázala lepší kvalita metody LBP oproti metodě jasovým histogramům. Metoda LBP se s otřesy vyrovnala poměrně dobře, ale metoda jasových histogramů je již zcela nepoužitelná (Obrázek 37).



(Obrázek 37) Středně silné otřesy kamery

V poslední video sekvenci otřesy kamery byly již velmi silné (jako při zemětřesení). V tomto případě se ani metoda LBP nebyla schopna vyrovnat s otřesy a stala se pro detekci nepoužitelnou (Obrázek 38).

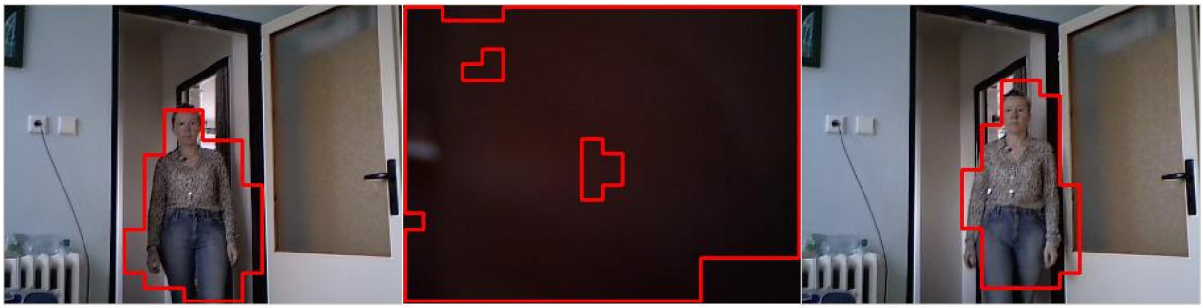


(Obrázek 38) Velmi silné otřesy kamery

5.5.4 Zakrytí kamery

Jako poslední jsem otestoval vlastnost aplikace, kterou nazývám zatmění. Pokud je v aplikaci povoleno zatmění, tak při překročného určitého počtu bloků v nichž byl detekován pohyb, se přestane tento pohyb v aplikaci zobrazovat (nepovažuje se za pohyb). Tohoto se dá využít, například pokud je kamera krátkodobě zakryta.

Pokud není zatmění povoleno, tak pří zakrytí kamery se tato změna obrazu bere jako pohyb (Obrázek 39). Pokud je zatmění povoleno, tak se při zakrytí tato změna nebere jako pohyb (Obrázek 40). Hodnota zatmění je nastavitelná.



(Obrázek 39) Bez povoleného zatmění

(Obrázek 40) Se zatměním 0,8

Pokud by kamera zůstala zakryta delší dobu, tak by se předmět, který zakrývá kameru, stal pozadím. I když pohyb není zobrazován, je stále prováděna aktualizace pozadí. Pozadí se aktualizuje,

protože by se nemuselo jednat o zakrytí kamery, ale například o posunutí kamery (změnil by se snímaný prostor).

Závěr

Cílem bakalářské práce bylo prozkoumat různé možnosti detekce pohybu osob, seznámit s nimi čtenáře, vytvořit aplikaci, která bude umožňovat detekci pohybujících se osob a otestovat tuto aplikaci v různých podmínkách.

V teoretické části jsem popsal řadu metod pro detekci pohybu. Uvedl jsem jejich výhody i nevýhody. Dále jsem se v teoretické části zabýval možnostmi aktualizace pozadí a určením místa ve snímku, kde se osoba pohybuje.

Z mnou uvedených metod pro detekci pohybu mě nejvíce zaujala metoda Local Binary Patterns. Tuto metodu jsem implementoval ve své aplikaci, protože se jedná o poměrně novou, a tudíž nerozšířenou metodu, a protože mě velmi zaujaly její vlastnosti, které se jevily pro mou aplikaci jako velmi vhodné. Protože tato metoda pracuje s histogramy a neurčuje přesné místo, kde k pohybu dochází, musel jsem vyřešit problém, jak určit místo, kde k pohybu dochází. Rozdělil jsem snímek do několika bloků a v nich jsem poté detekoval pohyb. Zbývalo vyřešit, jak tyto bloky spojit do jednolitého objektu, který bude představovat pohybující se osobu. Ke spojení bloků jsem využil metody detekce kontur.

Stěžejním bodem této práce bylo vytvoření a otestování aplikace pro detekci pohybujících se osob. Aplikaci jsem implementoval v programovacím jazyce C++. Testy jsem poté prováděl v různých prostředích, s různými osobami. Z mnou provedených testů vyplývá, že nejlepší nastavení pro video sekvenci s rozlišením 320x240 je následující: velikost bloku 32x24, s poloviční překrytím bloků (16 12). Práh pro porovnání histogramů metodou nejmenších čtverců bych doporučil v rozmezí 0,2 až 0,5. U metody logaritmické okolí bych doporučil hodnotu prahu 0,6 až 1,2.

Velmi doporučuji pro detekci pohybu využit metodu LBP. V mnou provedených testech jsem prokázal přednosti metody LBP oproti metodě jasových histogramů. LBP si lépe poradí s odlesky, stíny, změnou osvětlení i otřesy kamery.

Z pohledu dalšího vývoje by bylo dobré implementovat v aplikaci více metod pro detekci pohybujících se objektů. Dále bych doporučoval zaměřit se na rozpoznávání osob. V současném stavu aplikace detekuje pouze pohybující se objekt, ale není schopna rozlišit, zda se jedná o osobu, zvíře nebo auto. Další možnou cestou by bylo zaměřit se předvídáním pohybu již detekovaného objektu.

Literatura

[1] ŽÁRA, J., BENEŠ, B., FELKEL, P.: Moderní počítačová grafika. ComputerPress, 2004, 448 s., ISBN: 80-7226-049-9.

[2] ŠPANĚL, M.: Rozpoznávání gest ve video sekvencích. http://www.fit.vutbr.cz/~spanel/dp/spanel_dp_2003.pdf, (2003)

[3] Barevný model http://cs.wikipedia.org/wiki/Barevn%C3%BD_model, (2008)

[4] RGB model http://en.wikipedia.org/wiki/RGB, (2008)

[5] Ústav počítačové grafiky a multimédií: Základy počítačové grafiky - barevné modely, 2007

[6] OJALA, T., PIETIKÄINEN, M., MÄENPÄÄ, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. http://www.ee.oulu.fi/research/imag/texture/publications/show_pdf.php?ID=94, (2002)

[7] HEIKKILÄ, M., PIETIKÄINEN, M., HEIKKILÄ, M.: A texture-based method for detecting moving objects. http://www.ee.oulu.fi/mvg/publications/show_pdf.php?ID=533, (2004)

[8] JELÍNEK, T.: Detekce pohybujících se objektů ve video sekvenci. http://www.fit.vutbr.cz/study/DP/rpfile.php?id=5197, (2007)

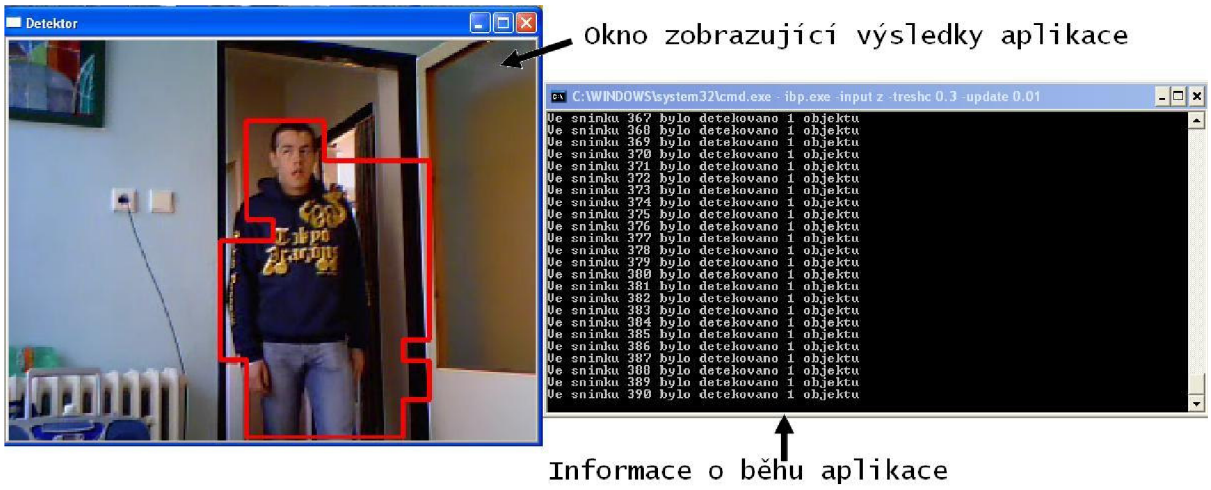
[9] Dokumentace k OpenCV http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/, (2008)

[10] Dokumentace k TinyXML http://www.grinninglizard.com/tinyxmldocs/index.html, (2008)

Seznam příloh

Příloha 1: Ovládání aplikace Detektor

Aplikace Detektor se spouští z příkazového řádku. Po spuštění se objeví okno, ve kterém se zobrazují výsledky pro právě zpracovávanou video sekvenci. Do okna příkazového řádku se vypisují informace o právě zpracovávaném snímku. Pokud je zapnuta detekce objektů, je vypisována i informace o počtu detekovaných objektů (Obrázek 41).



(Obrázek 41) Vzhled aplikace

Pokud zpracovávám video sekvenci snímanou přímo webkamerou, snímání ukončím stiskem klávesy Esc. V zobrazovacím okně zůstane zobrazen poslední zpracovaný snímek. Aplikaci poté ukončím stiskem libovolné klávesy.

Při zpracování video sekvence načtené z video souboru mohu zpracování ukončit stiskem klávesy Esc nebo počkat, až bude zpracován poslední snímek video sekvence. V obou případech zůstane zobrazen poslední zpracovaný snímek. Aplikaci opět ukončím stiskem libovolné klávesy.

Chybová hlášení se vypisují na standartní chybový výstup. Seznam chybových hlášení najdete v souboru error.h a texty chybových hlášení v souboru error.cpp.

Aplikace se dá spustit s následujícími parametry:

detektor.exe [-input název video souboru] [-output název video souboru] [-xmloutput název xml souboru] [-LBP | -brightness] [-hc chisquare | loglikelihood] [-treshc práh] [-treshl práh] [-size šířka výška] [-overlap šířka výška] [-update hodnota učiciho faktoru] [-blockvis | -objectvis] [-blackout hodnota] [-h]

Parametr	Popis parametru
-input	Pokud zadám tento parametr, bude vstup do aplikace brán místo z webkamery z video souboru zadaného jména. Aplikace podporuje video soubory formátu avi. Jméno vstupního video souboru je nutné zadat bez přípony avi (aplikace si sama příponu doplňuje). Pokud chci, aby byl vstup do aplikace snímán webkamerou, nezadávám tento parametr.

-output	Výsledky aplikace, jež, jsou zobrazovány v okně, se zároveň budou ukládat do video souboru formátu avi. Název výstupního video souboru je nutné zadat bez přípony avi. Do video souboru je zapisováno rychlostí 25 snímků za sekundu.
-xmloutput	Výsledky získané aplikací se budou, kromě zobrazení v okně, také zapisovat do xml souboru. Název xml souboru je nutné zadat bez přípony xml. Ukázku xml souboru lze nalézt v Příloze 2.
-LBP	Pro detekci pohybu se bude používat metoda LBP. Tato metoda je nastavena jako výchozí.
-brightness	Pro detekci pohybu se bude používat metoda jasových histogramů.
-hc	Tento parametr určuje, jaká metoda se bude používat pro porovnávání histogramů. Pokud zvolím **chisquare**, bude se používat metoda nejmenších čtverců. Při volbě **loglikelihood** se bude používat metoda logaritmické okolí.
-treshc	Nastaví práh metody nejmenších čtverců na zadanou hodnotu.
-treshl	Nastaví práh metody logaritmického okolí na zadanou hodnotu.
-size	Tento parametr nastavuje velikost bloků. První zadaná hodnota určuje šířku bloku, druhá určuje jeho výšku. Hodnoty nesmí být menší než 1 nebo větší než je velikost snímku.
-overlap	Parametr slouží k nastavení překrytu bloků. První hodnota udává překryt bloků na šířku (na ose x), druhá hodnota udává překryt bloků na výšku (na ose y).
-update	Nastaví učicí faktor na zadanou hodnotu. Hodnota musí být v intervalu <0,1>. Pokud bude zadána hodnota 0, nebudou se histogramy aktuálního snímku vůbec započítávat do histogramů pozadí. Při nastavení na hodnotu 1 budou histogramy pozadí nahrazeny histogramy aktuálního snímku.
-blockvis	Budou se zobrazovat bloky, v nichž došlo k pohybu. Všechny pixely bloku, v němž byl zjištěn pohyb, budou obarveny červenou barvou (dojde k nastavení červené složky pixelu na maximální hodnotu).
-objectvis	Budou se detekovat pohybující se objekty. Pohybující se objekty budou ohraničeny červenými hranicemi.
-blackout	Tento parametr povoluje detekci zatmění (kapitola 4.4) a nastaví hodnotu zatmění. Hodnota určuje, kolik procent obrazu se musí změnit, aby se tato změna nebrala jako pohyb. Hodnota se zadává v intervalu <0,1> (1 - celý obraz, 0 - nic se nebude brát jako pohyb).
-h	Vypíše nápovědu k aplikaci.

(Tabulka 6) Popis parametrů, s kterými lze aplikaci detektor spustit

Pokud nejsou zadány žádné parametry (spuštěno pouze detektor.exe), je aplikace spuštěna ve výchozím nastavení. Vstup je brán z webkamery, zobrazuje se pouze do výstupního okna. Pro detekci

pohybu se používá metoda LBP. Porovnání histogramů se provádí metodou nejmenších čtverců, která má nastaven práh na hodnotu 0,4. Velikost bloků je 32x24, se vzájemným překrytím bloků 16 12. Detekují se pohybující se objekty a zatmění je vypnuto. Výchozí nastavení odpovídá spuštění: detektor.exe -LBP -hc chisquare -treshc 0.4 -size 32 24 -overlap 16 12 -objectvis

Pokud zadám nebo změním pouze jeden parametr (případně zadám některý, který není ve výchozí konfiguraci), zůstanou ostatní parametry nastaveny na výchozí nastavení. Spustím aplikaci následovně: detektor.exe -size 64 48. Změní se pouze velikost bloků (odpovídalo by spuštění: detektor.exe -LBP -hc chisquare -treshc 0.4 -size 64 48 -overlap 16 12 -objectvis).

Pokud zadávám velikost bloků a překryt, musím si uvědomit, že překryt nesmí být větší než je velikost bloku nebo menší než nula (aplikace vypíše chybu při tomto nekorektním nastavení). Dále doporučuji zadávat parametry v pořadí -size -overlap.

Příloha 2: Ukázka výstupního xml souboru

```
<?xml version="1.0" ?>
<Detector_setting>
    <Detect_metod metod="LBP" />
    <Image_size width="320" height="240" />
    <Block_size width="32" height="24" />
    <Overlapping width_overlapping="16" height_overlapping="12" />
    <Histogram_compare_metod metod="Chi-square">
        <Treshhold value="0.4" />
    </Histogram_compare_metod>
    <Bins_update value="0.01" />
    <Blackout enabled="false" />
</Detector_setting>
<Frame number="1" count_of_blocks="0" count_of_detected_object="0" />
<Frame number="2" count_of_blocks="0" count_of_detected_object="0" />
<Frame number="3" count_of_blocks="0" count_of_detected_object="0" />
<Frame number="4" count_of_blocks="3" count_of_detected_object="1">
    <Block number_of_block="161" xmin="144" xmax="120" ymin="96" />
    <Block number_of_block="180" xmin="144" xmax="132" ymin="108" />
    <Block number_of_block="199" xmin="144" xmax="144" ymin="120" />
</Frame>
<Frame number="5" count_of_blocks="5" count_of_detected_object="1">
    <Block number_of_block="179" xmin="128" xmax="132" ymin="108" />
    <Block number_of_block="180" xmin="144" xmax="132" ymin="108" />
    <Block number_of_block="198" xmin="128" xmax="144" ymin="120" />
    <Block number_of_block="199" xmin="144" xmax="144" ymin="120" />
    <Block number_of_block="237" xmin="144" xmax="168" ymin="144" />
</Frame>

<Frame number="6" count_of_blocks="2" count_of_detected_object="1">
    <Block number_of_block="180" xmin="144" xmax="132" ymin="108" />
    <Block number_of_block="199" xmin="144" xmax="144" ymin="120" />
</Frame>
<Frame number="7" count_of_blocks="0" count_of_detected_object="0" />
<Frame number="8" count_of_blocks="0" count_of_detected_object="0" />
```

(Obrázek 42) Příklad xml souboru

Příloha 3: Popis příkladů obsažených na DVD

V této příloze popisuji, jak spustit ukázkové příklady obsažené na DVD. Dále uvádím požadavky nutné ke spuštění těchto příkladů.

Aby mohly být spuštěny příklady na DVD, je potřeba mít nainstalován **DivX 6.1** nebo vyšší, protože ukázková videa vyžadují ke svému spuštění kodek. Pokud nebudete mít kodek nainstalován, bude zobrazovat aplikace hlášení, že se nepodařilo otevřít příslušný video soubor. Instalátor pro DivX naleznete na DVD ve složce **Instal\DivXInstaller.exe**.

Předpřipravené příklady lze spustit pomocí bat souborů. Pokud budete chtít posléze s vlastními videi a nastaveními pracovat, tak spusťte soubor detektor.exe s příslušnými parametry.

Název příkladu	Popis
LBPvsJH.bat	Ukázka různých metod detekce pohybu. V první video sekvenci je osoba detekována pomocí metody LBP. V druhé video sekvenci je k detekci pohybu použita metoda jasových histogramů
CSvsLL.bat	Ukázka různých metod porovnání histogramů. V první video sekvenci se histogramy porovnávají pomocí metody nejmenších čtverců. V druhé video sekvenci se porovnávají pomocí metody logaritmického okolí.
Prah.bat	Tento příklad ukazuje, jak výrazně může hodnota prahu ovlivnit kvalitu detekce osob. V testu jsou nastaveny pro jednu video sekvenci tři různé prahy. Prvním případě je práh nastaven na vysokou hodnotu, v druhém na ideální hodnotu a ve třetím na příliš nízkou hodnotu.
VelikostBloku.bat	Ukázka, jak velikost bloku ovlivňuje kvalitu detekce pohybující se osoby. Obsahuje několik testů pro různé velikosti bloku (16x12, 32x24, 64x48). Bloky se vzájemně překrývají vždy z poloviny.
PrekrytBloku.bat	Test různých velikostí překrytu bloků. Testují se zde čtyři různé hodnoty překrytu: 0 0, 8 6, 16 12 a 24 18. Velikost bloku je 32x24.
Zobrazeni.bat	Ukázka, jak je aplikace schopna zobrazovat výstup uživateli. První test ukazuje výstup metodou bloků s detekovaným pohybem, druhý test ukazuje výstup na základě detekce objektů.
Pozadi.bat	Ukázka, jak hodnota učicího faktoru ovlivňuje rychlost aktualizace pozadí. Testy probíhají pro hodnoty učicího faktoru 0.1, 0.01 a 0.001.
Svetlo.bat	Ukázka reakce metody detekce pohybu na náhlou změnu osvětlení. V prvním testu je k detekci pohybu použita metoda LBP, v druhém testu metoda jasových histogramů.
Otresy.bat	V tomto testu se zkoumá reakce metod detekce pohybu na různou sílu otřesů kamery. Otřesy jsou odstupňovány podle síly: slabé, střední a silné. Pro každou sílu otřesů je zkoumáno, jak metoda LBP a metoda jasových histogramů detekuje pohyb.
Zakryt.bat	Ukázka funkce zatemnění. V prvním testu je scéna zkoumána bez povoleného parametru zatemnění. V druhém testu je nastaven parametr zatemnění na hodnotu 0,7.

(Tabulka 7) Popis příkladů obsažených na DVD