VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce chování chodců z videosekvencí

Detection of Pedestrians' Behaviour from Videosequences

2015 Martin Gold

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Gold**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce chování chodců z videosekvencí
Detection of Pedestrians' Behaviour from Videosequences

Zásady pro vypracování:

Cílem diplomové práce je vytvořit software pro analýzu videosekvencí obsahujících pohybující se lidské postavy. Analýzou se rozumí zjištění trajektorií pohybujících se osob a získání informací, které lze z nalezených trajektorií určit (např. neobvyklý směr a rychlost pohybu).

V diplomové práci proveďte následující:
1. Navrhněte/zvolte vhodnou metodu detekce postav ve videosekvencích.
2. Navrhněte způsob sledování osob (hledání a vyhodnocování trajektorií).
3. Navržené řešení realizujte v C/C++.
4. Řešení řádně experimentálně prověřte.

Způsob a výsledky řešení popište v textové části práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 21. ledna 2015 .....

Rád bych na tomto místě poděkoval mému vedoucímu diplomové práce doc. Dr. Ing. Eduardu Sojkovi za pomoc a užitečné rady při tvorbě této práce.

Abstrakt

Hlavním cílem této diplomové práce je vytvořit software, který by sloužil pro analýzu pohybu lidské postavy ve videosekvencích. Tato analýza bude zaměřena především na informace získané pomocí trajektorie detekované osoby. Vyhodnocovat se v tomto případě bude především rychlost a neobvyklý směr lidského pohybu.

Klíčová slova: C++, OpenCV, HOG, SVM, detekce, trajektorie, pohyb, osoba

Abstract

The main goal of this diploma work is to create software, which would analyze human figure in video sequences. This analysis will be especially aim on informations obtained from trajectory detected person. In this case we will primarily evaluate speed and unusual movement of human.

Keywords: C++, OpenCV, HOG, SVM, detection, trajectory, movement, person

Seznam použitých zkratek a symbolů

HOG	–	Histogram of Oriented Gradients
SVM	–	Support Vector Machine
RGB	–	barevný model založený na skládání tří základních barev (červené, zelené a modré)
YUV	–	barevný model založený na skládání dvou chromatických složek (UV) a jedné jasové složky (Y)
LAB	–	barevný model založený na složení jasové složky (L) a barevných složek (AB)
OpenCV	–	knihovna pro zpracování počítačového vidění (open source computer vision)
PCA	–	Principal Components analysis
SIFT	–	Scale-invariant feature transform
CUDA	–	Compute Unified Device Architecture

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

V dnešní době se stále více úsilí věnuje rozvoji a zdokonalování robotiky a různých autonomních systémů. Aby však jakýkoliv robot nebo autonomní systém byl schopen samostatného myšlení a rozhodování, musí nejprve co nejlépe analyzovat své okolí. I lidé totiž pomocí zraku nejprve získají viditelné informace o svém okolí a ty se následně v mozku pomocí složitých algoritmů zpracují na informace, podle kterých se rozhodujeme, nebo které nás například mohou varovat před blížícím se nebezpečím. Aby se robotika k tomuto konceptu co nejvíce přiblížila, jsou dnešní stroje často vybaveny kamerami, které zastupují funkci zraku. Hlavním problémem v tomto konceptu je však skutečnost, jakým způsobem zpracovat digitální obraz pořízený kamerou, abychom z něho získali potřebná data a přiblížili se tak fungování lidského mozku. V tomto odvětví se však nemusíme setkávat pouze s mobilními roboty, kteří analyzují překážky, ale například i s autonomními kamerovými systémy, které by byly schopny vyhodnocovat rizika a situace ve sledovaném okolí.

V této diplomové práci se konkrétně zabývám problematikou detekování osob ve videosekvencích a následnou analýzou jejich chování. Cílem je co nejlépe navrhnout a naimplementovat software, který by tuto problematiku řešil. Jako hlavní stavební kámen jsem využil knihovnu OpenCV, která poskytuje velké množství funkcí zaměřených na počítačové vidění a s ním spojené zpracování obrazu v reálném čase.

První část se zabývá problematikou detekce objektu v obraze. Cílem je zde seznámit čtenáře s aktuálně používanými metodami pro detekci lidí. Jelikož je však těchto metod mnoho a jejich přesný popis by byl velice obsáhlý, rozhodl jsem se vybrat pouze některé zástupce a stručně popsat jejich činnost. Pro informace o přesném průběhu zmíněných algoritmů je ale také vždy přiložen odkaz na originální publikaci, ze které byly informace čerpány.

Ve druhé části budou podrobně popsány metody, kterých výsledná aplikace využívá. Kapitola bude zaměřena spíše na popsání principu jednotlivých technik, než na popis jejich implementace.

Třetí část je věnována vysvětlením průběhu hlavního algoritmu. Znázorněn zde bude průběh algoritmu od prvotní detekce osob v obraze, přes jejich sledování až po závěrečné vyhodnocování zjištěné trajektorie. Kromě toho zde budou popsány i vlivy parametrů, které se zde vyskytují, na výsledek.

Čtvrtá část má za úkol popsat mnou implementovaný software. Na rozdíl od předešlých kapitol, zde bude popisná část zaměřena na implementaci jednotlivých technik do aplikace. Prostoru bude věnován popisu funkcionalit hlavních tříd včetně jejich vzájemné provázanosti.

Pátá část se zabývá testováním softwaru na zkušebních videosekvencích a vyhodnocování jeho úspěšnosti. Kromě ukázek situací, ve kterých pracovala aplikace správně, jsou zde i popsány situace, kde naopak program selhával.

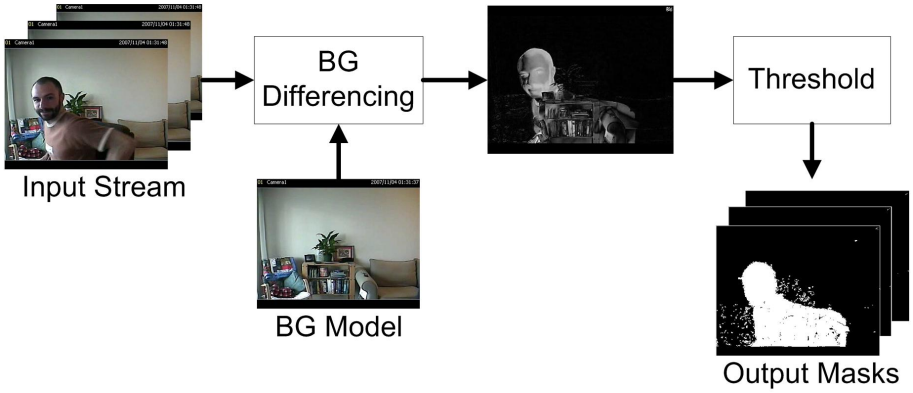
2 Aktuální techniky detekce

Předtím, než můžeme začít analyzovat trajektorii a chování osob z videosekvencí, musíme se nejprve zaměřit na základní problém každé podobné aplikace. Tímto základním problémem je, jakým způsobem budeme ze vstupního obrazu dostávat informaci o přítomnosti a pozici osoby (nebo osob). Tuto část bych proto věnoval právě problematice detekce osob. Pokusím se zde zaměřit na aktuální známé metody, které byly v odborné literatuře na toto téma publikovány. Nebudou zde popsány určitě všechny techniky, ale zaměřím se pouze na vybrané způsoby a pokusím se, kromě popisu jejich fungování, nastínit i jejich výhody a nevýhody. Určitě si nemůžeme dovolit prohlásit jeden přístup za nejlepší. Každý má totiž své přednosti, ale také hranice. Záleží totiž ve velké míře na prostředí ve kterém jsou tyto techniky používány. Pro vytvoření požadované aplikace, která je součásti této diplomové práce, bylo totiž zapotřebí seznámit se nejprve s možnými technikami a poté vyhodnotit, která z nich by v našem případě byla nejvíce vhodná.

2.1 Techniky využívající subtrakci pozadí

Tuto kapitolu bych chtěl věnovat široce známé a hojně používané skupině metod, které využívají techniku nazývanou jako subtrakce (odečítání) pozadí. Někdy ji však můžeme najít i pod označením detekce popředí. Tyto metody se používají pro detekci pohyblivých objektů (jako jsou například lidi, auta, atd.) a to výhradně za pomocí statických kamer. Celý proces detekce objektů bychom zde mohli zařadit do dvou základních úkonů.

Prvním z nich je **detekce pohybu**. V tomto kroku je cílem analyzovat vstupní obraz a oddělit pozadí, které většinou není předmětem našeho zájmu, od popředí, které naopak většinou obsahuje objekty důležité pro celkovou detekci. Základní myšlenka je zde postavena na tom, že se nejprve vytvoří model pozadí a poté se každý aktuální snímek od tohoto modelu pozadí odečítá. Rozdíl získaný touto operací potom můžeme považovat za popředí, neboli detekované pohybující se objekty. V této souvislosti se však ještě při odečítání snímků využívá jednoduchý práh, který by eliminoval velmi malé změny, které mohou být způsobeny šumem. Tudíž jestliže rozdíl jasů odpovídajících si pixelů přesáhne práh, je tato oblast určena jako popředí. V opačném případě je jejich rozdíl zanedbán. Tento proces je ilustrován na obrázku1. Pro model pozadí by přitom měla platit základní podmínka, a to, že model pozadí musí reprezentovat scénu neobsahující žádné pohybující se objekty a také, že musí být neustále aktualizován, aby se mohl adaptovat například na měnící se světelné podmínky. Jak jsem již zmínil, nejedná se zde pouze o jednu metodu, ale o celou skupinu přístupů, jejichž základní myšlenka, postavená na odečítání pozadí, je stejná. Liší se však v tom, jakým způsobem a s jakou přesností toho dosáhnout. V návaznosti na to můžeme také pozorovat rozdíly v rychlosti těchto algoritmů. Nicméně všechny tyto přístupy můžeme považovat za použitelné v reálném čase (real-time).



Obrázek 1: Diagram znázorňující získávání oblasti popředí pomocí modelu pozadí

Dalším bodem je **detekce objektů**. Metoda odečítání pozadí nám sice poskytne náhled na oblasti pohybu, ale nezaručí nám správnou segmentaci objektů. Při detekování jediného pohybujícího objektu ve scéně bychom možná dosáhli uspokojujících výsledků, ovšem při sledování scény se skupinami lidí, kteří se mohou překrývat, se naskýtá problém. Často totiž oblasti pohybu více objektů splynou do jediného regionu, který by mohl být lehce klasifikován jako jediný objekt. Proto existují metody, které se právě tímto úskalím zabývají. Podobně jako u detekce pohybu, i zde není určen přesný postup, jak kýženého cíle dosáhnout. Chtěl bych zde proto zmínit alespoň některé publikované techniky, které se považují za úspěšné. Každá z nich má své silné a slabé stránky, které se promítají jak do přesnosti vyhodnocování tak i na jeho rychlost. A proto výběr metody záleží vždy na okolnostech, za kterých se má detekce provádět.

2.1.1 Detekce pohybu

Jak již bylo řečeno v úvodu, myšlenka těchto metod je velmi jednoduchá. Při každém průchodu touto metodou máme k dispozici snímek, který zobrazuje aktuální dění ve scéně a také model pozadí. Tento model by měl vyjadřovat vzhled pozadí bez jakýchkoliv objektů, které bychom chtěli detekovat. V tomto bodě se však dostáváme k ústřednímu problému celého algoritmu, kterým je právě reprezentace pozadí. Jako základní rozdělení bychom proto mohli považovat přístup k modelu pozadí staticky nebo dynamicky.

Statický model

Myšlenka statického modelu pozadí by se dala považovat za velmi naivní přístup. Má sice mnoho nedostatků, ale je vhodné jej alespoň zmínit. Základním předpokladem pro tuto metodu je, že již známe model pozadí. Teoreticky bychom mohli využít faktu, že záznam provádíme na statické kameře a získání modelu pozadí bychom mohli zařídit manuální konfigurací. Pořídili bychom jednoduše jeden snímek, který by podle nás reprezentoval statické pozadí. Poté bychom pro každý aktuální snímek vytvořili rozdíl s tímto modelem pozadí. To znamená, že bychom pro každý pixel obrazu získávali abso-

lutní hodnotu rozdílu jasové složky aktuálního snímku a jasové složky modelu pozadí. Výsledek bychom poté mohli upravit prahováním, abychom eliminovali nízké hodnoty rozdílu, které by mohly být zapříčiněny například šumem. Jako výstup této operace bychom při použití prahování dostali obraz binární. Jestliže by prahování nebylo provedeno, získali bychom takzvaný pravděpodobnostní obraz, jehož hodnoty by byly v intervalu $[0, 1]$, a které by vyjadřovali pravděpodobnost, že daný pixel náleží popředí.

Jak již bylo řečeno, má tato metoda velké nedostatky. Mezi první by mohl patřit problém, jak získat snímek, nebo sekvenci snímků, která by reprezentovala model pozadí. Museli bychom nejprve scénu ručně nakonfigurovat, aby obsahovala skutečně pouze to, co je podle našeho uvážení statické pozadí. U některých scén je tato podmínka často nesplnitelná. Stačí totiž, aby byl jeden objekt z pozadí odebrán a při získávání rozdílu snímků by nám na tomto místě objevil jakýsi obtisk chybějícího objektu. Ukázku tohoto jevu můžete vidět na obrázcích 2. Dále bychom ještě mohli uvést, že by tento přístup selhával už při pouhé změně osvětlení scény. To by totiž mělo za následek, že i pixely, které by reálně patřily statickému pozadí, měly najednou jinou hodnotu jasu. Při porovnání se statickým modelem pozadí by pak byly tyto pixely, z důvodu velkého rozdílu jasu, označeny jako pixely náležící popředí, což by neodpovídalo realitě. Celkově můžeme prohlásit, že tento princip zcela zanedbává adaptaci modelu pozadí měnícím se podmínkám scény, což je jeho kámen úrazu.

(a) Statický model pozadí (b) Aktuální snímek (c) Výstup po detekci pohybu

Obrázek 2: Ukázka selhání statického modelu pozadí, kde je zaznamenána jak aktuální pozice objektu, tak i obtisk jeho minulé pozice.

Dynamický model

V tomto ohledu je myšlenka dynamického modelu pozadí daleko pokročilejší. I zde budeme sice jako základní operaci vypočítávat rozdíl aktuálního snímku a modelu pozadí, avšak model pozadí bude adaptivní. Ve skutečnosti to znamená, že bude při běhu probíhat i jeho neustálé učení. Tento přístup nám umožní vyhnout se problémům například se změnou osvětlení, při kterém měla myšlenka statického modelu problémy. V základě

se zde pracuje s předpokladem, že pozadí je viditelné mnohem častěji, než objekty na popředí. Proto bychom v jednoduchém verzi mohli využít vztahu

$$B_{i+1} = \alpha F_i + (1 - \alpha) B_i \tag{1}$$

který je známý pod označením exponenciální klouzavý průměr. Kde F_i je aktuální snímek, B_i je aktuální model pozadí, α je učící parametr (obvykle má hodnotu v řádu setin) a B_{i+1} je aktualizovaný model pozadí. Jiné metody jsou v tomto ohledu ještě pokročilejší.

Metoda podle Wrena a spol.

Například v roce 1997 Wrent a spol. publikovali metodu [1], která pro rozpoznávání pozadí a učení využívala normální rozdělení. Představme si, že pracujeme s obrazem pouze s jedním kanálem (obraz ve stupních šedé). Ke každému pixelu obrazu je zde přiřazen jeden Gaussián (normální rozdělení), který představuje rozložení hustoty pravděpodobností pro hodnoty jasové složky. Pro barevné obrazy bychom samozřejmě pro každý kanál určili samostatný Gaussián. Díky tomu můžeme celý histogram uchovávat pomocí parametrů pro definici Gaussiánu. Učení pak probíhá za pomocí vztahů23, kde μ_t je střední hodnota a ρ_t^2 vyjadřuje rozptyl Gaussiánu v čase t. Dále pak F_t vyjadřuje jasovou hodnotu pixelu aktuálního snímku a α je učící parametr. Rozhodování o tom, jestli daný pixel náleží popředí může být vyhodnocováno podle předem určeného prahu, nebo podle vztahu4, kde k je nějaká vhodně zvolená konstanta (v některých publikacích se uvádí optimum $k = 2.5$).

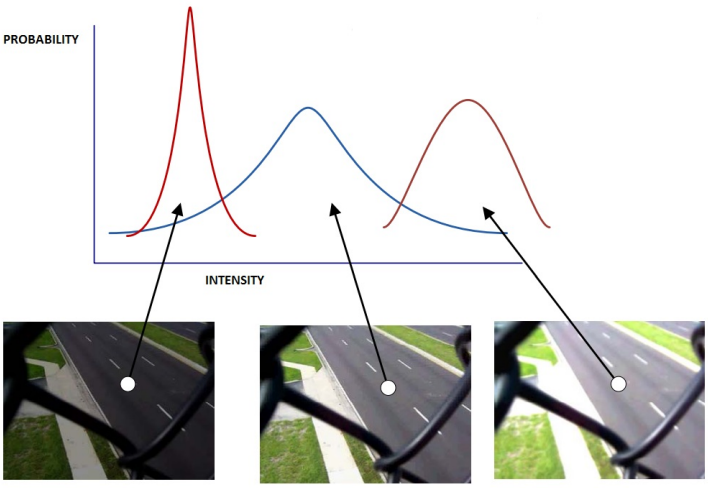
$$\mu_{t+1} = \alpha F_{t+1} + (1 - \alpha)\mu_t \tag{2}$$

$$\rho_{t+1}^2 = \alpha(F_{t+1} - \mu_{t+1})^2 + (1 - \alpha)\rho_t^2 \tag{3}$$

$$|F - \mu| > k\rho \tag{4}$$

Metoda podle Stauffera-Grimsona

V roce 1999 byl představen algoritmus Stauffera-Grimsona [2], který posunul adaptivitu modelu pozadí ještě dále. Jeden pixel totiž už není reprezentován pouze jedním Gaussiánem, ale směsí několika Gaussiánů. Ilustraci znázorňující toto chování můžete sladovat na obrázku 3. V některých scénách může totiž docházet k velkým a rychlým změnám pozadí, kterým se model pozadí nedokáže rychle přizpůsobit. Typickým příkladem může být venkovní scéna s budovou, která je částečně překrytá stromy. Jeden pixel tak může během krátké doby mít hodnoty jasu budovy, listů nebo větví stromu. Jako dalším příkladem může být déšť, sněžení nebo vodní hladina. Pixely v těchto případech sice budou rychle měnit své hodnoty, ale většinou budou nabývat pouze hodnoty několika opakujících se stavů. Jeden Gaussián by byl v tomto případě velice nevhodný a tudíž se používá



Obrázek 3: Snímek znázorňující model pozadí adaptovaný pomocí tří Gaussiánů na tři různé světelné intenzity

směs několika Gaussiánů, aby právě vyjádřily více stavů, kterých může pozadí nabývat. Každý Gaussián i má proto kromě definované střední hodnoty μ_i a rozptylu ρ_i^2 také přiřazenu váhu ω_i. Pravděpodobnostní náležení pixelu, který by byl definován složkami RGB, do pozadí by potom bylo definováno vztahem 5;. Kde n vyjadřuje zvolený počet Gaussiánů a pro funkci $\eta([r,g,b],\mu_i,\rho_i^2)$ můžeme definovat vztah 6;.

$$P([r,g,b]) = \sum_{i=1}^{n} \omega_i \eta([r,g,b],\mu_i,\rho_i^2) \tag{5}$$

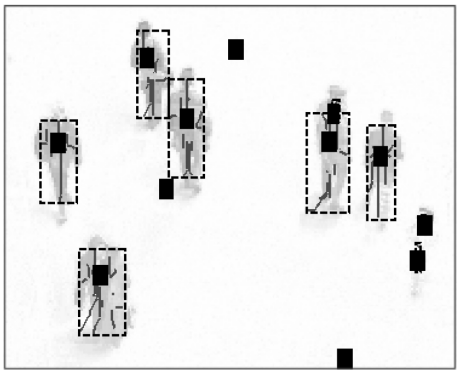
$$\eta([r,g,b],\mu_i,\rho_i^2) = \frac{1}{(2\pi\rho^2)^{\frac{3}{2}}} e^{\frac{(r-\mu_r)^2+(g-\mu_g)^2+(b-\mu_b)^2}{2\rho^2}} \tag{6}$$

2.1.2 Detekce objektů

V předešlých kapitolách jsme se věnovali algoritmům, pomocí kterých bychom byli schopni rozeznat popředí scény, které má pro nás velký význam, od pozadí. V dalším kroku zpracování popředí, které má obsahovat jednotlivé pohybující se objekty, by se nabízel velmi primitivní způsob segmentace objektů. V prostoru zobrazující popředí bychom jednoduše našli spojité oblasti a každou z nich bychom prohlásili za jeden objekt. V praxi bychom se ale setkali s problémy už při situaci, kdy bychom sledovali chodce ve velkém shluku. Algoritmus by je automaticky prohlásil za jeden objekt, což by bylo určitě chybné. Proto byly publikovány některé techniky, které při řešení tohoto problému dosahují dobrých výsledků. Právě těmto metodám bych chtěl věnovat prostor v následující sekci. Jelikož je však skupina těchto technik opravdu velká, zmíním zde pouze vybrané algoritmy.

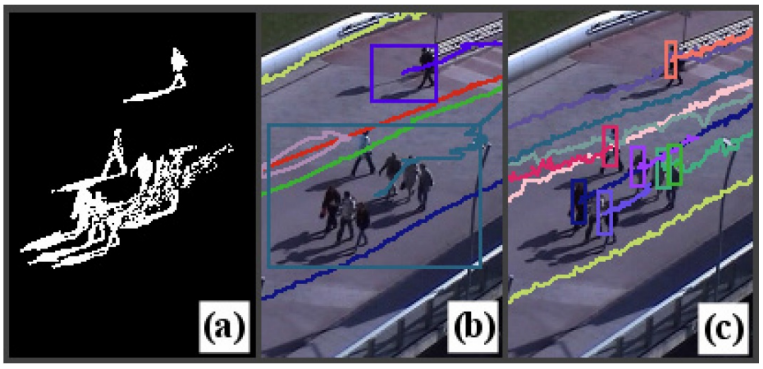
Metoda podle Beleznaie a spol.

Jako jeden ze známých algoritmů bych zde chtěl zmínit práci Beleznaie a spol., která byla publikována v roce 2006. V první fázi zpracování obrazu se zde využívá již dříve popsaná metoda subtrakce pozadí využívající směs Gaussiánů. Po provedení odečtení aktuálního snímku od modelu pozadí dostaneme takzvaný rozdílový snímek, který je základem pro další analýzu. Ve druhé fázi už se budeme zabývat hledáním objektů zájmu. Prvním krokem je nalezení výrazných lokálních maxim v tomto rozdílovém snímku. Výsledkem je množina bodů nacházející se v oblastech pohybu. Zde je poté využita procedura nazývaná Mean Shift (přesněji její rychlejší varianta zvaná Fast Mean Shift). Hlavním účelem této metody je najít pro určité okolí pozici s nejvyšší hustotou nadefinovaných bodů. V našem případě tyto nadefinované body představují již dříve nalezená lokální maxima. Opakované využití této metody, kdy jako počáteční bod volíme jedno z vyhledaných maxim, pak vede k tomu, že se nalezne jakési centrum objektu. Body, které k tomuto centru konvergují, představují oblasti patřící jednomu objektu. Snímek znázorňující tento proces lze vidět na 4.



Obrázek 4: Příklad procesu Mean Shift na rozdílovém snímku. Výrazné černé oblasti určují centra objektu. Přerušovanou čarou jsou vyznačeny hranice oblastí nalezených objektů. Také jsou zde vyznačeny cesty, které představují průběh metody Mean Shift, neboli cestu při hledání největšího shluku.

V jistých případech může vzniknout situace, kdy dva objekty i po tomto postupu splynou v jeden. Pro tyto případy je poté aplikováno vyhodnocování, které určí nejpravděpodobnější rozdělení na požadované objekty. Konečnou fázi detekce tvoří klasifikace každého nalezeného objektu, kdy se určí, zdali jde například o člověka. Ta je provedena jednoduchým vyhodnocením výšky a šířky objektu. Porovnání výsledků tohoto algoritmu s výsledkem jednoduché segmentace podle spojitých oblastí můžete vidět na 5 Poněvadž by byl podrobný popis tohoto algoritmu příliš obsáhlý, je zde nastíněn pouze obecný postup. Přesnější informace však jsou prezentovány v originální publikaci [3].



Obrázek 5: Ukázka úspěšnosti algoritmu. V části (a) lze vidět obraz se zvýrazněnými oblastmi popředí. V části (b) je výstup pro jednoduchou segmentaci. Pro porovnání úspěšnosti je v části (c) znázorněn výstup po použití algoritmu podle Beleznaie a spol.

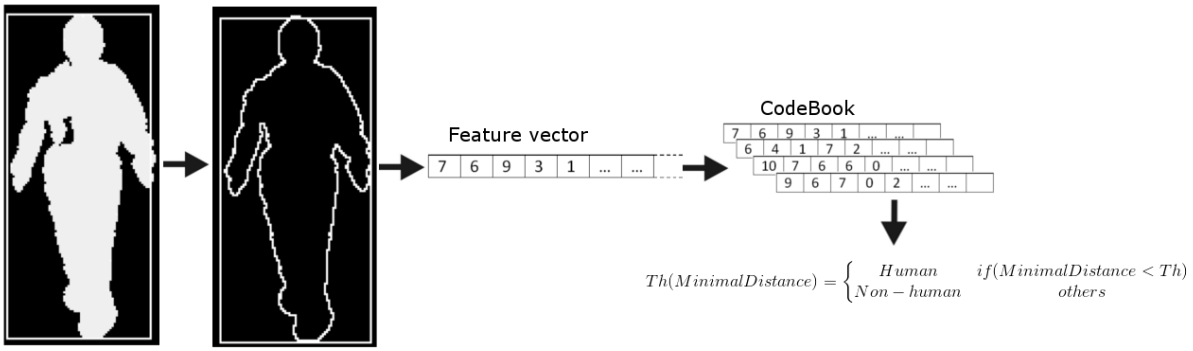
Metoda podle Zhoa a Hoanga

Jako další příklad jsem vybral metodu, která byla publikována v roce 2004 a jejíž autory jsou Zhou a Hoang. Jak jsem již dříve zmínil, slabou stránkou metody subtrakce pozadí je, že se špatně adaptuje na změny prostředí, což jsou například náhlé změny světelných podmínek, výrazné stíny nebo pohyb objektů, které patří do pozadí. Právě na správné vyhodnocování těchto situací se zmíněná technika soustřeďuje.

Jak by se již dalo předpokládat, prvním krokem algoritmu je subtrakce pozadí. Zde se využívá klasického postupu modelování pozadí pomocí směsi Gaussiánů. Pro získání diference aktuálního snímku a modelu pozadí zde ovšem není použit jednoduchý výpočet rozdílu příslušných pixelů pozadí a aktuálního snímku. Ten totiž neumí reagovat na nechtěné jevy vzniklé pohybem stromů nebo vlněním vodní hladiny. Místo toho je zde využit statistický model, který nezohledňuje vždy pouze jeden pixel snímku a pozadí, ale také jejich blízké okolí. Tím se zajistí, že jevy, jakou je například pohyb stromů, nebudou do rozdílového snímku zaznamenány.

Pro detekci stínů, které jsou ve výsledku nežádoucí je zde následně použit algoritmus využívající faktu, že stíny jsou semi-transparetní. Barevný model RGB je pro toto ohodnocování nežádoucí a proto je pixel převeden do barevného složení YUV, kde máme oddělenu jasovou složku pixelu (Y) od barevných složek (UV). Zde už můžeme jednodušeji vyhodnotit barevný rozdíl a určit tak, zda se jedná skutečně o objekt popředí, nebo jen pouhý stín.

Po těchto krocích už následuje nejpodstatnější úkol této techniky, což je identifikace osob na snímcích. K tomu je zde použita technika využívající strukturu CodeBook. Pro každou plochu, detekovanou pomocí minulých kroků jako popředí, se provede následující algoritmus. V prvním kroku se takovýto objekt normalizuje do velikosti 20x40. Z



Obrázek 6: Proces znázorňující průběh klasifikace pomocí metody CodeBook

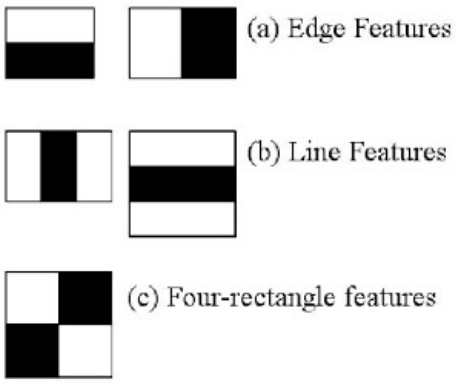
takto modifikovaného objektu se pomocí určitých pravidel vypočítá příznakový vektor. Ve zmíněné publikaci představují složky tohoto vektoru vzdálenosti 20ti vybraných bodů na hranici objektu od ohraničujícího obdélníku. Dimenze výsledného příznakového vektoru je tudíž 20. Vzniklý příznakový vektor se poté porovnává s vektory uloženými právě ve zmíněné struktuře CodeBook. Jestliže CodeBook obsahuje vektor, jež by byl shodný, nebo alespoň velmi podobný aktuálnímu vektoru příznaků, je vyhodnocovaný tvar určen jako člověk. V opačném případě se samozřejmě posoudí, že se o člověka nejedná. Schéma průběhu tohoto vyhodnocování je znázorněn na 6. Intuitivně ale musíme dodat, že při takovéto technice musí nejprve proběhnout proces učení, při kterém by CodeBook získal pozitivní ukázky vektoru příznaků, které by následně sloužily k vyhodnocování. Dodatečné korekce pro identifikování lidské postavy se provádí během procesu sledování objektu. Podrobný popis tohoto algoritmu lze dohledat v originální publikaci [4]. Zde nám jde v první řadě o nastínění hlavního principu a fungování této techniky. Bližší rozbor by byl pro tuto práci příliš rozsáhlý.

2.2 Přímá detekce

Druhou kategorii metod bychom mohli nazvat jako techniky využívající přímou detekci. Na rozdíl od výše zmíněných metod vyžadujících v prvním kroku subtrakci pozadí, metody přímé detekce vyhodnocují celý aktuální snímek a podle něho určují přítomnost lidské postavy. Obecně se snaží nejprve obraz popsat pomocí příznaků, což mohou být kupříkladu obrysy (pro vyhodnocení tvaru) nebo barvy (pro detekování kůže). Na základě těchto postupů se poté získává informace, zda je ve snímku detekována postava či nikoliv. Velká výhoda tkví v tom, že jelikož tyto techniky pracují ve většině případech jen s jedním konkrétním snímkem, odpadá tak zde nutnost používat pouze záznamy ze statických kamer.

2.2.1 Metoda autorů Viola, Jones, Snow

Mezi nejznámější představitele technik přímé detekce musíme určitě zařadit algoritmus publikovaný v roce 2003 Violou a spol. [5]. Šlo prakticky o rozšíření algoritmu pro rozpoznávaní obličeje, který publikovali titíž autoři v roce 2001. Jde o první zveřejněný algoritmus, kombinující přístupy detekce člověka ze statických snímků a také detekce modelu pohybu, takže určitě stojí za zmínění. Detektor je v podstatě postaven na mnoha jednoduchých obdélníkových filtrech. Ty se nacházejí vždy uvnitř klouzavého okénka, které prochází postupně celý obraz, a jejich výstup udává rozdíl intenzit dvou specifikovaných oblastí. Ukázky takových filtrů můžete vidět na obrázku 7.



Obrázek 7: Ukázka filtrů používaných u metody podle autorů Viola, Jones, Snow.

Komplexnost klasifikátoru je zajištěna následně metodou AdaBoost. Její myšlenka stojí na tom, že pomocí mnoha jednoduchých klasifikátorů, které jsou tvořeny právě zmíněnými obdélníkovými filtry, je sestaven jeden výkonný. Při každé iteraci se v podstatě ověřuje každý jednoduchý filtr, ke kterému se poté určí nejideálnější práh (threshold). Z takovýchto kombinací filtr/práh se poté vybere nejideálnější instance, jenž se zařadí do výsledného klasifikátoru. Takto se pokračuje, dokud nedosáhneme dostatečné přesnosti. Ověřování tohoto komplexního klasifikátoru je ale velmi časově náročná operace. Proto se pro urychlení detekce využívá metoda kaskádových klasifikátorů. Jde v podstatě o to, že každá ověřovaná oblast se nejprve ověří pomocí jednoduchého klasifikátoru. Jestliže už zde nastane rozpor, oblast se ihned vyhodnotí jako nevyhovující. V opačném případě oblast postupuje do další úrovně, kde je ověřována postupně složitějšími klasifikátory. Jelikož se většinou značná část oblastí zamítne již v prvních úrovních kaskády, kde se nacházejí jednoduché klasifikátory, ušetří se tak mnoho výpočetního výkonu. Tímto postupem je řešena detekce statických obrazů.

Jelikož však chceme do celkové detekce zachytit i model pohybu, musíme rozšířit množinu snímků, se kterými budeme pracovat. Pro jejich získání je použita vždy dvojice po sobě jdoucích snímků. Kromě aktuálního snímku I tedy budeme dále pracovat s δ, U, L, R, D jejichž vznik je dosažen pomocí následujících předpisů:

$$\begin{aligned}
\delta &= abs(I_t - I_{t+1}) \\
U &= abs(I_t - I_{t+1} \uparrow) \\
D &= abs(I_t - I_{t+1} \downarrow) \\
L &= abs(I_t - I_{t+1} \leftarrow) \\
R &= abs(I_t - I_{t+1} \rightarrow)
\end{aligned} \tag{7}$$

kde I_t a I_{t+1} jsou po sobě následující snímky a $\uparrow, \downarrow, \leftarrow, \rightarrow$ jsou translační operace. Například označení $I_t \uparrow$ představuje obraz I_t posunutý nahoru o hodnotu jednoho pixelu. Na takto získané snímky se poté může provést stejný postup, který je popsán výše. V některých případech ale může pro získání informaci o pohybu stačit jednoduchý filtr, který má předpis:

$$f_i = r_i(\delta) - r_i(S) \tag{8}$$

kde S je jeden z obrazů U, L, R, D a $r_i()$ vyjadřuje funkci pro součet jasů v detekčním okně. Pomocí porovnávání jednotlivých výsledků pro U, L, R, D, pak může být určen pravděpodobný směr pohybu. Díky tomu, že snímky U, L, R, D obsahují ve většině případech velké množství pixelů s nulovou intenzitou, můžou být výpočty nad nimi prováděny s relativně velkou rychlostí. Nakonec je třeba ještě zdůraznit, že pro správnou funkci detektoru je nejprve potřeba pořídit množství trénovacího materiálů, pomocí kterého by proběhlo jeho učení.

2.2.2 Metoda podle Dalala a Triggse

Na závěr tohoto výběru detekčních technik bych chtěl ještě zmínit práci, která byla veřejnosti představena roku 2005 a jejíž autoři jsou Navneet Dalal a Bill Triggs [6]. Jejím základním prvkem je, že pro tvorbu klasifikátoru je zde použit příznakový prostor tvořený histogramy orientovaných gradientů. V podstatě se zde využívá faktu, že tvar sledovaného objektu lze dobře popsat pomocí rozložení intenzit gradientů nebo směrů hran. Každý snímek je proto nejprve rozdělen do mnoha malých buněk (cells). Pomocí pixelů uvnitř každé takovéto buňky je získán histogram orientovaných gradientů. Pro klasifikaci objektů je následně použit dataset pozitivních (obsahující člověka) a negativních (neobsahující člověka) vzorků. Pomocí těchto dat poté probíhá učení lineárního klasifikátoru metodou SVM. Jelikož tento detektor dosahuje velmi dobrých výsledků, stal se v těchto letech velmi populárním. Na principu rozpoznávání objektů pomocí histogramu orientovaných gradientů vznikly ještě další metody, které tento přístup rozšiřují. To naznačuje, že by právě tímto směrem mohl jít i další vývoj technik pro detekování. Právě z tohoto důvodu jsem se rozhodl, že aplikace, která bude výsledkem této diplomové práce, bude využívat právě tohoto postupu detekce chodců. Proto zde již nebudu dále popisovat průběh algoritmu, jelikož ten bude podrobně popsán v dalších částech této práce.

3 Použité metody

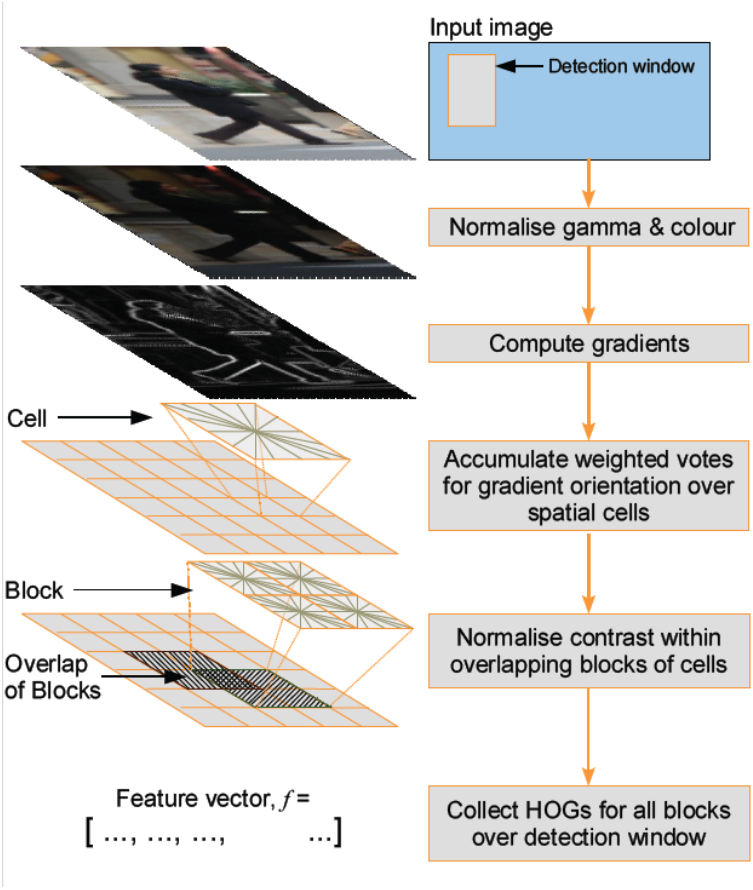
V této kapitole popíši metody, které jsou ve výsledné aplikaci využívány. Zaměřím se spíše na teoretický popis fungování jednotlivých technik, než na jejich konkrétní využití v aplikaci, ta bude totiž podrobně zmíněna v kapitole následující.

Jak již bylo naznačeno dříve, pro účely detekce osob v obraze bylo využito přímé detekce pomocí detektoru HOG. Jako klasifikátor je zde použita metoda SVM. Tento algoritmus je prováděn pro každý jednotlivý snímek celé sekvence a jeho výstupem je množina detekčních oken, signalizujících pozici a velikost nalezené osoby.

Pro sledování osob je zde následně použit algoritmus, jehož základem je funkce Kalmanova filtru. Díky jeho schopnosti predikovat pravděpodobné budoucí stavy zvýšíme přesnost sledování objektu, jejichž přesná detekce se v aktuálním snímku nezdařila. Tento problém může být způsoben například vlivem přecházení osoby kolem překážky, nebo nenadálým selháním HOG detektoru.

3.1 HOG detektor

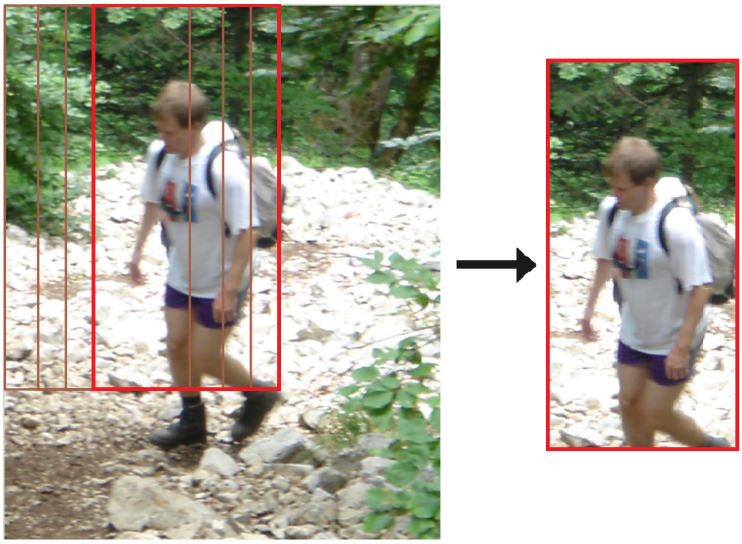
Jak již zkratka naznačuje, jde o příznakový detektor používající pro svou funkci histogramy orientovaných gradientů. Hlavní myšlenka je zde postavena na tom, že vzhled a tvar objektu mohou být často dobře charakterizovány pomocí rozložení směru hran nebo intenzit lokálních gradientů. Tento detektor byl původně vyvinut pro detekci osob, ale lze jej využít i pro detekci jiných objektů. Za zmínku určitě stojí i fakt, že autoři ve své publikaci [6] zveřejnili i srovnání jejich HOG detektoru s detektory využívající Haarovy vlnky, PCA-SIFT a tvarem orientované detektory. Ve výsledcích je poté zřetelně viditelné, že na testovacích datech dosahuje jejich technika zdaleka nejlepších výsledků. I to je jeden z faktů, proč byl tento přístup zvolen pro výslednou aplikaci. Princip detektoru je založen na posloupnosti několika kroků 8, které budou podrobně popsány níže.



Obrázek 8: Posloupnost kroků prováděna HOG detektorem pro extrakci příznakového vektoru z každého detekčního okna.

Detekční okno

Nejprve je třeba zdůraznit, že HOG detektor jako takový ve své podstatě nepracuje s celým vstupním obrazem najednou, ale vždy počítá jen s jeho částí, která se nazývá detekční okno. To je posouváno napříč celým vstupním obrazem a určuje regiony obrazu, u nichž se bude následně vyhodnocovat, zda obsahují osobu nebo ne. Jelikož ale nemůžeme zajistit, že osoba bude mít vždy jen jednu přesnou velikost, musíme kromě pozice okna měnit i jeho měřítko. Každé takto získané detekční okno je pak předáno jako vstup pro další zpracování. Tento přístup není ojedinělý pouze pro HOG detektor, ale je používán i pro celou řadu jiných technik pro detekci.



Obrázek 9: Vlevo je znázorněn průchod detekčního okna obrazem. Kromě pozice okna se však může měnit i jeho velikost. Vpravo je ukázka extrahované části obrazu, která se následně používá pro další zpracování a vyhodnocování.

Normalizace jasu

Prvním krokem prováděným pro vstupní podobraz definovaný detekčním oknem je, normalizace jasu. K tomuto účelu se využívá gama korekce, jejíž vztah je:

$$O(x, y) = I(x, y)^{\gamma} \tag{9}$$

$I(x, y)$ definuje vstupní a $O(x, y)$ výstupní obraz. Vliv parametru γ lze vidět na následující ilustraci 10.

Obrázek 10: Názorná ukázka vlivu nastavení parametru γ , jež se používá při korekci jasu, na výsledný snímek. Zde jsou pro γ voleny zleva tyto hodnoty: 0.5, 0.75, 1, 1.5, 2.

Díky tomu můžeme snížit vliv efektu způsobeného změnami světelných podmínek. Tento krok je ale často považován jako nepovinný. Může být přeskočen, jelikož se ukázalo, že při výpočtech používajících gradienty není vliv normalizace nijak silný.

Výpočet gradientů

Dalším krokem algoritmu je výpočet gradientů. Gradient je obecně vektor, jež definuje velikost a směr největšího růstu funkce v určitém bodě. Pomocí těchto gradientů tak můžeme lehce popsat i velikost a směr obrazové hrany, což nám usnadní pozdější detekci. Pro výpočet však nejprve musíme v našem obraze vypočítat parciální derivace ve směrech osy x a y, čehož u našeho diskrétního obrazu můžeme docílit pomocí následujících vztahů: 10; 11;

$$\frac{\partial F(x,y)}{\partial x} \approx \frac{F(x+1,y) - F(x-1,y)}{2} \tag{10}$$

$$\frac{\partial F(x,y)}{\partial y} \approx \frac{F(x,y+1) - F(x,y-1)}{2} \tag{11}$$

kde $\frac{\partial F(x,y)}{\partial x}$ udává parciální derivaci ve směru osy x, $\frac{\partial F(x,y)}{\partial y}$ parciální derivaci ve směru osy y a $F(x,y)$ je funkce určující intenzitu pixelů na daných souřadnicích. Místo těchto předpisů se však můžeme často setkat i s výpočty parciálních derivací podle konvolučních masek $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ a $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$, které jsou založeny na výše zmíněných matematických předpisech. Z takto zjištěných parciálních derivací můžeme následně již jednoduše vypočítat velikost a směr gradientu. Kde velikost je dána předpisem:

$$g(x,y) = \sqrt{\Big(\frac{\partial F(x,y)}{\partial y}\Big)^2 + \Big(\frac{\partial F(x,y)}{\partial x}\Big)^2} \tag{12}$$

a jeho směr vyjadřuje vztah:

$$\theta(x,y) = tan^{-1}\Big(\frac{\partial F(x,y)}{\partial y} \Big/ \frac{\partial F(x,y)}{\partial x}\Big) \tag{13}$$

Na obrázcích 11 jsou znázorněny konkrétní výsledky výpočtů pro ukázkový snímek o rozměrech 128x64 pixelů, kde hodnota výsledku je znázorněna hodnotou jasu.

Stanovení váhovaného histogramu gradientů

V této fázi autoři definují rozdělení obrazu do velmi malých regionů nazývaných buňky (cells) viz 12. Tyto oblasti mohou mít rozměry 6x6 pixelů, což bylo autory označeno jako ideální rozměr. V mnoha implementacích se však setkáme i s rozměry 8x8 pixelů, u kterého se dosahuje podobně dobré úspěšnosti. Z výsledků minulého kroku již máme pro každý pixel obrazu vypočítány gradienty určující velikost a směr hrany. Nyní pro

Obrázek 11: a)Původní snímek ve stupních šedi. b)Absolutní hodnoty derivací ve vertikálním směru. c)Absolutní hodnoty derivací v horizontálním směru. d)Výsledná velikost gradientu.

všechny gradienty uvnitř každé buňky stanovíme histogram orientovaných gradientů. Takže každá buňka bude tvořena jedním histogramem sestaveným z 64 gradientů (pro velikost buňky 8x8). Aby bylo množství informací co nejlépe komprimováno, histogram je tvořen 9 zásobníky (bins), které rozdělují rovnoměrně úhel 0°- 180°na jednotlivé úseky. Autoři ve své práci úmyslně využili takzvaného "unsigned"rozsahu 0°- 180°, ale lze samozřejmě využít i "signed"rozsahu 0°- 360°. Taktéž lze použít i jiného počtu zásobníků, ale experimentálně bylo zjištěno, že 9 zásobníků dosahuje nejideálnějších výsledků.

Samotné sestavení histogramu probíhá tak, že každý gradient podle svého směru přispívá poměrově dvěma nejbližším zásobníkům. Velikost příspěvku je dána jak velikostí vektoru, kdy větší vektory mají samozřejmě větší příspěvek, tak právě i jeho směrem. Jestliže máme například vektor se směrem 85°, $3/4$ jeho velikosti přispívá zásobníku pro 90°a zbylá $1/4$ pak přispívá zásobníku pro 70°. Tento postup má za úkol minimalizovat problém gradientů, jejichž směry se nacházejí blízko hranice dvou zásobníků. Kdybychom totiž počítali příspěvek vždy jen jednomu zásobníku, mohlo by se lehce stát, že už při minimální změně směru gradientu by došlo k velké změně celého histogramu, což by určitě neodráželo skutečnost.

Tento proces shlukování gradientů do jednoho histogramu nám kromě komprimace informací nabízí i jistou variabilitu. Jestliže například mírně změníme směry některých gradientů, nebude mít tato deformace vy výsledném histogramu takovou váhu. U změn pozice je připuštěna ještě větší volnost. Histogram totiž neuchovává informace o pozicích jednotlivých gradientů, ale vyjadřuje pouze jejich rozložení v dané buňce. Důsledkem toho může být hrana určená gradientem posunuta v rámci jedné buňky a výsledný histogram zůstane zcela zachován.

Normalizace kontrastu uvnitř bloku

Další částí algoritmu je normalizace kontrastu. Ta je prováděna normalizací histogramu orientovaných gradientů, který byl získán v předešlém kroku. Nejprve však ukažme, jaký efekt tento krok v důsledků má. Představme si nejprve neupravený snímek, ze kterého budeme počítat jednotlivé gradienty. Jestliže nyní upravíme snímek tím, že zvýšíme hodoty jasu o určitou konstantní hodnotu, velikost gradientu, kterou dostaneme pomocí vztahů 10 a 12, se v tomto případě nijak nezmění. Jestliže však jasy v původním obraze vynásobíme určitou nenulovou hodnotou k, můžeme za pomocí dřívějšího postupu dostat gradienty, jejichž velikosti jsou větší či menší (v závislosti na zvolené hodnotě k) než velikosti gradientů původního obrazu. Tento efekt se následně promítne i do výsledného histogramu gradientů tím, že hodnoty v jednotlivých zásobnících budou k-násobkem hodnot původního histogramu.

Jestliže však provedeme normalizaci histogramu, takže hodnotu v každém zásobníku vydělíme součtem hodnot ve všech zásobnících histogramu, dostaneme pro původní i upravený snímek zcela stejné výsledky. Postup normalizace nám tudíž zajistí odolnost výsledného detektoru vůči zmíněnému typu změn osvětlení.

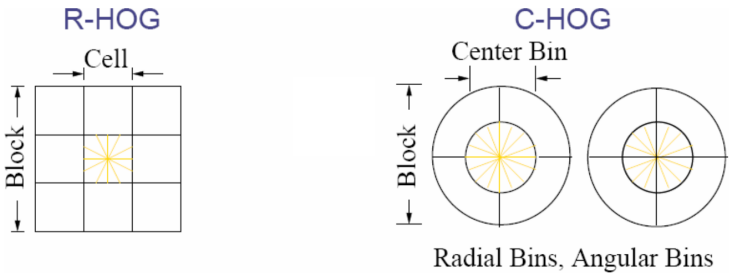
Autoři však neprovádí normalizaci pouze na jednotlivých histogramech. Aby zohlednili větší okolí, jsou nejprve buňky seskupeny do prostorově větších oblastí nazývaných bloky (blocks). Normalizace se následně provádí na základě všech histogramů nacházejících se uvnitř každého bloku. Jako nejideálnější velikost bloků autoři uvádí 3x3 buňky, ale ve většině implementací se můžeme setkat s doporučenou velikostí 2x2, která také dosahuje srovnatelných výsledků. Dále je třeba ještě zmínit, že na rozdíl od buněk, bloky mají tu vlastnost, že se mohou vzájemně překrývat. Pro bloky velikosti 2x2 se proto používá překrývání o 50%, které je znázorněno na následující ilustraci 12.

Obrázek 12: Ukázka seskupování jednotlivých buněk (cells) do větších bloků (blocks). Pro bloky je zde použito překrývání 50%. Velikost buňky je zde určena jako 8x8 pixelů a velikost každého bloku pak 2x2 buňky.

Důsledkem tohoto vzájemného překrývání bloků je, že se každá buňka může ve výsledném vektoru příznaků vyskytovat vícekrát. Přesněji řečeno rohové buňky se vyskytnou pouze jednou, ostatní buňky nacházející se na okraji snímku se ve výsledku budou vyskytovat dvakrát a nakonec všechny neokrajové buňky budou v deskriptoru obsaženy

celkem čtyřikrát. Rozdílem však bude, že pokaždé bude buňka normalizována za pomocí jiného bloku.

Obecně existují dva různé tvary bloku s nimiž autoři experimentovali. Prvním z nich je takzvaný pravoúhlý (R-HOG) blok, jež byl popisován výše. Pro něj se definují 3 základní parametry: počet buněk pro blok, počet pixelů pro buňku a počet zásobníků pro histogram buňky. Dalším tvarem je kruhový (C-HOG) blok. Jak již název napovídá, buňky zde nejsou čtvercové, ale jsou uspořádány do kruhu. Parametry v tomto případě představují čtyři parametry: počet úhlových a radiálních zásobníků, poloměr centrálního zásobníku a také expanzní faktor pro následující poloměry. Pro tento typ tvaru jsou poté ještě definovány dvě různé varianty. Pro první je definován pouze jeden centrální zásobník. Druhý tip se odlišuje tím, že je centrální zásobník rozdělen podle určených úhlů. Schémata bloků můžete vidět na ilustraci 13. Je však třeba zmínit, že ve většině implementací se využívá jednoduchého čtvercového R-HOG bloku.



Obrázek 13: Schémata jednotlivých tipů bloků. Vlevo lze vidět čtvercový R-HOG blok o velikost 3x3 buňky. Vpravo jsou pak znázorněny dvě varianty kruhových C-HOG bloků. Ty jsou vytvořeny se dvěma radiálními (centrální a okrajové) a čtyřmi úhlovými zásobníky.

Sestavení vektoru příznaků

Posledním krokem je, sestavení konečného vektoru příznaků pro celé detekční okno a jeho následné vyhodnocení. Jak již bylo zmíněno v předchozích krocích, výsledný příznakový vektor je stanoven pomocí dat všech histogramů nacházejících se uvnitř každého bloku detekčního okna.

Velikost takovéhoto vektoru příznaků je silně závislá na zvolených parametrech a lze ji poměrně snadno vypočítat. Například použijeme-li standardní hodnoty, kdy:

- velikost detekčního okna je 64x128 pixelů
- velikost buňky je 8x8 pixelů a histogram je tvořen devíti zásobníky
- každý blok je čtvercový o velikosti 2x2 buňky s překrýváním 50%

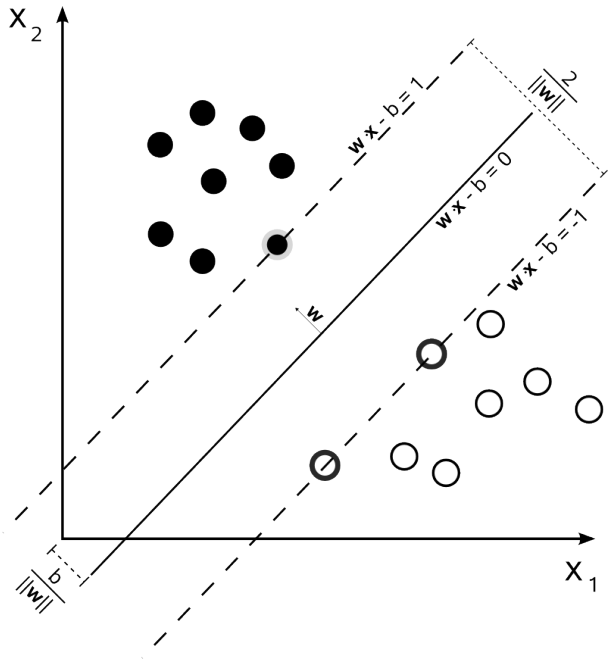
bude mít příznakový vektor popisující každé detekční okno přesně 3780 hodnot. Všechny hodnoty jsou následně předány klasifikátoru, jenž podle těchto informací vyhodnotí, zda se v detekčním osobě nachází osoba, nebo ne. Autoři pro tuto úlohu využili jednoduchý lineární klasifikátor SVM. Popisu této techniky bude věnován prostor v následující části.

3.2 SVM klasifikátor

Support vector machines (SVM) je metoda strojového učení. Obecným cílem této techniky je vyhledat v prostoru příznaků takovou nadrovinu, která by dokázala optimálně rozdělit trénovací data. Tím je myšleno, že body reprezentující stejnou třídu leží ve stejném poloprostoru. Navíc však chceme docílit i toho, že minimální vzdálenosti jednotlivých bodů trénovací množiny od dělící roviny jsou co největší, což obecně vede ke zlepšení klasifikátoru. Tato metoda je ve své základní podobě navržena pro účel binárního dělení, neboli separaci dvou různých tříd. V našem případě nám tento přístup postačí, protože ve výsledku chceme pouze oddělit negativní záznamy (neobsahující lidskou postavu) od těch pozitivních (obsahující člověka).

Lineární SVM

Základní úlohou této metody je nalezení optimální nadroviny pro lineárně separovatelná data jak je znázorněno na obrázku 14.



Obrázek 14: Schéma popisující definici optimální nadroviny pro lineárně separovatelná data. V tomto případě je zde zobrazen dvojrozměrný prostor příznaků x_1 a x_2.

Mějme tedy N vstupních trénovacích vzorů ve tvaru $\{x_i, y_i\}$, kde i je definován jako index vzoru nabývající hodnot 1..N. Dále ještě uveďme, že $x_i \in R^n$ označuje vstupní vektor příznaků a $y_i \in \{-1, +1\}$ označuje identifikátor, jež určuje zařazení do klasifikační třídy. Pro takto nadefinovanou úlohu hledáme optimální nadrovinu ve tvaru $< w, x > +b = 0$. V této definici označuje w normálový vektor nadroviny a b označuje posunutí.

Pro potřebu pozdější maximalizace hranice dále definujme podmínky pro pozitivní14 a negativní15 záznamy.

$$w \cdot x_p + b \geq 1 \tag{14}$$

$$w \cdot x_n + b \leq -1 \tag{15}$$

Kde x_p je označení pro pozitivní a x_n pro negativní záznamy. Tyto vztahy lze však ještě sloučit pomocí dříve definovaného identifikátoru y_i do jednotného vztahu 16

$$y_i(w \cdot x_i + b) \geq 1 \tag{16}$$

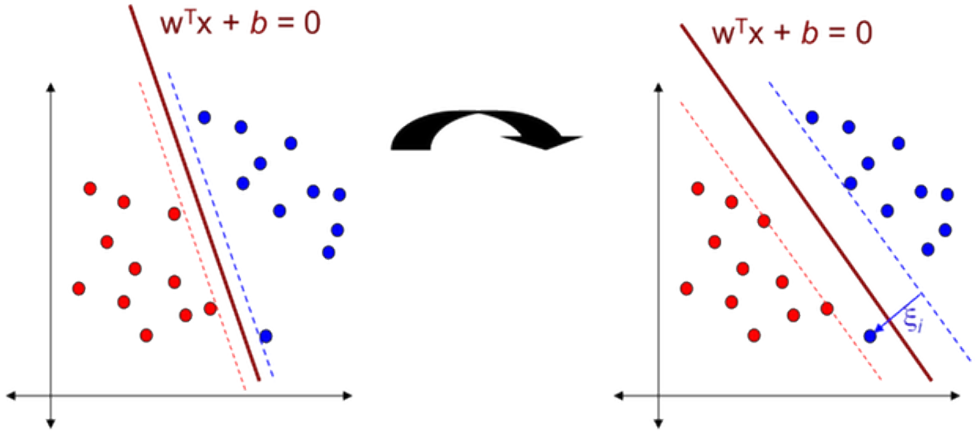
Pro výpočet velikosti hranice pak můžeme definovat následující vztah:

$$width = (x_{bp} - x_{bn}) \cdot \frac{w}{||w||} = \frac{(1-b) - (-1+b)}{||w||} = \frac{2}{||w||} \tag{17}$$

Kde $||w||$ vyjadřuje velikost normálového vektoru dělící nadroviny, x_{bp} představují pozitivní a x_{bn} pak negativní body ležící na hranici. Pro úpravu výrazu bylo využit vztahu16, kde při zvolení hraničních bodů je zajištěna rovnost obou stran.

Celkový princip metody SVM je tudíž založen na minimalizaci normálového vektoru $||w||$ za splnění omezující podmínky 16. Tím je zajištěno určení maximální hranice oddělující dané třídy a tudíž i větší výkonnost klasifikátoru. Navíc musíme ještě uvážit, že ne vždy jsou třídy ideálně lineárně separovatelné. Může například dojít k částečnému překrytí obou tříd, nebo nevhodnému zvolení dělící nadroviny, která by sice třídy oddělovala, ale dosáhla by tím velkého zúžení dělící hranice. Tento příklad je znázorněn na obrázku15 . V tomto případě se ještě používá takzvané Soft margin řešení, které se pokouší minimalizovat jak $||w||$ tak zároveň i chybu vzniklou nesprávným oddělením obou tříd. Tato optimalizační úloha by pak měla předpis

$$\min_{w \in R^d, \xi_i \in R^+} ||w||^2 + C \sum_{i=1}^{N} \xi_i \tag{18}$$



Obrázek 15: Ilustrační případ pro hledání ideální oddělující nadroviny. Vlevo lze vidět oddělení pomocí metody Hard margin, která striktně odděluje obě třídy. Vpravo pak lze pozorovat rozdělení pomocí metody Soft margin, která sice připouští částečnou chybu, ale zato razantně zvětšuje rozdělující hranici.

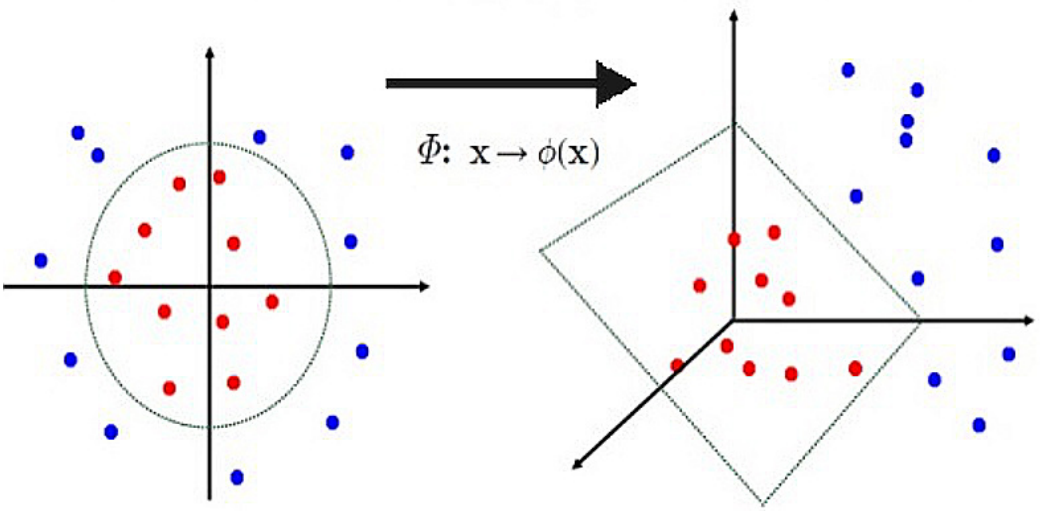
V předešlém vztahu 18 se kromě $||w||$, což představuje velikost normálového vektoru dělící nadroviny, vyskytuje navíc i suma vyjadřující součet chyb ξ_i pro všech N vzorků trénovací množiny. Navíc je zde ještě obsažena konstanta C, která určuje, zda chceme upřednostnit spíše minimalizaci prvního nebo druhého členu vztahu. Samotné hledání řešení této optimalizační úlohy však je už nad rámec této práce a proto se ji dále nebudeme věnovat. Uveďme zde však vztah19, pomocí kterého se v detekční fázi provádí klasifikace.

$$f(x) = w \cdot x + b \tag{19}$$

Neznámá proměnná x zde reprezentuje příznakový vektor neklasifikovaného objektu (v našem případě obsahu detekčního okna). Jestliže je funkční hodnota kladná nebo rovna nule, jde pravděpodobně o pozitivní vzorek. Jestliže je naopak hodnota negativní, jde s největší pravděpodobností o vzorek negativní.

Nelineární SVM

V některých případech však nemusí být lineární klasifikace úspěšná. Může jít tudíž o lineárně neseparovatelnou úlohu. V takovém případě se při řešení nejprve provádí transformace dat do nového prostoru často s vyšší dimenzí, ve kterém by byla již úloha řešitelná. Jednoduchou ilustraci případu neseparovatelné úlohy a následné transformace do jiného prostoru s vyšší dimenzí můžete pozorovat na obrázku 16.



Obrázek 16: Vlevo jsou znázorněna data v původním prostoru, kde je zřejmé, že třídy jsou lineárně neseparovatelné. Vpravo jsou původní data již transformována do prostoru s vyšší dimenzí, kde je lineární separace již proveditelná.

Celý proces transformování dat do nového příznakového prostoru, kde je možná lineární separace, je pak často označován jako kernel trick. Je to dáno tím, že pro vyjádření dat v tomto novém prostoru, se využívá takzvaných jádrových funkcí (kernel functions). Obecný tvar takovéto funkce, která vyjadřuje vztah k transformaci $\phi(x)$, je

$$k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{20}$$

, kde x_i a x_j vyjadřují vstupní vektory. Jako nejznámější varianty jádrových funkcí se potom uvádí tyto:

- Polynomální (homogenní) - $k(x, y) = (x_i \cdot x_j)^d$
- Polynomální (nehomogenní) - $k(x, y) = (x_i \cdot x_j + 1)^d$
- Radiální bázové funkce - $k(x, y) = exp(-\frac{||x_i - x_j||}{2\sigma^2})$
- Hyperbolická tangenta - $k(x, y) = tanh(kx_i \cdot x_j + c)$, pro některá $k > 0$ a $c < 0$

Pro využití těchto jádrových funkcí musíme nejprve uvést vztah pro w. Tento vektor definující rozhodovací hranici lze sestavit jako lineární kombinaci podpůrných vektorů:

$$w = \sum_i \alpha_i y_i \phi(x_i), \ \alpha_i > 0 \tag{21}$$

Díky tomu může být následně klasifikační pravidlo 19 přepsáno pomocí jádrové funkce jako:

$$f(x) = \sum_i \alpha_i y_i k(x_i, x) + b \tag{22}$$

3.3 Kalmanův filtr

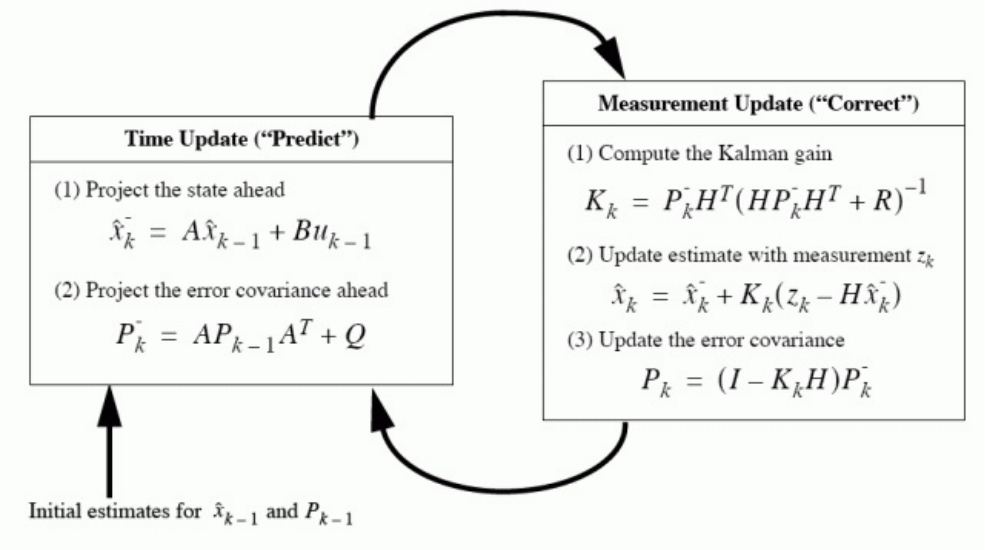
Pomocí výše zmíněných metod (HOG deskriptoru a SVM klasifikátoru) jsme popsali, jakým způsobem lze v obraze detekovat objekt zájmu, který by v našem případě představoval člověka. Nyní však musíme počítat s dynamickou povahou úlohy. Ve výsledném detektoru je totiž našim úkolem sledovat jednotlivé chodce v sekvenci snímků a co nejlépe určit jejich trajektorii. K řešení této problematiky jsem použil metodu zvanou Kalmanův filtr, jež byla publikována v 60-tých letech.

Obecně je Kalmanův filtr rekurzvní algoritmus, který pracuje se sérií vstupních dat, která jsou zatížena šumem, a snaží se produkovat statisticky optimální odhad stavu systému. Výhodou je, že tato technika je postavena právě na rekurzi. Nemusíme si tedy pamatovat rozsáhlou historii stavů, ale nový stav filtru je jednoduše získán pomocí opravy jeho předcházejícího stavu za pomocí nově získané informace. Konkrétně v našem případě nám tedy Kalamnův filtr umožní určit pravděpodobnou pozici sledovaného objektu v následujícím snímku a určit tak, zda oblasti detekované na dvou po sobě jdoucích snímcích určují tutéž osobu, či ne. V návaznosti na to, pozice detekovaných oblastí mohou být do jisté míry nepřesná. Jak už však bylo řečeno, Kalmanův filtr počítá s faktem, že změřená data jsou zatížena chybou, a snaží se ho redukovat tak, aby byl odhad stavu statisticky co nejoptimálnější.

Pro názorné vysvětlení algoritmu uvažujme případ, kdy chceme sledovat pohybující se objekt v sekvenci snímků. Kamera snímající scénu je statická a produkuje snímky v konstantních intervalech. Vyprodukované obrazy kamery poté označme $I_1, I_2, I_3, \cdots$. Sledovaný objekt v každém snímku je detekován a popsán souřadnicí jeho referenčního bodu $X = (u, v)$. Každý snímek I_t tak bude obsahovat náš bod X_t, který chceme sledovat, o souřadnicích (u_t, v_t). Nyní ještě poznamenejme, že detekce objektu zájmu v obraze je často zatížena chybou a proto je pro nás výhodnější použít Kalmanův filtr, který, jak již bylo řečeno, počítá s chybami měření a snaží se je minimalizovat. Navíc nám může Kalmanův filtr pomoci i v situaci, kdy bychom na krátký časový úsek neměli dostupnou detekci. Mohlo by jít například o situaci, kdy by byl sledovaný objekt zastíněn nějakou překážkou. V tomto případě bychom mohli využít Kalmanova filtru k predikci pravděpodobné polohy referenčního bodu objektu.

Samotný proces vyhodnocování pomocí Kalmanova filtru bychom mohli rozdělit do dvou fází. První z nich je fáze predikce, kdy Kalmanův filtr nabízí odhad stavu v současném čase pomocí minulých informací. Dalším krokem je fáze korekce, kdy probíhá

zpřesnění odhadu filtru pomocí informace z měření současného stavu. Názorné schéma průběhu Kalmanova filtru můžete vidět na obrázku 17. Pro další pokračování musíme však uvést, jak definovat jednotlivé stavy systému. V našem případě, kdy budeme analyzovat pohyb objektu ve scéně, bychom stav objektu mohli určit pomocí jeho pozice a rychlosti. Takže stav objektu v čase t můžeme definovat jako vektor $x_t = [u_t, u'_t, v_t, v'_t]^T$, kde u_t, v_t určují souřadnice referenčního bodu objektu a u'_t, v'_t rychlost pohybu ve směru souřadných os.



Obrázek 17: Schéma popisující průběh vyhodnocování pomocí Kalmanova filtru.

Predikce

Jak již bylo poznamenáno výše, první část Kalmanova filtru tvoří fáze predikce, kde je na základě dřívějších informací sestaven pravděpodobný odhad budoucího chování systému. V této části se uvádí obecně dva vzorce.

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_{t-1} \tag{23}$$

Tato rovnice určuje odhad nadcházejícího stavu $\hat{x}^-$ v čase t. Matice A je přechodová matice, která převádí stav v čase $t-1$ na stav v čase t, ovšem bez započítání vlivu chyby a řídící funkce. Pro představu zde může být například uveden způsob, jak pozice a rychlost v čase $t-1$ působí na pozici a rychlost v čase t. Pro náš příklad sledování objektů, by mohla tato přechodová matice vypadat následně:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Vektor $\hat{x}_{t-1}$ vyjadřuje stav systému v čase $t-1$. Dále se zde ještě může vyskytovat matice B, která aplikuje efekt řídících vstupů, které jsou reprezentovány vektorem u_{t-1}, do systému. Pod pojmem řídících vstupů si můžeme představit, že by vstupní vektor u_{t-1} udával například změnu směru pohybu. Matice B potom aplikuje tuto změnu na stav systému. Pro náš příklad se sledováním objektu v obraze bychom však matici B a vektor u_{t-1} mohli vypustit.

$$P_t^- = AP_{t-1}A^T + Q \tag{25}$$

Tato druhá rovnice pak určuje výpočet predikce P_t^-, která představuje disperzní matici chyby odhadu. Ve zjednodušenosti by se dalo říct, že P_t^- popisuje chybu odhadu, jež je vyjádřen vektorem $\hat{x}_t$. Jestliže tedy obsahuje tato matice vysoké hodnoty, vypovídá to o velké variabilitě stavu $\hat{x}_t$. Samotnou matici bychom mohli definovat jako:

$$e_t^- \equiv x_k - \hat{x}_t^-;\ P_t^- = E[e_t^- e_t^{-T}] \tag{26}$$

Nakonec matice Q je nazývaná kovarianční matice procesního šumu. Neboli tato matice vyjadřuje jistou nedůvěru v to, že popis průběhu systému je správný.

Korekce

Druhou fázi Kalmanova filtru představuje jeho korekce. Cílem této části je upravit informace obsažené ve filtru a zajistit tak následnou přesnější predikci. Základem této operace je, že získané hodnoty měření v čase t, které reprezentujme vektorem z_t, porovnáme s hodnotami predikce. Ty jsme získali jako výstup v předchozí fázi. Pomocí tohoto porovnámí potom provedeme aktualizaci filtru pro budoucí stavy. V této části se obecně využívá následujících vztahů:

$$\tilde{y}_t = z_t - H\hat{x}_t^- \tag{27}$$

Tento vztah jednoduše vyjadřuje rozdíl mezi naměřenými hodnotami z_t a odhadovaným stavem $\hat{x}_t^-$, který jsme obdrželi před měřením ve fázi predikce. Jelikož však může mít vektor stavu jinou strukturu informací než vektor měření, je třeba provést násobení takzvanou matici měření H. Tato matice popisuje, jak měřené veličiny závisí na veličinách popsaných ve stavovém vektoru. Opět můžeme uvést příklad pro naši situaci, kdy sledujeme pohyb objektu v obraze. Jak jsme již výše zmínili, stavový vektor obsahuje informace jak o souřadnicích referenčního bodu, tak i o rychlosti pohybu. Z měření však dostáváme informace pouze o poloze, proto by mohl být zápis následující:

$$x_t = \begin{bmatrix} u_t \\ u'_t \\ v_t \\ v'_t \end{bmatrix};\ z_t = \begin{bmatrix} u_t \\ v_t \end{bmatrix};\ H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{28}$$

Další vztah lze pro přehlednost rozepsat do dvou na sebe navazujících částí:

$$S_t = H P_t^- H^T + R$$
$$K_t = P_t^- H^T S^{-1} \tag{29}$$

Matice S_t v první části vyjadřuje odhadovanou kovarianční matici pro naměřené hodnoty vektoru z_t. Její hodnoty vypovídají o variabilitě měření. Jestliže je tato variabilita vysoká, vypovídá to o velké změně v naměřených hodnotách a naopak. Matice R zde následně vyjadřuje odhadovanou chybu v měření. Druhý výraz udává, jak získat Kalmanův zisk neboli zesílení vyjádřené maticí K_t. V zjednodušené podobě bychom mohli prohlásit, že Kalmanův zisk nám udává, jak moc budeme měnit vypočtený odhad $\hat{x}_t^-$ na základě naměřených hodnot z_t. V souvislosti s prvním výrazem můžeme ještě pro vysvětlení poznamenat, že čím větší bude variabilita měření S_k, tím menší bude výsledný Kalmanův zisk a tím méně bude stav ovlivněn aktuálním měřením.

$$\hat{x}_t = \hat{x}_t^- + K_t \tilde{y}_t \tag{30}$$

$$P_t = (I - K_t H) P_t^- \tag{31}$$

Výrazy 30 a 31 už přímo udávají, jakým způsobem dostat hodnoty stavu $\hat{x}_t$ a disperzní matice chyby odhadu P_t aktualizované pomocí naměřených veličin reprezentovaných vektorem z_k.

Na těchto dvou popsaných krocích (predikce a korekce) stojí celý algoritmus Kalmanova filtru. Výše zmíněný popis nemá za úkol do hloubky popsat celou tuto techniku, ale spíše pouze nastínit její průběh. Záměrně zde není například uvedeno, jakým způsobem by se měly stanovovat hodnoty matic Q a R, které představují kovarianční matice procesní chyby a chyby měření. Také zde nejsou obsaženy důkazy platnosti jednotlivých vztahů, poněvadž to by již přesahovalo rámec této práce. Účelem bylo spíše objasnit fungování tohoto procesu a vyjádřit také jeho vztah při použití v aplikaci pro sledování pohybujících se objektů v obraze, jež je nepostradatelnou součásti cílové aplikace.

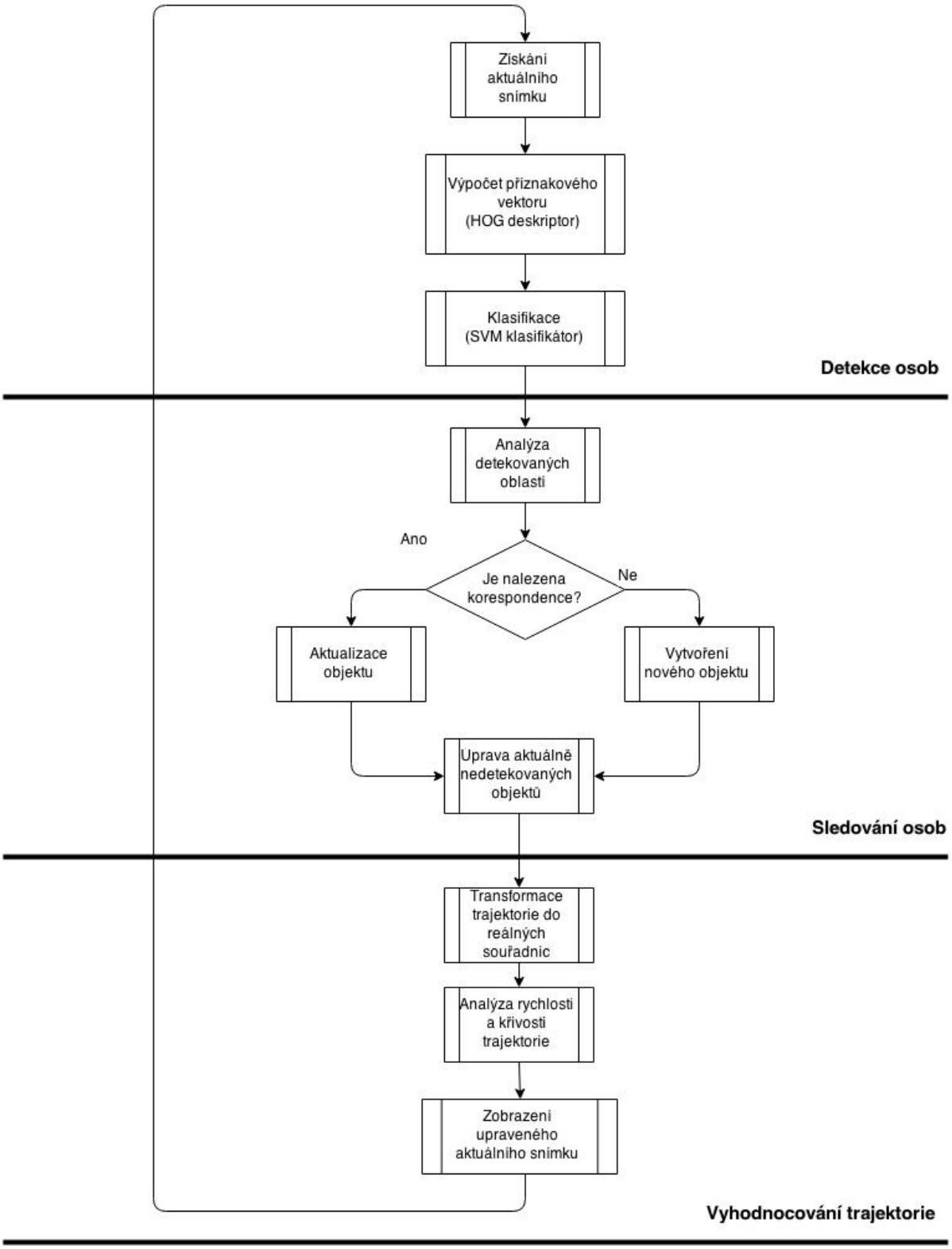
4 Proces sledování osob a vyhodnocování trajektorií

Tuto část práce bych chtěl zaměřit již jako pohled na celkovou aplikaci a objasnit její průběh. Aplikace je stavěna tak, že se pro každý snímek dané sekvence snaží detekovat všechny vyskytující se osoby a v návaznosti na to poté upravovat jejich aktuální informace. Pro jednotlivé snímky je třeba provést celou posloupnost procesů, jejíž stručné zobrazení lze vidět na přiloženém diagramu 18. Jak je z obrázku patrné, celý proces můžeme rozdělit do tří základních úkonů, které představují:

1. Detekci osob
2. Sledování osob
3. Vyhodnocování trajektorie

Následující podkapitoly tudíž budou věnovány právě popisu každého takto zmíněného procesu. Kromě detailnějšího vysvětlení průběhu každého úkonu, bude zde věnován také prostor pro analýzu volitelných parametrů. Celková aplikace totiž stojí na metodách, které silně závisí na zvolených parametrech. Zde tedy bude zdůrazněno, jak hodnoty parametrů mohou ovlivňovat například úspěšnost detekce nebo rychlost celého programu.

Je důležité upozornit, že obsahová část této kapitoly již nebude zaměřena na teoretický popis HOG deskriptoru, SVM klasifikátoru ani Kalmanova filtru. Tyto tři techniky jsem totiž zařadil jako hlavní pokročilé metody a pro jejich vysvětlení jsem věnoval samostatnou kapitolu. Jestliže se tudíž s těmito metodami chcete blíže seznámit, využijte poznatků obsažených v kapitole 3. V následujícím textu se na tyto metody budeme odkazovat, ale nebudeme již jejich algoritmus detailně popisovat.



Obrázek 18: Diagram popisující stručný průběh algoritmu výsledné aplikace.

4.1 Detekce osob

První fází, kterou každá iterace prochází by se souhrnně dala nazvat detekce osob v aktuálním snímku. Jak již název vypovídá, vstupem pro tento krok je obraz reprezentující scénu, v níž máme hledat osoby. Výstupem je pak seznam oblastí, které nám detektor vyhodnotil jako pozitivní, neboli obsahující osobu. Každá detekovaná oblast je definována jak svými souřadnicemi, tak i svou výškou a šířkou. Hlavním prvkem je zde tedy detektor, který se skládá z HOG deskriptoru a SVM klasifikátoru, jejichž technika byla podrobně popsána výše v podkapitolách 3.1 a 3.2.

Deskriptor

Prvním základním kamenem detekce je deskriptor. Jeho účelem je co nejlépe popsat obsah vyhodnocované oblasti a sestavit z něho příznakový vektor, který by obsahoval co nejpodstatnější informace potřebné k detekci. Častou problematikou deskriptorů je otázka, jaké veličiny pro samotný popis zvolit. Obecně by příznaky měli co nejlépe splňovat podmínku, že jestliže se jedná o oblasti stejné třídy, měly by mít podobné hodnoty. Naopak, jestliže se jedná o objekty různých tříd, měly by být i hodnoty příznaků co nejrůznější.

Důležitou roli zde hraje i velikost příznakového vektoru. Pro představu si definujme, n-dimenzionální příznakový prostor, kde n představuje počet hodnot v příznakovém vektoru. Každá popisovaná oblast tak definuje jeden vektor v tomto prostoru. Klasifikaci oblastí potom můžeme definovat jako porovnávání vzdálenosti zkoumaného vektoru od pozitivních a negativních případů, nebo také zkoumání jeho polohy vzhledem k hranici dělící obě třídy. Jestliže se zvětší velikost příznakového vektoru, naroste i dimenze příznakového prostoru. Nejen, že bude následkem toho klasifikace výpočetně náročnější, ale také se zvýší nároky na trénovací data, podle kterých se poté stanoví dělící hranice mezi oběma třídami. Naopak jestliže bude příznakový vektor až příliš stručný, může nastat ztráta některých potřebných informací a výsledná klasifikace tak může být neúspěšná.

Detekce lidí je navíc složitá i tím, že například na rozdíl od objektů, které nemění svůj tvar (např auta), lidé se velmi často vyskytují v různých postojích, což komplikuje práci detektoru. Studie problému vhodného souboru příznaků pro detekci lidí ukázala, že lokálně normalizované histogramy orientovaných gradientů poskytují výborný výkon ve srovnání s ostatními existujícími soubory příznaků.

Jak již bylo popsáno dříve v kapitole 3.1, algoritmus HOG deskriptoru bychom mohli definovat řadou po sobě jdoucích kroků:

1. Normalizace jasu
2. Výpočet gradientů
3. Stanovení váhovaného histogramu gradientů
4. Normalizace jasu uvnitř bloku

5. Sestavení vektoru příznaků

Téměř v každé této fázi však můžeme narazit na nastavitelný parametr, který do určité míry ovlivňuje celý průchod a má tím pádem také vliv na pozdější detekci.

Normalizace jasu V první fázi, která se nazývá normalizace jasu, se HOG deskriptor snaží snížit vliv světelných podmínek v obraze. Jde o jedinou úpravu, kterou snímek prochází, než je analyzován. Zde se úprava provádí podle vztahu 9. U HOG deskriptoru implementovaného pomocí OpenCV však může mít parametr γ pouze hodnotu 1, což by ve skutečnosti znamenalo vynechání korekce, nebo 0.5. Výchozí hodnota pro výslednou aplikaci je právě hodnota 0.5.

Vliv však může mít i barevná reprezentace snímku. Mezi ně můžeme zařadit jak obrazy ve stupních šedi, tak i barevné snímky v RBG nebo LAB reprezentaci. Autoři dospěli k tomu, že díky barevné reprezentaci sice dojde k malému zvýšení úspěšnosti, ale to za cenu větší výpočetní náročnosti. Z tohoto důvodu byl pro konečnou aplikaci zvolen postup, kdy každý vstupní snímek je nejprve převeden do stupňů šedi a až poté je analyzován. Dojde tak sice k malému snížení výkonnosti, ovšem celá detekce se podstatně zrychlí.

Výpočet gradientů Dalším krokem je výpočet gradientů, který je pro tento postup nepostradatelný. Díky gradientům totiž můžeme popsat velikost a směr hran v obraze a vytvořit si tak lepší představu o jeho celkovém obsahu. Logicky bychom v této fázi mohli uvažovat o různých způsobech výpočtů gradientů, při kterých bychom mohli využít následující masky:

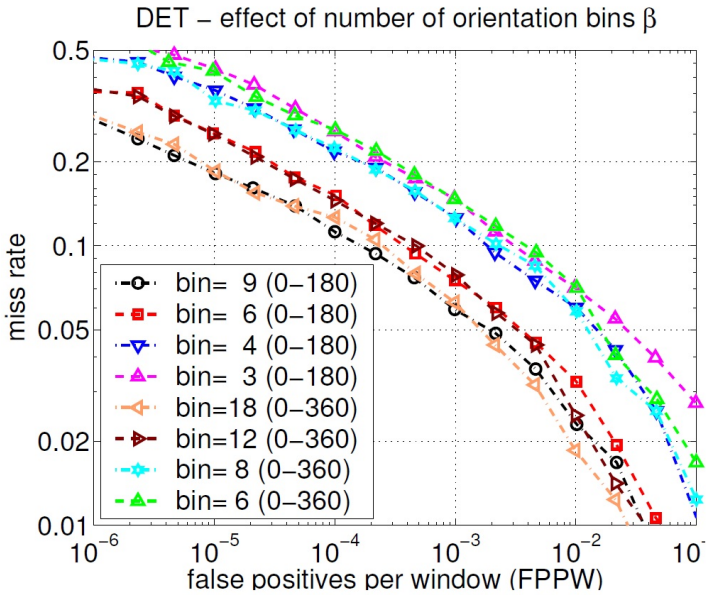
$$1D : \begin{bmatrix} -1 & 0 \end{bmatrix};\ \begin{bmatrix} -1 & 0 & 1 \end{bmatrix};\ \begin{bmatrix} 1 & -8 & 0 & 8 & -1 \end{bmatrix} \tag{32}$$

$$2D : \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix};\ \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix};\ \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{33}$$

Ovšem pomocí experimentálních testů se ukázalo, že pomocí masek $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ a $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ lze dosáhnout nejlepších výsledků. Proto se využívají i pro tuto aplikaci.

Stanovení váhovaného histogramu gradientů V tomto kroku je účelem seskupit vypočtené gradienty do větších celků, které se nazývají buňky (cells). V rámci každé buňky se následně vytvoří histogram obsažených gradientů. Tento krok sice povede k tomu, že ztratíme informace o přesné poloze jednotlivých gradientů uvnitř buňky, ovšem experimentálně se ukázalo, že pro popis nám postačí vědět pouze rozložení jednotlivých gradientů. K tomu nám slouží právě již zmíněný histogram. Zde však může nastat otázka, jaké parametry pro seskupování zvolit a jaký je jejich vliv na výkonnost.

V první volbě se budeme zabývat parametry pro histogram orientovaných gradientů. Můžeme volit jak z různého počtu zásobníků (bins), tak i z typu rozlišovaného rozsahu (signed - 0°- 360°, unsigned - 0°- 180°). Výsledky, kterých dosáhli autoři HOG detektoru jsou k vidění na následujícím grafu 19.



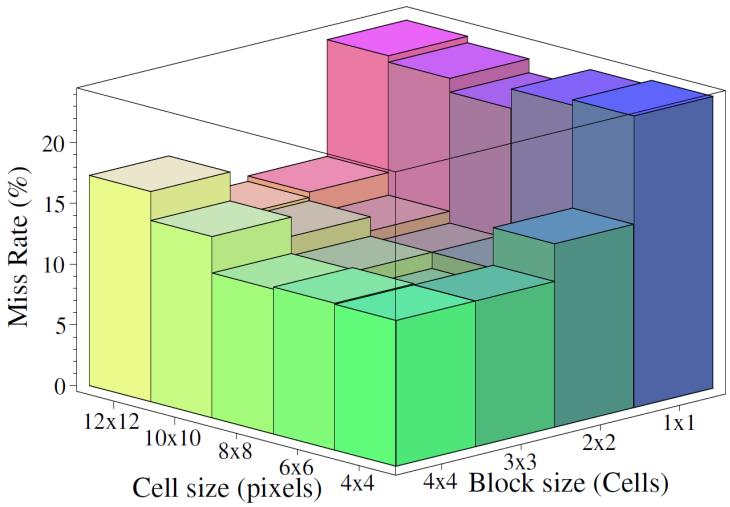
Obrázek 19: Vizuální zobrazení úspěšnosti detektorů s různými parametry pro histogramy orientovaných gradientů. Každá křivka reprezentuje jednu instanci. Vysvětlení měřených hodnot je uvedeno níže.

$TruePositive$ - označují případy, kdy detektor správně vyhledá objekt zájmu.
$FalsePositive$ - detektor označí oblast, která není ve skutečnosti objektem zájmu.
$FalseNegative$ - detektor neoznačí oblast, která ve skutečnosti představuje objekt zájmu.

$$MissRate = \frac{FalseNegative}{TruePositive + FalseNegative}$$

$FPPW$ - počet nesprávných označení ($FalsePositive$) na jedno detekční okno

Dále se zde vyskytuje ještě možnost měnit velikosti celků, do kterých se gradienty a histogramy orientovaných gradientů sdružují. Zde částečně zasáhneme i do následující fáze, protože kromě volené velikosti buňky, která dává vzniku histogramu gradientů, se budeme zabývat i velikostí bloku, podle kterého se provádí normalizace. I zde je však možné vycházet z experimentálního měření, které provedli autoři. Jejich závěry jsou k vidění na následujícím obrázku 20.



Obrázek 20: Grafické znázornění vlivu zvolené velikosti buňky a bloku na úspěšnost celého detektoru.

Pro výslednou aplikaci je standardně pro histogram voleno 9 zásobníků v rozmezí 0°-180°. Velikost každé buňky je 8x8 pixelů. Blok je zvolen jako čtvercový o rozměrech 2x2 buňky. Překrývání každého bloku je pak nastaveno na 50%.

Normalizace kontrastu uvnitř bloku Posledním krokem, který zde v souvislosti s HOG deskriptorem zmíním je normalizace kontrastu uvnitř každého bloku. Jak již bylo vysvětleno dříve, bloky (blocks) jsou prostorově obsáhlejší celky do kterých se shlukují jednotlivé histogramy. Normalizace se zde provádí za účelem velkého zvýšení odolnosti deskriptoru vůči změnám osvětlení, které se mohou ve snímcích vyskytovat.

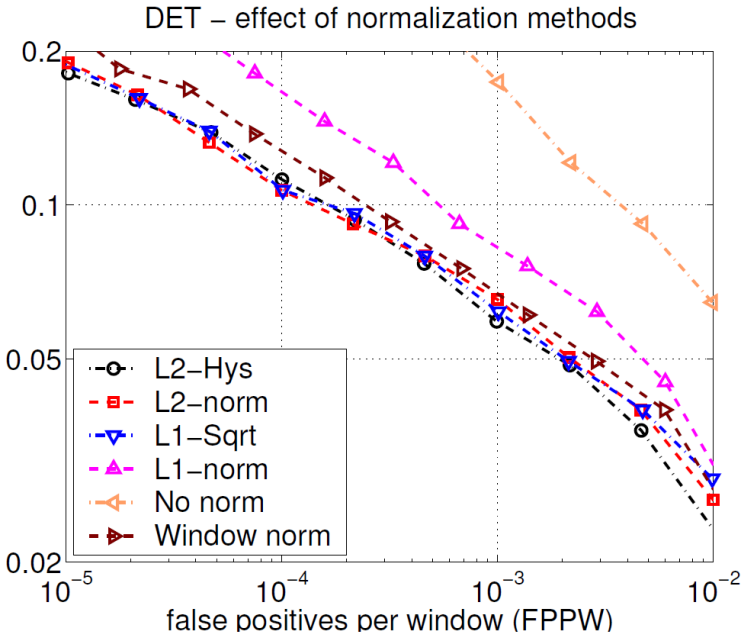
Zde se nabízí hned několik metod normalizace:

- L2-norm - $f = \frac{v}{\sqrt{||v||_2^2 + e^2}}$
- L2-hys - L2-norm metoda následovaná omezením maximálních hodnot jistým prahem t (např. $t = 0.2$) a poté je vektor hodnot opět normalizován
- L1-norm - $f = \frac{v}{||v||_1 + e}$
- L1-sqrt - $f = \sqrt{\frac{v}{||v||_1 + e}}$

Kde nenormalizovaný vektor v obsahuje všechny hodnoty histogramů v bloku a e představuje velmi malou konstantu, z důvodu aby nedošlo k dělení nulou. Pro výraz $||v||_p$ platí vztah:

$$||v||_p = (\sum_{i=1}^{n} |v_i|^p)^{\frac{1}{p}} \tag{34}$$

Porovnání vlivu normalizace na úspěšnost celého detektoru můžete vidět na následujícím grafu 21. Implementace v OpenCV sice dovoluje použít pouze L2-hys metodu, ale ta podle zjištění dosahuje nejlepších výsledků, takže toto omezení nijak nepřekáží. Volitelný je však parametr t, který představuje práh pro výsledky první normalizace. Standardně je zvolen jako hodnota 0.2.



Obrázek 21: Vizuální zobrazení úspěšnosti detektorů pro rozdílné typy použité normalizace bloků. Každá křivka reprezentuje jednu instanci.

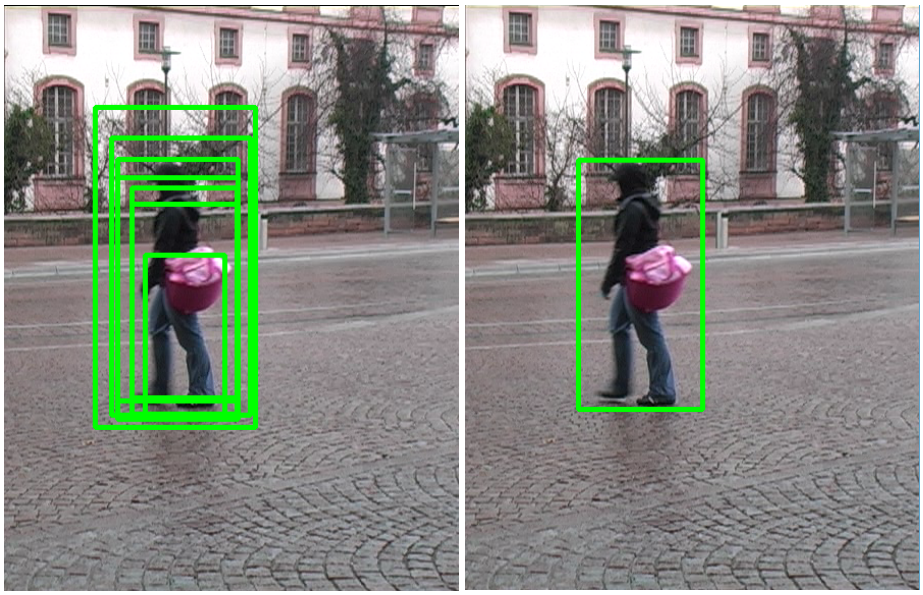
Klasifikátor

V minulém kroku nám použitý deskriptor pomohl popsat co nejužitečněji obsah v detekčním okně obrázku. Jak již bylo popsáno dříve v kapitole 3.1, HOG deskriptor popisuje každý snímek pomocí velkého množství překrývajících se detekčních oken. Poskytne nám tedy mnoho příznakových vektorů, u kterých musíme určit, zda obsahují lidskou podobu, či nikoliv. K tomuto účelu slouží SVM klasifikátor.

Aby byla dodržena jeho funkčnost, musí být nejprve podroben učení. V tomto procesu mu je předloženo velké množství trénovacích dat, ve kterém jsou obsaženy jak pozitivní (obsahující člověka) tak negativní (neobsahující člověka) případy. Tyto záznamy jsou samozřejmě předkládány ve formě příznakových vektorů. Z toho logicky plyne, že nastavení HOG deskriptoru (jako je třeba velikost okna, velikost bloků, atd.), které jsme použili pro trénování klasifikátoru, musí být stejné jako to, které používáme pro detekci. V opačném případě by mohla klasifikace naprosto selhat, protože by si dimenze příznakového vektoru a příznakového prostoru nemusely odpovídat, nebo by hodnoty vyjadřovaly jiné veličiny, než pro které byly naučeny.

I zde by mohly být k dispozici varianty lineárního nebo nelineárního SVM klasifikátoru. To by ovšem mohlo vést k přílišné složitosti nastavení a proto se v celé aplikaci používá jednoduchý SVM klasifikátor s parametrem $C = 1$, jehož funkce je vysvětlena pomocí vztahu 18.

Tímto postupem tak dostaneme množinu detekčních oken, které byly klasifikátorem vyhodnoceny jako pozitivní. Jelikož se však jednotlivá detekční okna překrývají a navíc se testují i jejich různé velikosti, můžeme pro jeden reálný sledovaný objekt dostat velké množství pozitivních odpovědí. Tento případ můžeme sledovat na následující ukázce 22. Pro zabránění tohoto efektu se používá takzvaného grupování, které se pokouší seskupit vysoce se překrývající detekční nálezy.



Obrázek 22: Ilustrace ukazující funkci seskupování detekcí. Vlevo je ukázka s vypnutou funkcí seskupování. Vpravo je grupování zapnuto.

Kromě nastavení zmíněného seskupování, vyskytují se u HOG detektoru ještě další dva důležité parametry. Prvním z nich je takzvaný $hitThreshold$, který dovoluje posouvat hranici, při které se rozhoduje, zda se jedná o pozitivní či negativní případ. Kladné hodnoty tohoto parametru určují zpřísnění podmínky pro pozitivní vyhodnocení. Negativní hodnoty naopak vyhodnotí pozitivně i některé případy, které by byly za normálního nastavení negativní. Standardně má tento parametr nastavenu nulovou hodnotu. Nakonec tedy zmiňme ještě parametr $scale$. Jak jsme již dříve uvedli, detekční okno se testuje v různých velikostech, takže bychom mohli předpokládat, že parametr $scale$ určuje míru změny velikosti okna. Ta je však přesně dána svými rozměry v pixelech, takže nemůže být měněna. Parametr $scale$ tedy ve skutečnosti určuje míru podvzorkování celého snímku, takže velikost okna sice zůstane zachována, ale bude pokrývat větší oblast obrazu.

4.2 Sledování osob

V tomto kroku budeme vyhodnocovat detekce, které byly výstupem předešlé fáze. Hlavním cílem je zde vnést do systému jistou dynamiku. Detekce osob totiž pracuje s každým snímkem sekvence nezávisle, což je ve výsledku nevyhovující. Abychom totiž mohli sestavit trajektorii sledovaného objektu, musíme analyzovat jeho pohyb napříč celou sekvencí.

V této sekci tudíž popíšeme algoritmus, jehož účelem je spravovat informace o osobách vyskytujících se ve scéně. Toho je dosaženo za pomocí analýzy detekcí získaných v aktuálním snímku. Vstupem pro tento krok tedy bude dříve získaná množina pozitivních detekcí. V zjednodušenosti bychom mohli tento proces popsat jako hledání pozice dříve detekovaných postav v novém snímku.

Základním kamenem je již dříve popsaný Kalmanův filtr. Připomeňme si, že Kalmanův filtr pracuje ve dvou fázích. První z nich je predikce, kdy pomocí již dříve získaných informací o stavu objektu můžeme určit odhad jeho budoucího stavu. Druhou fází je korekce. Zde se provádí aktualizace filtru pomocí reálných naměřených hodnot zatížených chybou. Tento krok má za následek jak získání odhadu opravených hodnot měření, tak i zvýšení přesnosti predikce pro následující stavy. Detailnější informace o tomto algoritmu lze najít v samostatné kapitole 3.3.

Pro snadnější manipulaci je každá detekovaná oblast popsána souřadnicemi jejího referenčního bodu. Právě s těmito hodnotami pak Kalmanův filtr pracuje. Pro každou instanci osoby ve scéně je tedy v prvním kroku predikována poloha referenčního bodu detekčního okna. Poté následuje hlavní část tohoto kroku a tou je nalezení odpovídající detekované oblasti pro každou sledovanou osobu ve scéně.

Ke hledání bychom mohli využít velice jednoduchého postupu porovnávání vzdáleností. Jednoduše bychom jako odpovídající detekovanou oblast označili tu, která by se nacházela nejblíže. Tento přístup by však fungoval pouze ve velmi jednoduchých scénách. Problémy by však mohly nastat už při zastínění osoby překážkou, nebo při míjení se několika osob. Proto je zde využita poněkud robustnější varianta.

Ta k přesnějšímu nalezení správného detekčního okna používá tři různé aspekty. Prvním z nich je již zmíněný výpočet vzdálenosti.

$$D_{X_i,Y_j} = \sqrt{(x_{i1} - y_{j1})^2 + (x_{i2} - y_{j2})^2} \tag{35}$$

Kde $X_i = (x_{i1}, x_{i2})$ udává vektor popisující predikci referenčního bodu i-tého sledovaného objektu. $Y_i = (y_{j1}, y_{j2})$ vyjadřuje vektor pro referenční bod j-té detekované oblasti.

Další aspekt tvoří porovnání rozměrů což vystihuje situaci, kdy se lidé míjejí a navzájem se zastiňují. Každý z nich je tudíž v jiné vzdálenosti od kamery a jejich velikosti

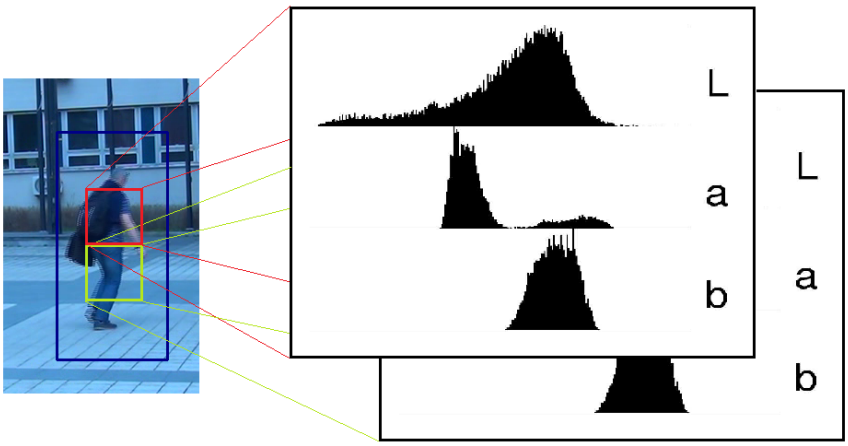
v obraze jsou tudíž ve většině případů rozdílné. Jelikož má každé detekční okno pevně daný poměr stran, stačí nám pro porovnání rozměrů vypočítat pouze rozdíl výšky sledovaného objektu a výšky detekčního okna.

Během testování HOG detektoru se však ukázalo, že velikost detekčního okna pro sledovaný objekt může být nestabilní. Proto zde byl přidán ještě třetí aspekt, čímž je barevné složení objektu. Pro vyhodnocení shodnosti barevného rozložení je využito Bhattacharyova porovnání histogramů, jež je realizováno vztahem 36, během kterého probíhá i normalizace.

$$b(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_j H_1(j) \cdot \sum_j H_2(j)}}} \tag{36}$$

Jelikož chceme docílit barevného porovnání, musíme porovnat histogramy v rámci každého kanálu obrazu. Místo standardního barevného modelu RGB je však využit model LAB, jehož výhodou je oddělení jasové složky (L) od barevných (AB). Jestliže v návaznosti na to zmenšíme váhu hodnoty získané porovnáním histogramů jasového kanálu, získáme větší odolnost celého testování vůči změnám osvětlení.

Dále si musíme uvědomit, že velkou část detekčního okna může tvořit pozadí. Našim úkolem je však analyzovat pouze barevné složení sledovaného objektu. Z toho důvodu se analyzuje pouze její část, jež byla experimentálně určena jako středová oblast o poloviční rozměrech detekčního okna. Ta je následně ještě horizontálně rozdělena na dvě poloviny, které oddělují barevné složení oblečení horní a dolní části těla člověka. Níže můžete vidět ilustraci znázorňující analyzované oblasti.



Obrázek 23: Ukázka získávání informace o barevném složení osoby.

Mějme tedy instanci X_i, která představuje i-tou detekční oblast, a instanci Y_j, která vyjadřuje j-tou detekovanou osobu. Pro porovnávání detekční oblasti X_i a osoby Y_j nyní

definujme součet tří hodnot získaných výše zmíněnými vztahy. Jelikož platí, že čím různější porovnávané objekty budou, tím vyšší bude také jejich součet, můžeme výslednou hodnotu $p_{i,j}$ nazvat jako penalizaci. Abychom však mohli lépe určovat významnost jednotlivých aspektů, je každý sčítanec vynásoben koeficientem určujícím jeho váhu. Výsledný vztah pro penalizaci vypadá následně:

$$p_{i,j} = w_d \cdot d_{i,j} + w_h \cdot h_{i,j} + w_b \cdot (b^L_{i,j} \cdot w_L + b^A_{i,j} + b^B_{i,j}) \tag{37}$$

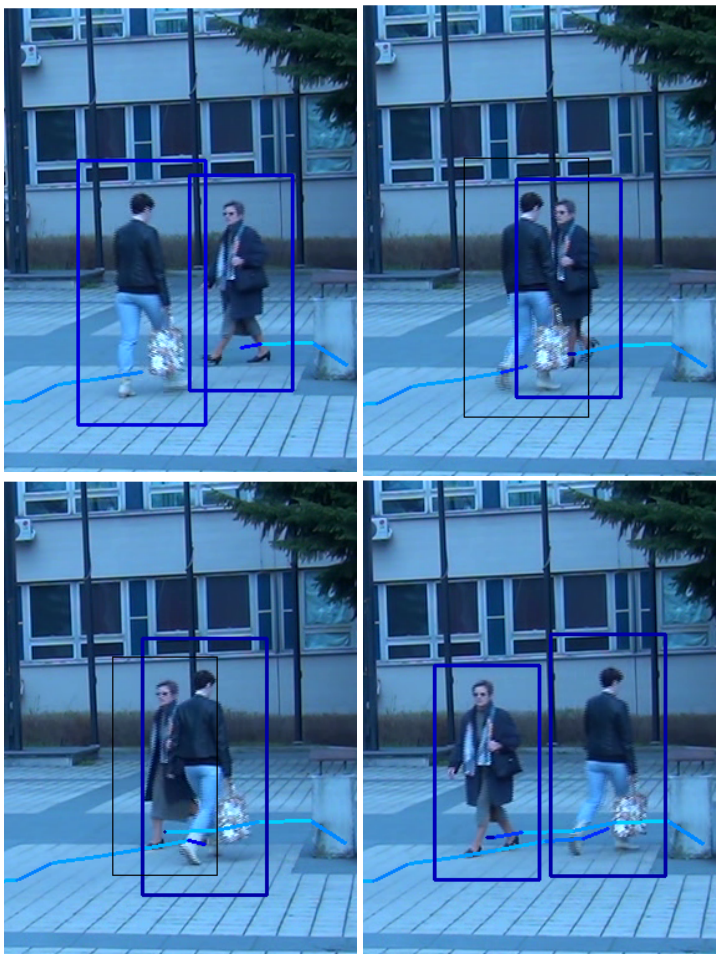
Kde $d_{i,j}$ je hodnota euklidovské vzdálenosti referenčních bodů, $h_{i,j}$ určuje rozdíl výšek a $b^L_{i,j}, b^A_{i,j}, b^B_{i,j}$ jsou hodnoty pro Bhattacharyovo porovnání histogramů jednotlivých kanálů LAB. Koeficienty w_d, w_h, w_b určují váhy přiřazené jednotlivým aspektům a nakonec člen w_L určuje koeficient snižující váhu pro porovnání jasových histogramů. Jak již bylo řečeno, důvodem je zlepšení odolnosti proti světelným změnám.

Algoritmus v této části tedy nejprve vypočítá penalizaci pro každou dvojici X_i a Y_j a vytvoří z nich tabulku. Následným úkolem je najít vhodné dvojice X_i a Y_j, jejíž penalizace by byla co nejnižší. Během tohoto procesu však musí být splněny určité podmínky. První z nich je, že penalizace pro určitou dvojici musí být menší, než práh T, což je volitelný koeficient. Takovýto práh nám tak definuje maximální přípustnou penalizaci pro přiřazení. Druhou podmínkou je, že pro každou detekovanou oblast musí existovat nejvýše jeden sledovaný objekt. Zároveň však také musí platit, že pro každý sledovaný objekt musí existovat nejvýše jedna detekovaná oblast. Tato podmínka má za úkol správně vyhodnotit případy, kdy se dvě osoby míjí a navzájem se tak zastiňují. Během takovéto situace by mohlo nastat, že by dva objekty splnili první podmínku pro jedno společné detekční okno. V návaznosti na druhou podmínku však algoritmus tento případ vyhodnotí tak, že detekované okno přiřadí pouze objektu, který má menší penalizaci. Názornou ukázku lze vidět na sekvenci snímků přiloženou níže 24.

Tímto procesem vyhodnocování tak můžeme logicky dojít ke třem různým případům:

Nalezne se vhodná dvojice X_i a Y_j
: V tomto případě jsme tedy k osobě vyskytující se ve scéně úspěšně přiřadili detekční okno, které s nejvyšší pravděpodobností popisuje jeho polohu v novém snímku. Pro aktualizaci Kalmanova filtru použijeme souřadnice referenčního bodu okna, ale pro určení aktuální polohy použijeme až následného zpřesněného odhadu měření. Provede se také aktualizace všech atributů popisující osobu, jako je například barevné složení a rozměr popisné oblasti.

Pro detekční oblast X_i nebyla nalezen vhodná osoba
: Jestliže pro detekční okno nenalezneme vyhovující osobu, znamená to pravděpodobně, že jsme ve scéně detekovali nového člověka. V tomto případě tedy jednoduše vytvoříme novou instanci osoby, kterou zahrneme do scény. Provedeme také inicializaci Kalmanova filtru a ostatních popisných atributů.



Obrázek 24: Sekvence snímků ukazující případ překrývání osob a vyhodnocování této situace aplikací.

Pro detekovanou osobu Y_j nebylo nalezeno vhodné detekční okno

V tomto případě máme sice ve scéně evidovanou osobu, ale pomocí detekce nebyla v aktuálním snímku nalezena. Tato situace může nastat například, když osoba opustí scénu, je zakryta překážkou, nebo když se její detekce v aktuálním snímku nepodaří. Ve všech případech se postupuje tak, že její poloha je predikována Kalmanovým filtrem a pokus o opětovnou detekci proběhne v dalším snímku. Postupně je však zaznamenán počet po sobě jdoucích snímků, po který byl objekt nedetekovaný. Jestliže tato hodnota dosáhne určitého prahu, je objekt označen za ztracený a ze scény je vymazán.

4.3 Vyhodnocování trajektorie

V prvním kroku bylo našim cílem detekovat osoby v aktuálním snímku. Obsahem druhého kroku bylo zajistit jistou dynamičnost sytému v rámci sekvence snímků. K detekovaným objektům z minulých snímků se přiřadí nejpravděpodobnější detekovaná oblast a tím se zajistí aktualizace jeho stavu v novém snímku. Pomocí předchozích kroků již tedy nyní známe pozice, na kterých se objekt vyskytoval v průběhu sekvence snímků.

Konečným krokem tohoto procesu je analyzovat získanou posloupnost pozic a získat tak potřebné informace o pohybu osoby. Jako hlavní informaci bychom mohli považovat grafické zobrazení trajektorie. K tomu je však ještě vyhodnocována aktuální rychlost pohybu a také neobvyklé změny jeho směru.

Nejjednodušším úkolem je získání grafického zobrazení trajektorie. K tomu nám postačí proložit posloupnost bodů, které vyjadřují pozice objektu v obraze během detekování, příslušnou křivkou. Pro účely aplikace byla tedy zvolená jednoduchá lomená čára. Zde však musíme vzít v potaz fakt, že jak již bylo dříve zmíněno, polohy referenčních bodů mohou být zatíženy šumem. Ten by mohl mít nepříjemný dopad na pozdější výpočet rychlosti a křivosti. Z toho důvodu je pro vyznačení trajektorie použit pouze každý n-tý referenční bod. Vypočtené informace budou sice vztaženy na delší časový úsek, ale ovlivnění šumem bude menší. Navíc dojde i k částečnému zrychlení aplikace.

Pro výpočet rychlosti a křivosti pohybu si však musíme uvědomit, že získané pozice nám neudávají reálnou polohu, ale pouze polohu objektu v obraze. Musíme tedy nejdříve zajistit transformaci, která by k obrazovým souřadnicím pomohla získat odhad pozice bodu v reálném prostoru. K tomu účelu nám postačí výpočet příslušné perspektivní matice.

Pro každý referenční bod nyní přesně specifikujme jeho polohu vůči detekčnímu oknu, což bude mít důležitý vliv. Našim předpokladem je, aby referenční bod určoval obrazové souřadnice dotyku sledované osoby s podlahou. Tento postup nám zajistí přenesení informace o poloze objektu do roviny podlahy, což nám zjednoduší výpočet reálných souřadnic. Pomocí experimentálních pokusů byla pozice referenčního bodu určena jako pětina výšky a polovina šířky detekčního okna.

Z předešlého postupu již máme zajištěn předpoklad znalosti pozice bodu v rovině podlahy. Pomocí snímání kamerou však dostáváme pouze její projektivní zobrazení. Musíme tedy vypočítat projektivní matici, která by pro každou souřadnici v rovině obrazu dokázala určit souřadnice v rovině podlahy. Toho docílíme pomocí procesu kalibrace. Kdy hledáme ke zvoleným obrazovým bodům $X = (x, y)$ jejich korespondující obrazy $X' = (x', y')$ v rovině podlahy. Pro výpočet projektivní matice můžeme vyjít z následujícího vztahu:

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{38}$$

Kde po vyjádření x' a y' dostaneme

$$x' = \frac{ax + by + c}{gx + hy + 1}$$
$$y' = \frac{dx + ey + f}{gx + hy + 1} \tag{39}$$

Tyto tvary lze přepsat do homogeních rovnic jako:

$$x'(gx + hy + 1) - (ax + by + c) = 0$$
$$y'(gx + hy + 1) - (dx + ey + f) = 0 \tag{40}$$

Jak lze vidět, tato soustava obsahuje osm neznámých (a,b... h). Každá dvojice korespondujících bodů (X, X') nám do vzniklé soustavy přispívá dvěma rovnicemi, takže pro vyřešení soustavy a nalezení projektivní matice musíme určit alespoň čtyři body, u nichž známe jak obrazové, tak reálné souřadnice.

V aplikaci je pro toto nastavení určen takzvaný kalibrační panel. Uživatel si v něm může následně načíst snímek pomocí něhož bude provedeno seřízení. Aplikace je nastavena tak, že je třeba v obraze vyznačit všechny vrcholy obdélníkové oblasti ležící v rovině podlahy, čímž získáme obrazové souřadnice. Místo toho, aby uživatel zadával i reálné souřadnice těchto označených bodů, stačí aby uvedl pouze rozměry obdélníkové oblasti. V našem případě není volba počátku souřadného systému nijak omezena, protože nemá vliv na pozdější analýzu. Proto je jako počátek jednoduše označen jeden z vrcholů a jelikož jsou jeho sousedící strany navzájem kolmé, můžeme je použít pro určování směru souřadných os. Reálné souřadnice označených bodů vztažených k určenému počátku mají následující tvar:

$$a_1 = (0, 0);\ a_2 = (width, 0);\ a_3 = (width, height);\ a_4 = (0, height) \tag{41}$$

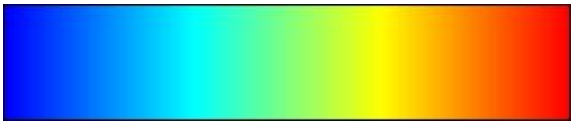
Kde $width$ určuje reálnou šířku a $height$ reálnou výšku označené obdélníkové oblasti.

Při znalosti takovéto transformace již můžeme jednoduše vypočítat reálné pozice obrazových bodů trajektorie. Můžeme tedy také zjistit jejich reálné vzdálenosti podle vzorce 35. Jelikož máme také zajištěn pravidelný interval během snímkování, zhodnocení rychlosti tak již není problémem.

Pro konečnou analýzu nám již zbývá jen určit křivost trajektorie. Zde ovšem nemůžeme použít standardní vzorce, poněvadž nemáme k dispozici spojitou křivku, ale prakticky jen posloupnost bodů, kterými je trajektorie určena. Proto je zde využita metoda, kdy pro každý bod $X_i = (x_i, y_i)$, ležící na trajektorii, se určí jeho nejbližší sousedé X_{i-1} a X_{i+1}. Těmito třemi body se následně proloží kružnice, která definuje jakousi oskulační kružnici pro bod X_i. Z následného výpočtu jejího poloměru r_i pak můžeme vyjádřit křivost $k_i = 1/r_i$. Přičemž pro výpočet poloměru kružnice se využívá následujících vztahů.

$$
\begin{aligned}
a &= \sqrt{(x_{i-1} - x_i)^2 + (y_{i-1} - y_i)^2} \\
b &= \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \\
c &= \sqrt{(x_{i-1} - x_{i+1})^2 + (y_{i-1} - y_{i+1})^2} \\
T &= \tfrac{1}{4}\sqrt{(a+b+c)(a-b+c)(a+b-c)(-a+b+c)} \\
r &= \tfrac{abc}{4T}
\end{aligned} \tag{42}
$$

Pro zobrazení informací o rychlosti a neobvyklé změně směru je ve výsledné aplikaci využito barevné stupnice znázorněné na obrázku 25. Díky ní sice nepoznáme přesné hodnoty vypočítaných informací, ale lépe nás upozorní na neobvyklé chování sledované osoby ve scéně. Jednotlivé změřené veličiny jsou zobrazovány separátně a to tak, že rychlost pohybu je vyjádřena barvou trajektorie a míra změny směry pohybu je vyjádřena barvou detekčního okna. Přičemž k hodnocení míry změny směru je použit klouzavý exponenciální průměr 1. Ten nám na jednu stranu zajistí zanedbání malých výkyvů a na straně druhé nám dovolí uchovat informaci o velké změně po delší časový úsek.



Obrázek 25: Barevná stupnice použita pro vyjádření hodnot rychlosti pohybu a křivosti trajektorie.

5 Implementace

V této části bych se chtěl alespoň okrajově zmínit o konkrétní struktuře implementace celé aplikace. Budou zde stručně popsány základní vytvořené třídy, kde bude zmíněn jak jejich účel, tak i návaznost na ostatní třídy.

Pro tvorbu této aplikace jsme zvolil programovací jazyk C++ a jako vývojové prostředí jsem používal Microsoft Visual Studio 2012. Jako hlavní opěrný bod byla zvolena knihovna OpenCV 2.4.10 (Open-source Computer Vision). Pro tvorbu uživatelského rozhraní jsem využil aplikační toolkit Qt.

OpenCV je multiplatformní knihovna napsána v jazyce C/C++. Její zaměření se soustřeďuje hlavně na oblast zpracování obrazu v reálném čase. Licenční podmínky pro tuto knihovnu dovolují volné použití jak pro akademické, tak i komerční aplikace. Jak již bylo dříve zmíněno, pro naši aplikaci byla použita verze 2.4.10, která byla zkompilována s podporou CUDA, pro rychlejší výpočty za využití výkonu grafické karty.

Qt je aplikační toolkit pro snadné a efektivní vytváření aplikací s grafickým uživatelským rozhraním. Jde o multiplatformní nástroj, jehož nativním jazykem je C++. Konkrétně při mé práci jsem využil verzi 5.1.1 a její plugin pro Microsoft Visual Studio 2012.

Níže již jsou konkrétně uvedeny některé významné třídy, které byly pro tuto aplikaci napsány. Jde o hlavní třídy aplikace obsahující logiku pro detekci, sledování trajektorií a jejich následnou analýzu.

5.1 HOGDetector

Jde o jakousi hlavní třídu, která přímo komunikuje s grafickým uživatelským rozhraním. V první fázi se zde provede inicializace a nastavení všech potřebných parametrů pro běh procesu. Další fáze již probíhá iterativně. Nejdříve je instanci této třídy předán jeden snímek celé sekvence, který je zpracován a doplněn o potřebné informace. Poté již může být jednoduše vrácen uživatelskému rozhraní, které se postará o jeho korektní zobrazení. V kompetencích této třídy je, získat snímek a pomocí HOG deskriptoru a SVM klasifikátoru zjistit všechny detekované oblasti. Ty jsou následně předány pro další vyhodnocování třídě PedestrianController. Abychom získali lepší výsledky, je zde také provedená dodatečná filtrace detekovaných oblastí. Samotná funkcionalita grupování se totiž v některých případech ukázala jako nedostatečná, takže proto vznikl tento dodatečný filtr.

5.2 PedestrianController

Instance této třídy představuje takzvaný sprácovský objekt. Třída HOGDetector představovala jakousi statickou složku, kde jen vyhodnocovala jednotlivé snímky nezávisle na sobě. Třída PedestrianController však zajišťuje potřebnou dynamičnost a dokáže tak vyhodnocovat i informace z předešlých stavů. Vstupními argumenty pro instanci této třídy tvoří aktuální snímek a detekované oblasti, které jsme dostali pomocí předešlé třídy. Výstupem je pak aktuální snímek doplněný o požadované informace. Hlavní funkcí je

spravovat informace o všech osobách ve scéně. Tato množina osob je vyjádřena jako seznam objektů třídy Pedestrian. Prvním krokem třídy PedestrianController je tedy získání a analýza detekovaných oblastí, kdy se například vypočítá barevné složení objektu obsaženého uvnitř každé oblasti. Druhým krokem je vytvoření tabulky penalizací, jejíž hodnoty jsou vypočítány pomocí evaluační funkce, která právě porovnává informace o osobách s informacemi o detekovaných oblastech. Dalším krokem je nalezení vhodných dvojic a aktualizace parametrů. Tento postup je názorně popsán v kapitole 4.2. Posledním krokem je doplnění aktuálních informací o trajektoriích osob do získaného snímku. Ten je poté poslán zpět do objektu třídy HOGDetector.

5.3 Pedestrian

Jak již název napovídá, instance třídy Pedestrian vyjadřuje jednu osobu vyskytující se ve scéně. Pro každý takovýto objekt jsou příznačné jak atributové data jako je třeba pozice, rozměry nebo barevné složení, tak i Kalmanův filtr, který popisuje jeho pohyb ve scéně. Jestliže se pro danou osobu nalezne vhodné detekční okno, dojde k aktualizaci popisných parametrů. Kromě aktuálních atributových dat, si tato instance uchovává i pozice obrazovýh bodů určujících trajektorii. Pozice těchto bodů budou hrát velkou roli pro následující třídu.

5.4 TrajectoryMapper

Objekt vytvořený pomocí této třídy má dvě hlavní využití. Prvním z nich je transformace souřadnic obrazových bodů trajektorie do roviny, která představuje podlahu. Tím získáme náhled na skutečné pozice jednotlivých bodů trajektorie a dokážeme tak přesněji určit rychlost a změnu směru pohybu objektu. V první fázi je však třeba zajistit výpočet příslušné transformační matice. Toho je docíleno pomocí kroku kalibrace zmíněném v podkapitole 4.3. Pak již jednoduše ke každému zvolenému bodu můžeme vypočítat jeho souřadnice v rovině podlahy. Druhé využití na tento proces navazuje. Jelikož už víme reálnou polohu bodů, musíme tyto informace vyhodnotit. Instance této třídy tudíž analyzuje reálné body trajektorie a pomocí barevné stupnice určí hodnoty rychlosti či křivosti. Tyto informace jsou poté zaneseny do upraveného aktuálního snímku.

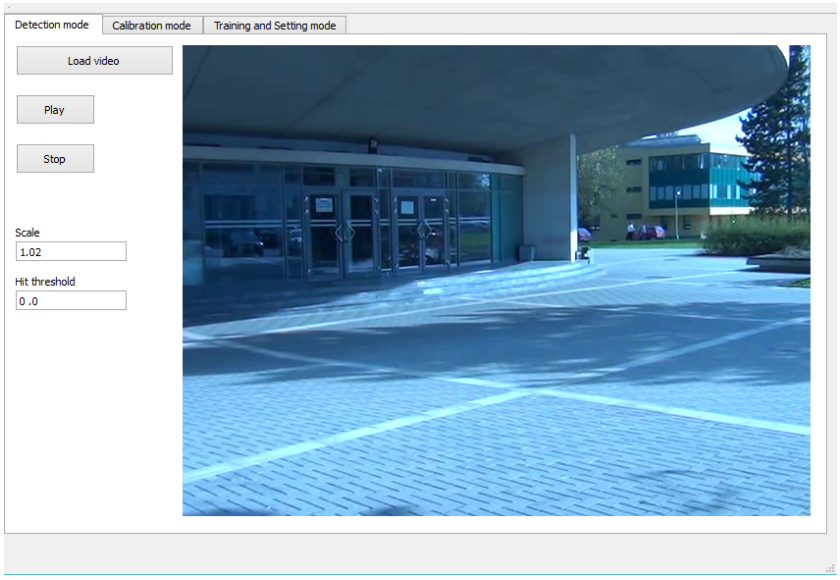
5.5 Player

Tato třída se již stará o reálné zobrazování dat. Nejprve je načteno video, jež bude použito pro analýzu. Při spuštění přehrávače se pak postupně získávají jednotlivé snímky, které jsou předávány objektu třídy HOGDetector. Po analýze je každý upravený snímek předán zpět přehrávači, který jej přizpůsobí na potřebou velikost a zobrazí na grafické uživatelské rozhraní. Abychom však mohli zajistit možnost ovládání aplikace za běhu analýzy, je pro tento přehrávač vytvořeno vlastní vlákno.

GUI

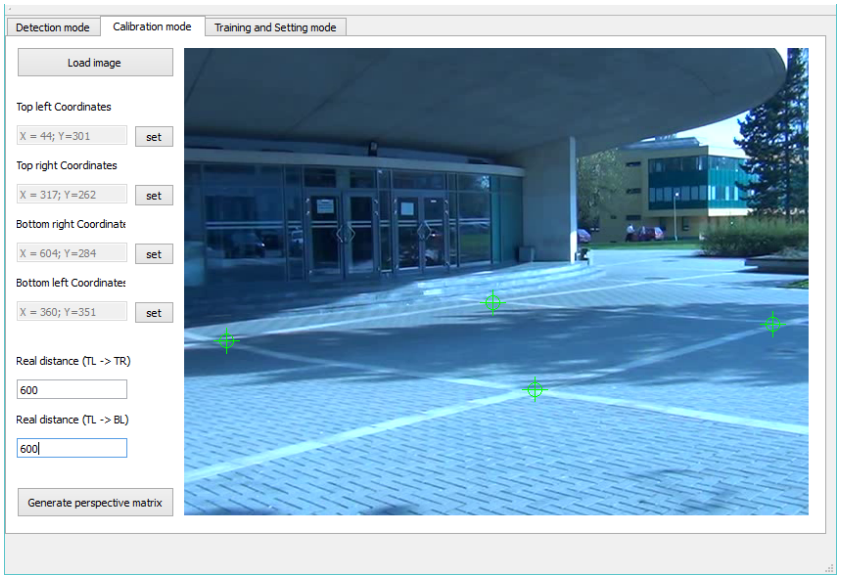
Dále ještě alespoň okrajově zmíníme celkovou vizualizaci. Pro snadnější manipulaci s aplikací bylo pomocí Qt toolkitu vytvořeno jednoduché grafické uživatelské rozhraní. Program jako celek je rozdělen do tří oken, kde každé má svou oblast využití. Náhledy na jednotlivé části jsou k vidění níže.

Prvním z nich je okno takzvaného "Detekčního módu". Jak již název napovídá, jedná se o okno, kde jsou vizualizovány výsledky získané prostřednictvím procesu sledování osob a vyhodnocování trajektorie. Zde je uživateli umožněno načíst video soubor, jež má být analyzován a pomocí tlačítek pro spuštění, pozastavení nebo zastavení poté řídit jeho průběh. Kromě tohoto ovládání je zde ještě možné nastavit dva parametry, jež ovlivňují detekci osob. Těmi jsou parametry $hitThreshold$ a $scale$, jejichž účel je vysvětlen na konci podkapitoly 4.1.



Obrázek 26: Ukázka vzhledu okna pro detekci.

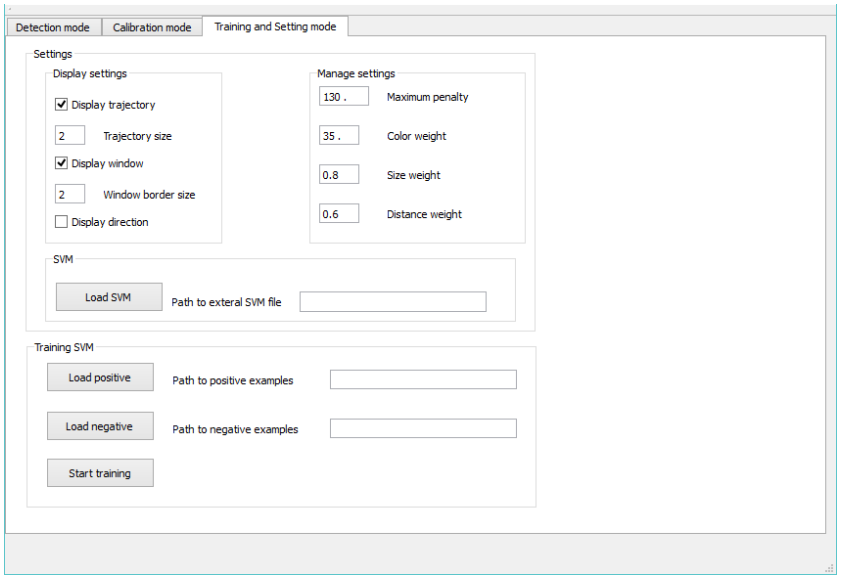
Druhé okno je nazváno jako "Kalibrační mód"a slouží pro stanovení projektivní matice, která zajišťuje transformaci obrazových bodů do roviny podlahy. Tento krok je možné vynechat, ovšem to bude znamenat, že analýza rychlosti pohybu a křivosti trajektorie bude vztažena pouze k obrazovým souřadnicím. Nebude tak možno vyhodnocovat reálnou rychlost objektu v obraze, ani jeho změny směru. Detailnější popis principu této kalibrace lze nalézt v podkapitole 4.3.



Obrázek 27: Ukázka vzhledu okna pro kalibraci.

Poslední okno nazvané jako "Mód pro trénování a nastavení"se zaměřuje na dodatečnou funkčnost této aplikace. Toto okno je rozděleno do dvou základních sekcí, které určují hlavní odvětví. První část dává uživateli možnost nastavit jednak zobrazované informace ve výstupních snímcích a také dovoluje určit parametry, jež se používají pro vyhodnocování vhodné dvojice detekované osoby a detekovaného okna. Pro lepší pochopení významu jednotlivých parametrů je vhodné věnovat pozornost podkapitole 4.2. Déle je zde možné nastavit používaný klasifikátor SVM, jež je tvořen souborem ve formátu XML. K tomuto aspektu se následně váže druhá část okna, která je věnována možnosti vytvořit a uložit si vlastní klasifikátor. Uživatel jen jednoduše zvolí složky, kde se nacházejí obrázky pozitivních a negativních případů o velikosti 64x128 pixelů a ve formátu PNG. Následně je možné spustit proces učení, který povede k vytvoření příslušného XML souboru.

Pro správné nastavení a kalibraci detektoru by mělo být dodrženo postupu, kdy se nejprve nastaví hodnoty všech parametrů včetně $scale$ a $hitThreshold$. Následně by měl uživatel zvolit požadované video a v posledním kroku by měl provést kalibraci aplikace na zvolenou scénu.



Obrázek 28: Ukázka vzhledu okna pro nastavení a učení SVM.

6 Testování

Tato kapitola bude zaměřena na testování výsledné aplikace. Zohledněna bude jak rychlost výpočtů, tak i celková úspěšnost vyhodnocování. Pro tyto účely byly vytvořeny zkušební videosekvence pořízené v areálu VŠB-TU Ostrava. Pro snímání scény byla využita osobní digitální videokamera SONY HDR-CX130, jež byla upevněna na stativu. Pro implementaci a běh aplikace byl využit osobní notebook s procesorem Intel ®Core ™i7-4700HQ, 2,4GHz operační pamětí 8 GB RAM a grafickou kartou GeForce GT 750M. Grafická karta je uvedena, poněvadž část výpočtů se provádí pomocí technologie CUDA.

V úvodu je také třeba zdůraznit, že pro korektní vyhodnocování je zapotřebí dodržet základní předpoklady. Jelikož jde o aplikaci sledující a vyhodnocující trajektorii osob v obraze, je třeba zajistit, aby byly veškeré videosekvence pořízeny jako statické. Dalším předpokladem je, že známe pozice kalibračních bodů a máme také zjištěny informace o jejich vzájemné vzdálenosti. Tento předpoklad je důležitý pro měření rychlosti pohybu a křivosti trajektorie. Uživatel sice nemusí kalibraci provést, ale v tom případě však budou analytické výpočty vztaženy pouze na obrazové pozice bodů trajektorie. Dodatečné získání korektního výstupu však může uživatel ovlivnit i množstvím nastavitelných parametrů, které se v aplikaci vyskytují.

6.1 Rychlost algoritmu

Z pohledu rychlosti běhu aplikace byly provedeny celkem tři testy. Jejich cílem bylo poukázat na důležitost velikosti obrazu a také volby parametru $scale$ z pohledu výpočetní rychlosti programu. Prvním testem byla ověřována rychlost detekce HOG detektoru za pomocí výkonu grafické karty. Ve druhém testu byly pro zajímavost provedeny tytéž testy, ovšem v tomto případě k výpočtu nebyla použita grafická karta, ale pouze procesor. Třetí test může být použit pro porovnání s výsledky prvního testu. Byly zde zkoumány časové intervaly výpočtu celého algoritmu i s trasováním a analýzou trajektorie. Výsledky všech tří testů jsou k nahlédnutí níže.

scale / rozlišení	1,01	1,02	1,05	1,08	1,12
1024x768 px	693 ms	453 ms	203 ms	125 ms	106 ms
800x600 px	422 ms	265 ms	112 ms	78 ms	63 ms
640x480 px	266 ms	172 ms	78 ms	47 ms	31 ms
320x240 px	63 ms	32 ms	16 ms	5 ms	3 ms

Tabulka 1: Tabulka ukazující časový interval potřebný pro výpočet jednoho snímku. Testován je HOG detektor s podporou grafické karty.

scale / rozlišení	1,01	1,02	1,05	1,08	1,12
1024x768 px	3421 ms	2492 ms	1452 ms	940 ms	859 ms
800x600 px	1999 ms	1350 ms	778 ms	527 ms	478 ms
640x480 px	1156 ms	731 ms	379 ms	297 ms	203 ms
320x240 px	203 ms	121 ms	74 ms	56 ms	33 ms

Tabulka 2: Tabulka ukazující časový interval potřebný pro výpočet jednoho snímku. Testován je HOG detektor, kdy všechny výpočty obstarává procesor.

scale / rozlišení	1,01	1,02	1,05	1,08	1,12
1024x768 px	705 ms	454 ms	204 ms	125 ms	106 ms
800x600 px	425 ms	270 ms	121 ms	78 ms	61 ms
640x480 px	266 ms	174 ms	78 ms	47 ms	32 ms
320x240 px	63 ms	32 ms	17 ms	4 ms	2 ms

Tabulka 3: Tabulka ukazující časový interval potřebný pro výpočet jednoho snímku. Testován je celý algoritmus pro detekci a sledování osob a následnou analýzou trajektorie.

6.2 Testování detekce

Dalším aspektem testování byla výkonnost detekčního algoritmu. Pro tyto účely byla vytvořena krátká testovací videosekvence zobrazující osoby, které se potupně vzdalují od kamery. Cílem této analýzy bylo zkoumat vliv parametrů $scale$ a $hitThreshold$ na výslednou úspěšnost detektoru. Testovací sekvence byla pořízena v rozlišení $640x480$ pixelů a obsahovala 301 snímků. V těchto snímcích mělo být nalezeno 634 správných detekcí, ovšem žádné z nastavení nedosáhlo dokonalého výsledku. V přiložené tabulce jsou zobrazeny jednotlivé výsledky analýzy úspěšnosti pro vybrané nastavení $scale$ a $hitThreshold$. U každé možnosti jsou uvedeny dvě hodnoty. První z nich udává počet $FalsePositive$ detekcí, neboli případů, kdy je označena oblast, jež neobsahuje lidskou postavu. Do těchto případů jsou započítány i detekce, které přesně nevystihují celou postavu, ale například pouze její ruku. Druhá hodnota udává počet $FalseNegative$ případů, což jsou situace, kdy detektor neoznačí lidskou postavu. Pod tabulkou jsou následně umístěny ukázky snímků z testovací sekvence.

hitThreshold	scale	FalsePositive	FalseNegative
0	1.01	27	18
0	1.015	7	22
0	1.02	7	26
0	1.04	1	47
0	1.06	3	64
0	1.1	0	102
0.2	1.01	14	26
0.2	1.015	4	30
0.2	1.02	3	33
0.2	1.04	1	74
-0.2	1.01	44	13
-0.2	1.015	12	16
-0.2	1.02	11	17
-0.2	1.04	5	29
-0.2	1.06	6	44

Tabulka 4: Tabulka reprezentující úspěšnost detekce pro konkrétní nastavení parametrů $scale$ a $hitThreshold$.

Obrázek 29: Ukázka snímků použitá při testování. Vlevo je zobrazen první a napravo poslední snímek sekvence.

Jak lze vidět z tabulky 4, nacházely se v sekvenci snímky, které zapříčinily selhání detektoru. Prvním z nich může být takzvaná $FalsePositive$ detekce, kdy detektor pozitivně vyhodnotí oblasti, které nepředstavují lidskou osobu. Tyto případy se ovšem dají jen stěží předem identifikovat. Jde jednoduše o oblasti, které svou siluetou a rozpoložením hran mohou navozovat dojem lidské postavy. Může jít například o některé stromy, lampy, koše nebo jen nevhodné odrazy na skle. Ukázalo se, že tyto nechtěné odezvy de-

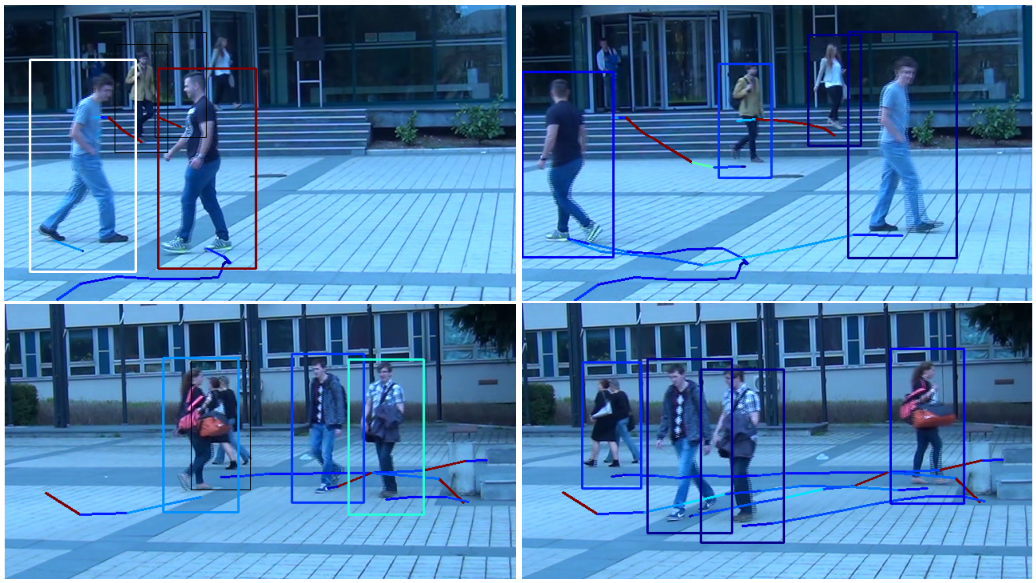
tektoru může mít vliv parametr $scale$, ovšem pro vhodnější filtraci by se doporučovalo použít zvýšení hodnoty $hitThreshold$.

Opačnou situaci představují takzvané $FalseNegative$ detekce. Při nich naopak nastává situace, kdy v oblasti je viditelná osoba, ale detektor na ní nemá pozitivní odezvu. Testováním se ukázalo, že tyto situace nejčastěji nastávají ve chvílích, kdy je osoba již ve velké vzdálenosti od kamery. Z dřívějšího popisu je zřejmé, že minimální velikost detekované oblasti je dána nastavením velikosti okna pro HOG deskriptor, což je v našem případě 64x128 pixelů. Jestliže tedy velikost osoby v obraze dosáhne těchto mezních rozměrů, je už detekce velmi obtížná a často i nedosažitelná. Jako částečné řešení pro detekci těchto mezních rozměrů můžeme snížit hodnotu parametru $scale$, který ovlivňuje jaké velikosti oblastí se budou analyzovat.

Dále je třeba ještě zmínit, že SVM klasifikátor nabízený pomocí OpenCV je naučen pro detekování celých lidských postav. Můžeme tudíž zaznamenávat selhávání detekce pro osoby, které zasahují do obrazu jen z části. V praxi může jít například o situace, kdy člověk prochází velmi blízko kamery. V těchto případech tedy detekce nebude úspěšná.

6.3 Testování sledovacího algoritmu

Sledovací algoritmus je další důležitou součástí této aplikace. Jeho hlavním úkolem je hledat v sekvencích snímků objekty, které si reálně odpovídají a sledovat trajektorii jejich pohybu. Pro ověření správné funkčnosti byly vytvořeny testovací sekvence, které tuto funkcionalitu ověřovaly. Záměrem bylo vybrat úseky, kde se vzájemně míjelo více lidí, a na nich sledovat úspěšnost sledovacího algoritmu. V mnoha případech se ukázalo, že je dosaženo velmi uspokojivých výsledcích, kdy si algoritmus dokázal poradit i se situacemi, které jsou znázorněny níže.



Obrázek 30: Ukázka situací, pro které byl sledovací algoritmus vyhovující.

Pro lepší vyhodnocení úspěšnosti byl algoritmus testován na několika sekvencích. V jejich průběhu se postupně zaznamenávaly jak vyhovující případy sledovacího algoritmu, tak i jeho selhání. Dosažené výsledky byly sepsány do tabulky, která je přiložena níže.

název sekvence	vyhovující sledování	problémy při sledování	úspěšnost
S1	16	4	80%
S2	12	3	80%
S3	18	5	78%
S4	22	4	85%
S5	20	3	87%
celkem	88	19	82%

Tabulka 5: Tabulka znázorňující úspěšnost sledovacího algoritmu při testovacích sekvencích.

Sekvence byly testovací a záměrně obsahovaly poměrně velký počet pasáží, kde se vyskytovalo více chodců najednou. Ukázalo se, že v jednoduchých situacích pracuje algoritmus více než dobře, i když i zde se vyskytly problémy, jejichž příčinou však byly většinou chyby v detekci. Naprostá většina zaznamenaných problémů však vznikala ve scéně s více chodci. Dále tedy budou uvedeny druhy zaznamenaných chyb i s vysvětlením příčiny vzniku.

Problémy

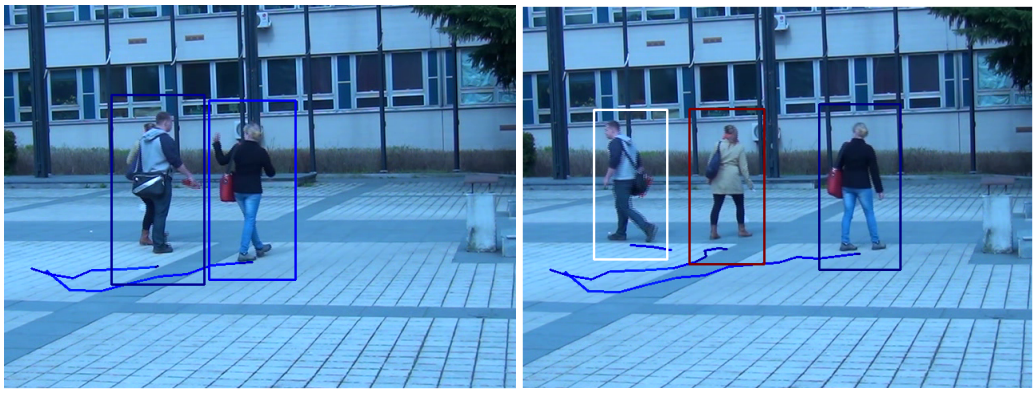
Jako první bychom mohli uvést chybu, kdy je sledovaná osoba krátce po zařazení do scény zastíněna. Sledovací okno tedy standardně předpovídá pravděpodobný směr pohybu, avšak ten vůbec neodpovídá skutečnému pohybu. V této situaci je příčinou právě to, že viditelnost osoby bylo dosaženo jen po velmi krátkou dobu. Kalmanův filtr v té chvíli ještě není ustálen, takže z toho důvodu je predikce velmi nepřesná.

Dalším problémem navazujícím na Kalmanův filtr je situace, kdy jdou osoby v těsné blízkosti vedle sebe. Algoritmus například nejprve detekuje obě osoby, ale po chvíli se ocitne jedna z těchto osob v zákrytu. Tím se ztratí přímá detekce a objekt je predikován pouze pomocí Kalmanova filtru. Stačí tedy, aby obě osoby změnily směr a jelikož program nemá přímou viditelnost na zakrytou osobu, bude její pozici detekovat na zcela nesprávném místě. Tomuto problému lze však jen stěží zabránit, poněvadž právě na viditelnosti osob je celý program založen, takže jestliže se tato informace ztratí, je sledování velmi obtížné.

Velkou skupinou problémů se však staly situace, kdy na jednu osobu byly chybně připisovány dvě sledovací trajektorie. Osoba v těchto případech měla správně přiřazeno jedno detekční okno o odpovídajících rozměrech, ovšem mohlo se zde vyskytnout i menší okno sledující jen malou část těla. Jako příčina problému se ukázalo již dříve zmíněné chování detektoru, kdy pro jednu osobu může být velikost detekčního okna velmi nestabilní. V této situaci tedy v sekvenci dojde k náhlému výkyvu velikosti, se kterým si

přiřazovací algoritmus neporadí. Přiřazování totiž pracuje na základě porovnávání velikosti, barvy a pozice referenčního bodu. Jestliže tedy dojde k velkému výkyvu detekce, algoritmus u něj vypočítá velkou penalizaci, takže je zde vytvořen nový objekt scény. Řešením této situace by pak bylo zvětšení maximální přípustné penalizace, to by ovšem mohlo vést zase k problémům. Mohlo by se stát, že přiřazení by proběhlo i pro velmi odlišné osoby.

Dalším problémem je pak velký shluk osob na malém prostoru, které se postupně částečně zakrývají. Stává se tedy, že detektor v každé části sekvence detekuje jinou osobu, ovšem přiřazovací algoritmus v těchto případech nemůže přesně určit, jestli nově nalezená detekce patří stejné osobě nebo ne. Vzdálenosti i velikosti jsou totiž v těchto případech velmi podobné a tak algoritmus tyto různé osoby ve shluku často prohlásí jako stejné osoby. Řešením by zde mohlo být snížení maximální přípustné penalizace, nebo zvýšení váhy u aspektu barevného složení objektu. To by ovšem mohlo vyprodukovat chyby jiného druhu. Ukázku tohoto selhání můžete vidět na snímcích 31. V tomto případě se dvě osoby zcela zakrývají, takže je algoritmus vyhodnotí jako jednu osobu. V další části však dojde k prudké změně směru první osoby, takže jsou již obě postavy viditelné. Kalmanův filtr však nestačí na tuto změnu pohybu reagovat a nesprávně přiřadí původní trajektorii nově odkryté osobě.



Obrázek 31: Ukázka situace selhání sledovacího algoritmu.

Nedostatky

Při pohledu na přiložené ukázky 30, detekční okna byla v těchto případech správně přiřazována odpovídajícím reálným objektům i přes složitost situace. Ovšem z pohledu zaznamenávání a analýzy trajektorie jsou zde viditelné jisté nedostatky. Lze si například všimnout, že zbarvení trajektorie, které má představovat rychlost pohybu objektu v některých případech neodpovídá reálné situaci. Červená barva znázorňuje rychlý pohyb, modrá naopak pomalý. Špatně vyhodnocena je například rychlost pohybu osob jdoucích po schodech. Zde je příčinou algoritmus, jež provádí transformaci obrazových bodů do roviny podlahy. Pro zmíněné osoby neplatí, že se pohybují v rovině podlahy a proto je v těchto případech transformace i výsledná vizualizace informací velmi zkreslená.

Dalším nedostatkem je také již zmíněný fakt, že detekční okno velmi lehce a často mění svou velikost. Důsledkem toho je, že trajektorie zobrazující pohyb osoby vykazuje šum. To má za následek nepřesnosti ve výpočtech rychlosti a křivosti pohybu. Ve výsledku se tedy i při pomalém pohybu mohou vyskytovat úseky, které vykazují zvýšenou rychlost, což se projeví jako červené úseky na jinak modré trajektorii.

Na závěr této kapitoly je však třeba říct, že mnoho těchto vyhodnocování je možné silně ovlivnit pomocí nastavení prahu a vah u jednotlivých porovnávacích aspektů jak je uvedeno v 4.2. Pomocí těchto parametrů jde výsledné vyhodnocování ještě dodatečně vyladit, takže může dojít k omezení vzniku výše zmíněných chyb. Nebo na druhé straně může tato činnost vést v navýšení nepřesností.

7 Závěr

Tato práce byla zaměřena na vytvoření aplikace pro analýzu videosekvencí obsahujících pohybující se lidské postavy. Analýza byla zaměřena na vyhodnocování rychlosti pohybu a také míry změny směru.

V první části byly stručně popsány aktuálně používané metody pro detekci lidí v obraze. Analyzovány nebyly všechny techniky, ale pouze jejich úzký výběr. Po prozkoumání těchto technik byl pro výslednou aplikaci zvolen HOG detektor.

Ve druhé části byly podrobně popsány hlavní metody, kterých aplikace využívá. Konkrétně šlo o HOG deskriptor, SVM klasifikátor a Kalmanův filtr. Porozumění těmto algoritmům bylo důležité při celém návrhu řešení.

Třetí část byla věnována na celý proces detekování osob až po vyhodnocování trajektorie. Zde byly poznatky z minulé teoretické části názorně převedeny do souvislostí s výslednou aplikací. Prostor zde byl věnován jak popisu jednotlivých navazujících částí, tak i vysvětlením vlivu parametrů, které se při sestavování vyskytovaly.

Následná čtvrtá část byla věnována konkrétní implementaci. Zmíněny zde byly použité technologie, hardware pro vývoj a testování. Ovšem část byla také věnována popisu funkcionalit hlavních tříd pro výslednou aplikaci. Krátce zde bylo popsané i grafické uživatelské rozhraní.

Poslední část již byla věnována testování aplikace. Byly zde ověřovány dva aspekty. Prvním z nich byla rychlost aplikace a způsoby jejího ovlivnění. Druhý aspekt tvořila úspěšnost celého procesu od detekce po analýzu trajektorií. Všechny získané poznatky zde byly sepsány včetně zdůraznění některých nedostatků algoritmu.

Na datovém nosiči jsou poté přiloženy některé zkušební sekvence, na kterých byla aplikace testována. Je zde přiložen i projekt pro zmiňovanou aplikaci. Jak již bylo řečeno, pro její vývoj byly použity knihovny Qt 5.1.1 a také zkompilované knihovny OpenCV 2.4.10 s podporou CUDA. Tyto aspekty jsou tedy požadavky, které by měly být splněny pro správné spuštění aplikace. Je třeba i poznamenat, že software byl vyvíjen jako 64-bitová aplikace.

Celkově by se dalo prohlásit, že cíle práce byly do jisté míry splněny. Úspěšně byla vytvořena aplikace, která dokáže vě většině případů správně detekovat a sledovat trajektorii pohybujících se osob ve scéně. Vyhodnocování trajektorie je však do jisté míry závislé na přesnosti kalibrace a odezvy detektoru. Jestliže není kalibrace provedena řádně, lze lehce dostat velké zkreslení ve výsledcích. Druhým aspektem je přesnost detekce, kterou již nelze tak snadno ovlivnit. Velikostní a poziční odchylky detekovaných oblastí totiž mohou lehce vést ke zkreslení trajektorie a následnému znehodnocení poskytovaných informací.

Martin Gold

8 Reference

[1] WREN, C.R, A. AZARBAYEJANI, T. DARRELL a A.P PENTLAND. *Pfinder: real-time tracking of the human body.* Pattern Analysis and Machine Intelligence, IEEE Transactions on [online]. 1997, roč. 19, č. 7, s. 780-785 [cit. 2015-04-28]. DOI: 10.1109/34.598236. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=598236&isnumber=13123

[2] STAUFFER, Chris a W.E.L. GRIMSON. *Adaptive background mixture models for real-time tracking* In: Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. [online]. 2. vyd., 1999 [cit. 2015-04-28]. ISBN 0-7695-0149-4ISSN 1063-6919. DOI: 10.1109/CVPR.1999.784637. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=784637&isnumber=17024

[3] BELEZNAI, Csaba, Bernhard FRÜHSTÜCK, Horst BISCHOF. *Human Tracking by Fast Mean Shift Mode Seeking.* Journal of Multimedia [online]. 2006-04-01, roč. 1, č. 1, s. - [cit. 2015-04-28]. DOI: 10.4304/jmm.1.1.1-8. Dostupné z: http://ojs.academypublisher.com/index.php/jmm/article/view/233

[4] ZHOU, Jianpeng a Jack HOANG *Real Time Robust Human Detection and Tracking System.* In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops [online]. IEEE, 2005, s. 149-149 [cit. 2015-04-28]. ISBN 0-7695-2372-2. DOI: 10.1109/CVPR.2005.517. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1565316

[5] VIOLA, P., M.J. JONES a D. SNOW. *Detecting pedestrians using patterns of motion and appearance.* In: Proceedings Ninth IEEE International Conference on Computer Vision [online]. IEEE, 2003, 734-741 vol.2 [cit. 2015-04-28]. ISBN 0-7695-1950-4. DOI: 10.1109/ICCV.2003.1238422. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1238422

[6] DALAL, N. a B. TRIGGS. *Histograms of Oriented Gradients for Human Detection.*In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) [online]. IEEE, 2005, s. 886-893 [cit. 2015-04-28]. ISBN 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467360

[7] SOJKA, Eduard. *Digitální zpracování a analýza obrazů [online].* 1. vyd. Ostrava: VŠB - Technická univerzita, 2000, 133 s. [cit. 2015-04-29]. ISBN 80-707-8746-5. Dostupné z: http://mrl.cs.vsb.cz/people/sojka/dzo/digitalni_zpracovani_obrazu.pdf

[8] FARAGHER, Ramsey. *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes].* IEEE Signal Processing Magazine [online]. 2012, vol. 29, issue 5, s. 128-132 [cit. 2015-04-29]. DOI: 10.1109/MSP.2012.2203621. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6279585

[9] MANNING, Christopher D, Prabhakar RAGHAVAN a Hinrich SCHUZE. *Introduction to information retrieval [online].* New York: Cambridge University Press, 2008, xxi, 482 p. [cit. 2015-04-29]. ISBN 05-218-6571-9. Dostupné z: http://nlp.stanford.edu/IR-book/

[10] ČERNÍN, Jan. *Vizuální detekce osob v komerčních aplikacích.* Brno, 2012. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. KAREL HORÁK, Ph.D.

[11] Itseez. *Documentation OpenCV [online].* 2015 [cit. 2015-04-29]. Dostupné z: http://docs.opencv.org/

[12] THE QT COMPANY LTD. *Qt Documentation [online].* 2015 [cit. 2015-04-29]. Dostupné z: http://doc.qt.io/

A Datové DVD

Jako příloha k práci je zde přiloženo datové DVD obsahující potřebné soubory. Adresářová struktura je následující:

Text Zde je obsažen text diplomové práce.

Aplikace Zde jsou přiloženy zdrojové soubory aplikace.

Ukázky Zde se nachází některé ukázkové sekvence použité pro testování.