VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

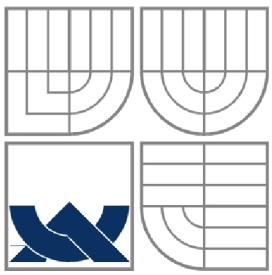
DETEKCE POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEO SEKVENCI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

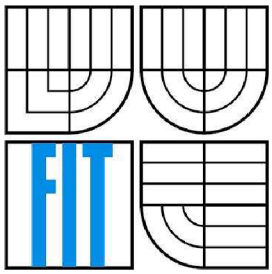
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ NĚMEC

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEO SEKVENCI

MOVING OBJECTS DETECTION IN VIDEO SEQUENCES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ NĚMEC

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2012

Abstrakt

Tato práce se zabývá metodami detekce osob a sledování objektů ve video sekvenci. Její součástí je také návrh a implementace systému, který provádí detekci a následné sledování hráčů v záznamu sportovního utkání, např. hokeje nebo basketbalu. Navržená aplikace používá kombinaci histogramu orientovaných gradientů a SVM (support vector machines) pro detekci hráčů v obraze. Pro sledování hráčů je použit částicový filtr. Celý systém je důkladně otestován a výsledky jsou uvedeny přehledně v grafech a tabulkách včetně slovního popisu.

Abstract

This thesis deals with methods for the detection of people and tracking objects in video sequences. An application for detection and tracking of players in video recordings of sport activities, e.g. hockey or basketball matches, is proposed and implemented. The designed application uses the combination of histograms of oriented gradients and classification based on SVM (Support Vector Machines) for detecting players in the picture. Moreover, a particle filter is used for tracking detected players. The whole system was fully tested and the results are shown in the graphs and tables with verbal descriptions.

Klíčová slova

sledování objektů, detekce hráčů, histogram orientovaných gradientů, částicový filtr, support vector machines

Keywords

object tracking, players detection, histogram of oriented gradients, particle filter, support vector machines

Citace

Němec Jiří: Detekce pohybujících se objektů ve video sekvenci, diplomová práce, Brno, FIT VUT v Brně, 2012

Detekce pohybujících se objektů ve video sekvenci

Prohlášení

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

.....
Jiří Němec
17.5.2012

Poděkování

Tímto děkuji vedoucímu práce za kvalitní vedení a pomoc při vypracování diplomové práce.

Obsah

1 Úvod

S rozvojem výpočetní techniky výrazně roste oblast, kde ji lze využívat. Jednou z nich je i počítačové vidění, s jehož pomocí se počítače snaží napodobit lidské vidění. Oproti lidskému vnímání má ale počítač ztíženou práci, protože pracuje s omezeným množstvím informací.

Díky videotechnice se získává a hromadí velké množství dat. Jejich zpracování by bylo neefektivní a drahé, a proto se nabízí možnost je zpracovávat automatizovaně pomocí výpočetní techniky.

Jednou z oblastí zpracování vizuálních dat je i detekce pohybujících se objektů ve video sekvenci s následným sledováním těchto objektů, což je i cílem této diplomové práce. Tato problematika má využití v nejrůznějších oblastech, např. ve vojenství, v zabezpečovacích systémech nebo v dopravních systémech. Často jsou tyto systémy používany pro obchodní domy, kdy slouží k sledování vytíženosti jednotlivých oddělení. Mohou být použity pro zabezpečení objektů, detekci překážek na vozovce, sledování dopravy nebo navádění střel. Vzhledem k velkému potencionálnímu využití je tato oblast perspektivní a rychle se rozvíjí.

Konkrétním zaměřením této práce je analýza pohybu v záznamech různých sportovních utkání, například hokeje, fotbalu, basketbalu, kdy se provádí detekce a rozpoznávání objektů, tj. hráčů, kteří se poté sledují. Pro jejich detekci v obraze se používá kombinace histogramu orientovaných gradientů a SVM (support vector machines). K účelu sledování je použit částicový filtr.

První část je věnována teoretickému úvodu, kde jsou popsány metody pro sledování objektů ve video sekvenci, jsou zde přiblíženy i klasifikační metody, které mají za úkol rozhodnout, zda se jedná či nejedná o hráče, a také jsou zde rozebrány možnosti reprezentace objektů. Další kapitola přibližuje nejvýznamnější přístupy pro detekci osob a je nastíněn postup pro detekci hřiště. Čtvrtá kapitola se zabývá popisem a návrhem programu včetně případných problémů, které nastávají. Pátá kapitola popisuje implementaci programu z hlediska tříd a použitých technologií. Následující kapitola je zaměřena na testování a experimenty s programem. Testy se specializují hlavně na úspěšnost detekce a sledování. Kapitola obsahuje také testování z hlediska časových nároků.

Text práce navazuje na semestrální projekt, ze kterého přebírá a upravuje část teorie kapitol 2, 3 a 4. Zároveň byla využita již vytvořená implementace sledovacího algoritmu.

2 Zpracování obrazu

Cílem této kapitoly je seznámit čtenáře s problematikou zpracování obrazu, což zahrnuje sledování (tracking), reprezentaci a klasifikaci daných objektů.

2.1 Sledování objektů

Sledování objektů (tracking) je důležitou činností, která spadá do oboru počítačového vidění a slouží k určení pozice cíle ve videu (v několika po sobě následujících snímcích). Lidský mozek tento problém vyhodnocuje zcela intuitivně, a proto je schopen snadno určit směr i rychlost pohybu. Počítač má naproti tomu velmi málo informací o daném objektu a celý obraz je reprezentován jen sadou pixelů.

Sledování objektů je založeno na stanovení vztahu mezi prvním výskytem objektu a jeho výskytem v následujících snímcích. Nebo známe-li pozici v čase *t-1* a chceme zjistit pozici v čase *t*. Sledování objektů se používá pro zjištění aktuální pozice sledovaného objektu v prostoru a určení informace o jeho výskytu v časovém intervalu. To znamená, že se určuje jeho trajektorie. Díky tomu můžeme dále určit například chování, pohyby, gesta, změny či nezvyklé projevy objektu.

Přehledně lze tuto problematiku shrnout do následujících bodů:

Vstup:

- pozice sledovaných objektů v minulém snímku
- aktuální snímek
- modely objektů

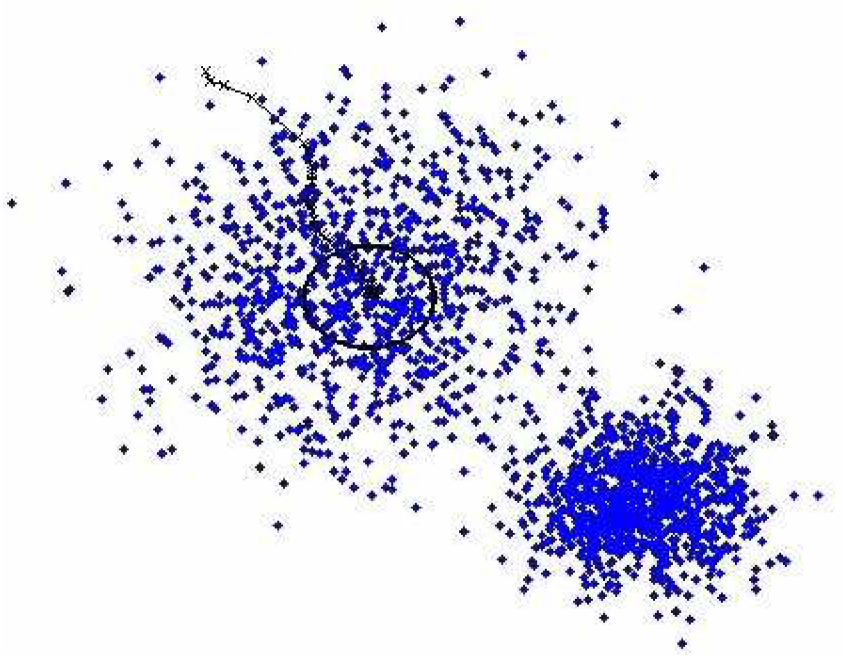
Výstup:

- poloha sledovaných objektů v aktuálním snímku
- ztracené objekty

Metod pro sledování objektů existuje celá řada, a proto je třeba brát v úvahu i další vlastnosti objektu, jeho pozadí a kameru, protože každá metoda má své výhody i nevýhody. V tomto případě, kdy se jedná o sledování hráčů, nelze opomenout například velikost hráčů, pohyb a zoomování kamery, mírné změny osvětlení nebo možné překrývání hráčů. Proto zde budou přehledně uvedeny nejznámější a nejpoužívanější metody.

2.1.1 Mean shift

Mean shift je univerzální algoritmus [3], který se používá v mnoha odvětvích vědy. Velmi často se využívá v počítačovém vidění, jako například při hledání modusu, segmentaci dat nebo ve shlukové analýze.



Obrázek 2.1: Ukázka hledání těžiště pomocí algoritmu Mean shift. Převzato z [4].

Mean shift je možné použít také ke sledování objektů [1, 2], které spadá do oblasti jádrového sledování (kernel-based metody). Tato metoda se vyznačuje nízkou výpočetní náročností a jednoduchostí. Mean shift pracuje většinou s barevnou reprezentací cíle (barevný histogram), kdy sledovaný objekt je vymodelován pomocí vhodného barevného prostoru. Potom se tedy nepracuje přímo s obrazem, ale s jeho modelem. Princip sledování vychází z názvu tohoto algoritmu a spočívá v hledání pomyslného středu (obrázek 2.1) neboli těžiště modelu daného sledovaného objektu, kdy se postupuje iteračně a skončí se, až se dosáhne dané přesnosti (aktuální posun středového bodu je menší než konstanta) nebo proběhne předem daný počet iterací.

2.1.2 Lucas-Kanede

Tato široce používaná diferenční metoda [5] optického toku byla vyvinuta Brucem D. Lucasem a Taeko Kanadem v roce 1981. Metoda vychází ze dvou po sobě jdoucích snímků a jejich diferencí. Využívá se zde pouze malých lokálních oblastí okolo význačných bodů, kdy se počítá s tím, že tok v okolí těchto bodů je konstantní v čase. Okolí daných bodů má tedy konstantní směr a rychlost toku (konstantní obrazovou rychlost). Díky tomuto předpokladu je možné sestavit rovnici:

$$Av = b. \tag{2.1}$$

A, v a b mají následující význam:

$$A = \begin{bmatrix} I_X(q_1) & I_Y(q_1) \\ I_X(q_2) & I_Y(q_2) \\ \vdots & \vdots \\ I_X(q_n) & I_Y(q_n) \end{bmatrix}, \quad v = \begin{bmatrix} V_X \\ V_Y \end{bmatrix} \text{ a } \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}. \tag{2.2}$$

q1,q2,...,qn jsou pixely uvnitř obrazu, *Ix* a *Iy* je derivace obrazu, I_t je derivace mezi obrazy za čas *t* a *v* je změna pohybu, kterou hledáme.

Pro nalezení optimálního řešení lze použít například metodu nejmenších čtverců a získáme následující rovnici:

$$\begin{bmatrix} V_X \\ V_Y \end{bmatrix} = \begin{bmatrix} \sum_i I_X(q_i)^2 & \sum_i I_X(q_i) I_Y(q_i) \\ \sum_i I_X(q_i) I_Y(q_i) & \sum_i I_Y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_X(q_i) I_Y(q_i) \\ -\sum_i I_X(q_i) I_Y(q_i) \end{bmatrix}. \tag{2.3}$$

Jelikož se pracuje s malými oblastmi okolo význačných bodů, rychlost algoritmu je velmi dobrá a metoda není tolik citlivá na šum v obraze. Na druhou stranu, jelikož je integrační okno malé, lze počítat pouze s malou změnou optického toku, protože metoda nezachytí rychlý či prudký pohyb. Tuto nevýhodu lze ovšem odstranit použitím pyramidové implementace Lucas-Kanadeho algoritmu, který postupně probíhá od nejvyššího patra pyramidy (nejnižší detaily) až k patrům nejnižším (nejvyšší detaily).

Základní předpoklady:

- Konstantní jas – pixely (body) ve scéně budou mít relativně stejný jas v čase t i t+1.
- Malé pohyby – sousední body ve scéně leží ve stejné rovině a vykonávají stejný pohyb.
- Prostorová koherence – předpokládá se, že posun mezi jednotlivými sledovanými body je relativně malý a přibližně konstantní.

2.1.3 Kalmanův filtr

Kalmanův filtr je matematická metoda [6], která se používá pro odhad stavu lineárního systému. Na základě naměřených výsledků z předchozích měření se snaží najít hodnoty, které blíže odpovídají skutečnému stavu. Tento filtr je velmi používaný hlavně v měřicích systémech, navigačních a sledovacích systémech nebo ekonomických aplikacích. Mezi nejzajímavější nasazení v praxi patří například navigace balistických střel, střel Tomahawk nebo raketoplánů.

Pokud se zaměříme na sledování polohy, Kalmanův filtr se zabývá odhadem polohy x_k, která se řídí lineární stochastickou diferenciální rovnicí v závislosti na měřitelném výstupu systému z_k:

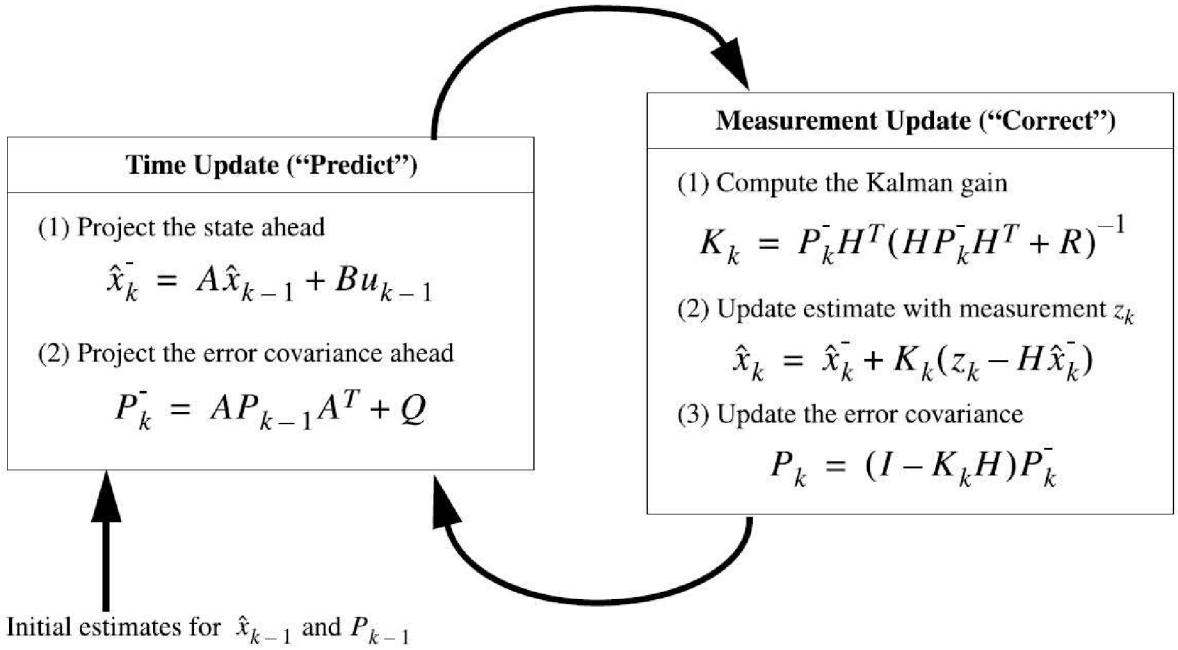
$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.4}$$

$$z_k = Hx_k + v_k\,. \tag{2.5}$$

Matice *A* je přechodová matice, která představuje vztah mezi předešlým časovým krokem *k-1* a aktuálním časovým krokem *k*. Matice *B* udává volitelný kontrolní vstup systému ve stavu *x*.

Náhodné proměnné w_k a v_k představují procesní šum a šum měření s normálním rozložením pravděpodobnosti:

$$\begin{aligned} p(w) &\sim N(0,Q) \\ p(v) &\sim N(0,R) \end{aligned} . \tag{2.6}$$



Obrázek 2.2: Schéma algoritmu Kalmanova filtru. Převzato z [6].

Algoritmus (obrázek 2.2) se skládá ze dvou částí: predikce (aktualizace v čase) a korekce (aktualizace měřením).

Při predikčním kroku je tedy odhadnut stav systému x_k a také kovariance chyby P_k:

$$x_k^- = Ax_{k-1} + Bu_k \tag{2.7}$$

$$P_k^- = AP_{k-1}F^T + Q, \tag{2.8}$$

kde Q je kovarianční matice.

Při korekčním kroku se nejprve spočítá Kalmanův zisk K, dále je vypočítán aktualizovaný stav systému x_k a také je aktualizovaná hodnota kovariance chyby P_k:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.9}$$

$$x_k = x_k^- + K_k(z_k^- - H_k X_k^-) \tag{2.10}$$

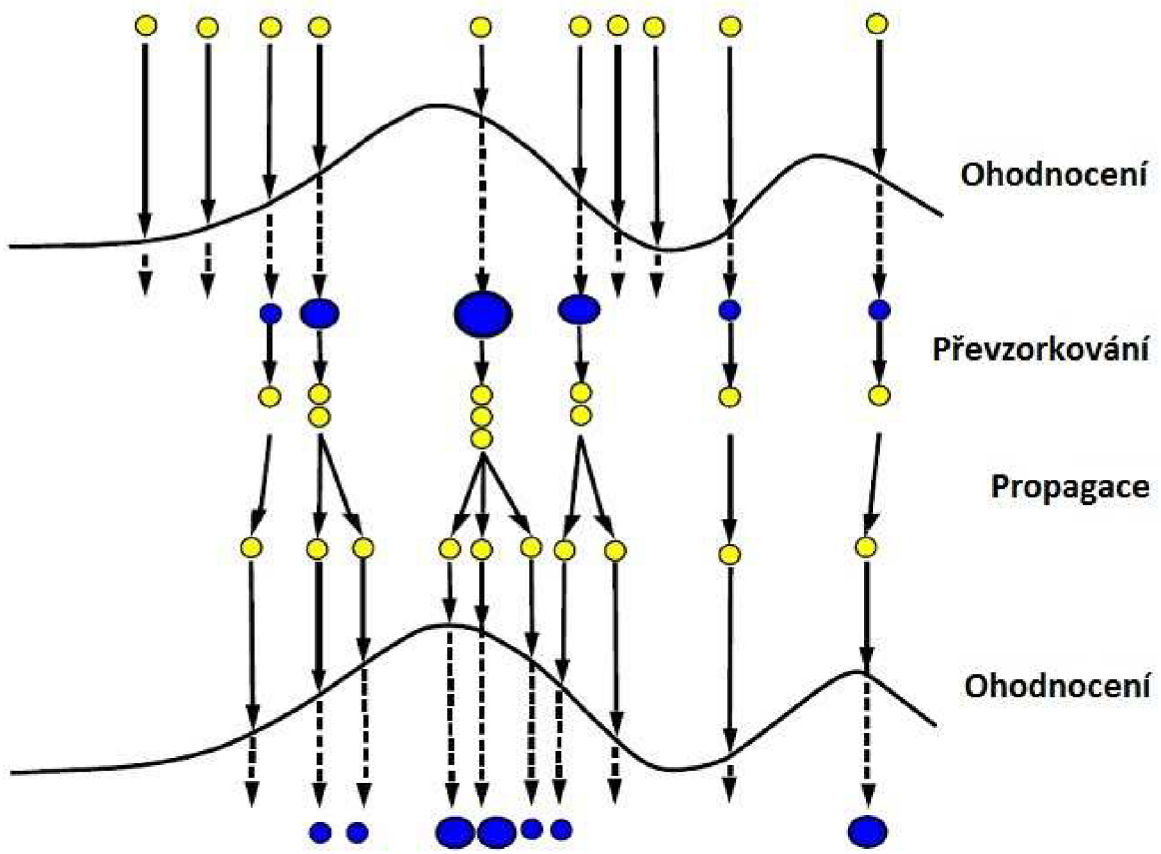
$$P_k = (I - K_k H_k)P_k^-, \tag{2.11}$$

kde H je matice určující vztah mezi naměřenými hodnotami modelem systému, matice R udává hodnotu chyby měření a z_k označuje změřené hodnoty.

2.1.4 Částicový filtr

Částicový filtr [8], který je také někdy označován jako Sekvenční Monte Carlo Metoda (SMC), je numerická metoda, která k odhadu modelu obvykle používá Baysovské modely. Tento filtr řeší problém odhadu stavu systému pro nelineární a ne-gaussovské dynamické modely.

Částicové filtry jsou založené na stejném principu jako Kalmanův filtr, ale z hlediska predikce pozice sledovaného objektu jsou obecnější metodou. Hlavní rozdíl oproti Kalmanovu filtru je v jeho dynamickém modelu objektu, který je zde modelován pomocí sady vzorků (částicemi) o určité váze, jež aproximují hustotu rozložení pravděpodobnosti. Mezi nejpoužívanější metodu založenou na částicovém filtru je condensation algoritmus (Conditional Density Propagation) [58].



Obrázek 2.3: Ukázka činnosti částicového filtru. Převzato z [7].

Postup výpočtu (obrázek 2.3) se stejně jako u Kalmanova filtru skládá ze dvou částí: predikce a korekce, které se provádějí pro každou částici (vzorek). Při predikci se nejprve provádí převzorkování, které zajišťuje, aby se u částic s vysokou váhou tato váha snížila tak, že se v jejím okolí vygenerují nové částice. To znamená, že částice s vyšší váhou mají větší pravděpodobnost, že se v jejich okolí vygenerují nové vzorky.

Ve fázi korekce je přiřazována váha na základě shody reprezentace dané částice se skutečným stavem. Částicím, které dobře reprezentují skutečnou polohu objektu, je váha zvyšována. Hůře

reprezentujícím částicím je naopak váha snižována a částice, které mají nulovou váhu, jsou odstraněny. Díky převzorkování ale nedojde k postupnému odstranění všech částic.

Kromě výše zmíněného condensation algoritmu, existují i jiné modifikace, jež se zpravidla liší v postupu převzorkování:

- Auxiliary particle filter [50]
- The Regularist particle filter [51]
- Gaussian particle filter [52]
- Unscented particle filter [53]
- Monte Carlo particle filter [54]
- Gauss-Hermite particle filter [55]
- Cost Reference particle filter [56]
- Rao-Blackwellized particle filter [57]

2.2 Klasifikátory

Klasifikace obrazu je proces, při kterém se určuje s jakou pravděpodobností se na daném místě nachází hledaný objekt. V tomto případě jde o to rozhodnout, zda hledaný objekt je hráč či nikoli. Klasifikátory můžeme rozdělit na binární nebo vícehodnotové. Binární klasifikátor vrací hodnoty ano nebo ne, které udávají, zda se objekt nachází na daném místě nebo ne. Vícehodnotový klasifikátor určuje pravděpodobnost, že se objekt nachází na daném místě.

Vstup pro klasifikátory je nejčastěji množina příznaků, jež popisuje daná data. Obvyklé vlastnosti objektu jsou barva, tvar nebo textura. Existují ale i jiné vlastnosti, například plocha objektu či natočení objektu, kterými lze data popsat. Přesnost klasifikace je tedy dána kvalitou a množstvím vzorků. Pokud máme předmět, jejž chceme klasifikovat, vhodně popsán (vstupní n-tice v podobě příznaků), může klasifikátor zařadit předmět do tříd. Výstupem klasifikátoru je tedy zařazení do tříd [9].

2.2.1 Trénování klasifikátorů

Základní rozdělení trénování klasifikátorů je:

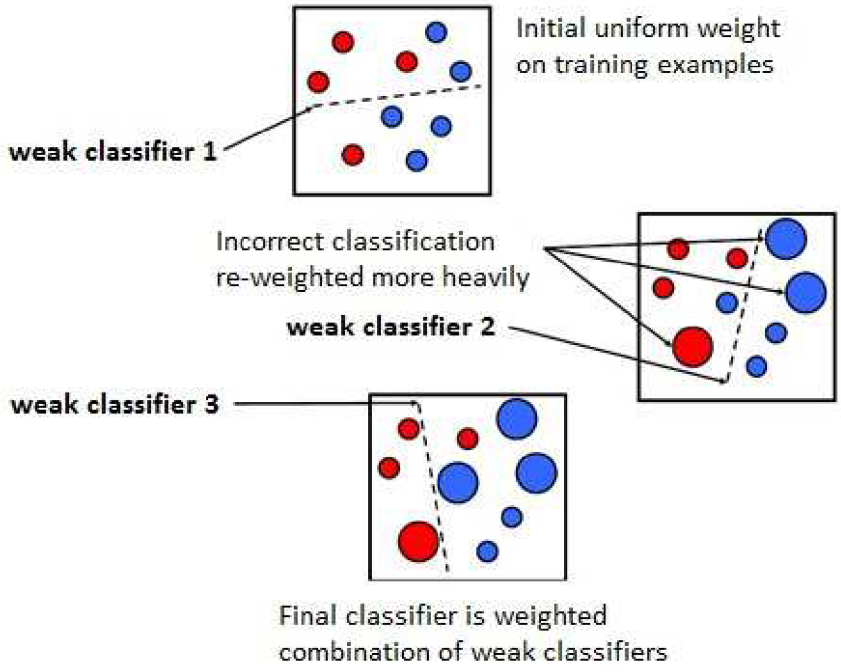
- učení bez učitele
- učení s učitelem

Při učení bez učitele probíhá dělení vstupních dat do tříd na základě daného kritéria kvality (tzv. clustering), kdy není k dispozici trénovací multimnožina.

Při učení s učitelem jsou k dispozici trénovací data, u nichž je známo, do které třídy patří, a touto sadou je klasifikátor natrénován. Poté už je klasifikátor schopen určit, kam má neznámá data zařadit.

2.2.2 AdaBoost

AdaBoost (Adaptive Boosting) [10, 11] je algoritmus strojového učení, který je využíván pro klasifikaci a detekci objektů. Boosting v názvu označuje vytváření velmi přesného klasifikátoru s výrazně lepšími vlastnostmi, kterému se říká silný klasifikátor, pomocí lineární kombinace většího množství jednoduchých klasifikátorů (obrázek 2.4), které se nazývají slabé klasifikátory a bývají založené na jednom příznaku. Tento slabý klasifikátor může být reprezentován například prahováním konkrétního příznaku, rozhodovacím stromem, perceptronem atd. Jediným omezením je, aby chyba slabého klasifikátoru byla menší než 0,5.



Obrázek 2.4: Postup vytvoření klasifikátoru. Převzato z [10].

Samotný algoritmus je iterační a v každém jeho kroku vybírá slabý klasifikátor, který nejlépe odděluje špatně klasifikovaná data, a ten je pak přidán k výslednému klasifikátoru.
Silný klasifikátor *H(x)* můžeme poté zapsat jako:

$$h(x_i) \in \{-1,1\}$$
$$H(x) = sign\sum_{i=1}^{N} \alpha_i h(x_i)\,, \tag{2.12}$$

kde α_i je váha nastavovaná při trénování a $h(x_i)$ jsou jednotlivé slabé klasifikátory.

2.2.3 SVM

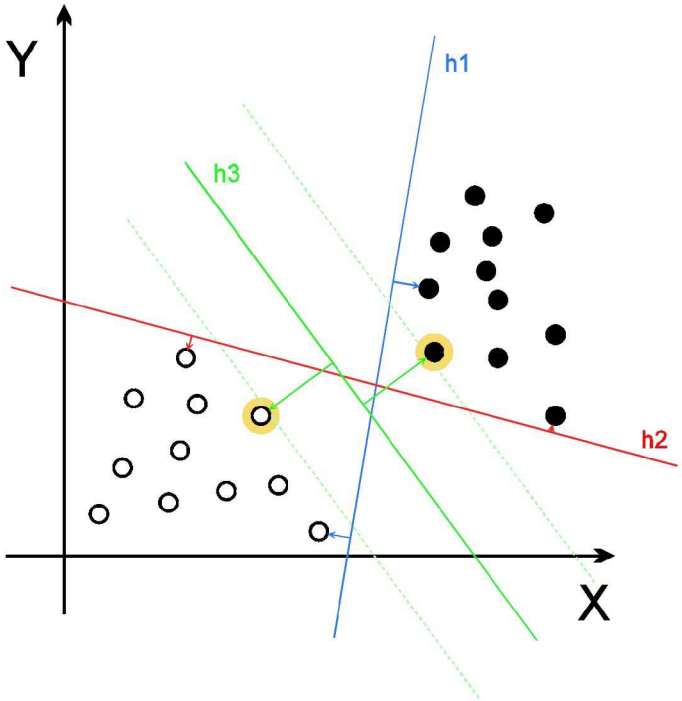
Support Vector machine (dále jen SVM) [12] patří mezi oblíbené klasifikační metody. Byla navržena C. Cortesem a V. Vapnikem. Jedná se o metodu strojového učení s učitelem, která je binární a rozděluje tedy vstupní data do dvou tříd.

Některá data není možné separovat lineárně a také často obsahují velké množství příznaků. Taková data je nutno transformovat do prostoru s vyšší dimenzí pomocí jádra, kde jsou již prvky mezi třídami snadno oddělitelné.

Vstupní sada je definována jako:

$$\{(x_i, y_i) \mid x_i \in \mathbb{R}^n, y_i \in \{-1,1\}\}, \tag{2.13}$$

kde y_i určuje třídu, kdy vzorek patří buď do jedné (-1), nebo do druhé (+1) třídy, a x_i je bod v obecně n-dimenzionálním prostoru.



Obrázek 2.5: Body v rovině a oddělující pásy. Převzato z [13].

Podstatou trénování je vytvoření roviny, která rozdělí vstupní sadu do dvou tříd tak, aby tato rovina měla od nejbližších prvků z každé třídy největší vzdálenost. Tím se vytvoří model, který se použije pro následnou klasifikaci.

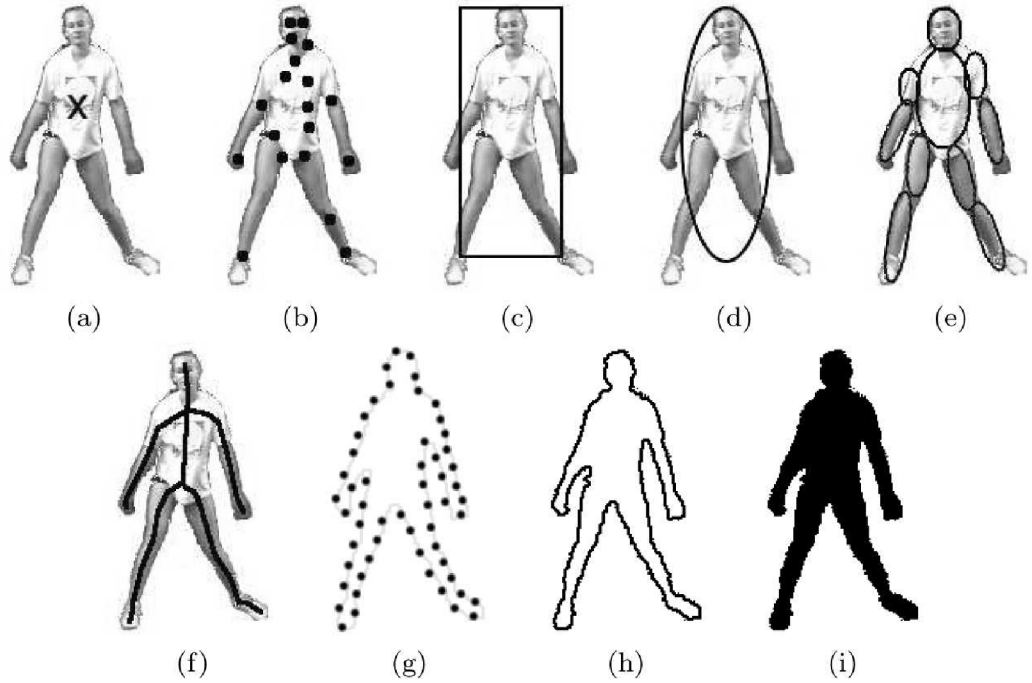
Princip lze také vysvětlit pomocí obrázku 2.5. Cílem SVM je nalézt optimální řešení, jež oddělí dané dva shluky. Z obrázku je patrné, že oddělení lze provést více způsoby, a proto nás zajímá řešení, které bude mít nejširší „okraj“. Nejbližší body k rovině z každého shluku jsou zvýrazněny žlutou barvou a můžeme říct, že tvoří její podporu (support).

2.3 Reprezentace objektů

Objekt lze reprezentovat velkým množstvím modelů, kdy je třeba brát v úvahu konkrétní použití, protože neexistuje univerzální popis, který by byl vhodný pro každý sledovaný objekt. V tomto případě, kdy je snaha vyjádřit co nejlépe daného hráče, musíme brát zřetel na vlastnosti, jakými jsou velikost hráčů, jejich pohyb, jejich vzájemné překrytí, změna velikosti hráčů, změny osvětlení nebo třeba šum v obraze.

Objekt lze popsat pomocí základních vlastností, což může být například barva, dominantní barva, histogram, rozložení jasu, textury atd. Kromě těchto základních popisů je možné objekt reprezentovat podle jeho tvaru. Nejpoužívanější způsoby (obrázek 2.6) popisu objektů [14] jsou pomocí :

- Bodu. Jedná se o bod, který je umístěný v těžišti objektu, nebo několika body. Bodová reprezentace je vhodná pro sledování objektů, které se nacházejí v malé oblasti.
- Základních geometrických tvarů. Jedná se hlavně o obdélník nebo elipsu.
- Siluety a kontury objektu. Kontura reprezentuje hranice objektu a silueta znamená vnitřek objektu.
- Popisu jednotlivých částí objektu. U lidského těla se jedná například o trup, nohy, ruce a hlavu, které jsou spojené pomocí kloubů.
- Kostry objektu. Tato reprezentace může být získána skeletonizací objektu, kdy vzniká tzv. střední osa objektu.



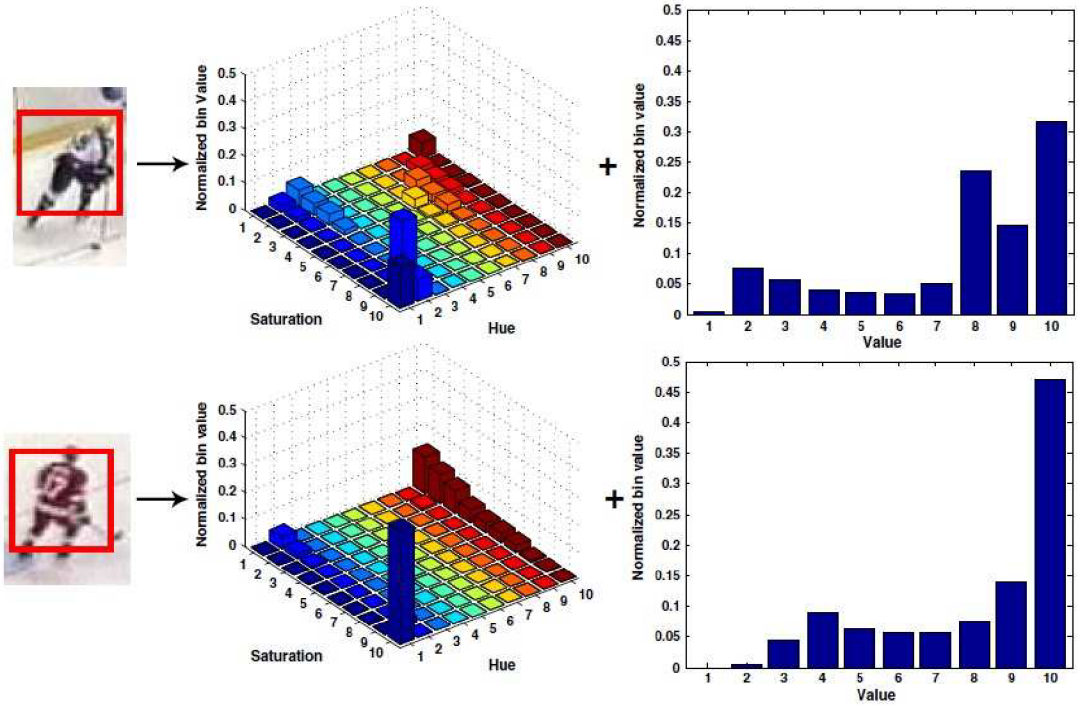
Obrázek 2.6: Reprezentace objektů. (a) Střed, (b) sada bodů, (c) obdélník, (d) elipsa, (e) části těla, (f) kostra, (g) kontura z bodů, (h) kontura, (i) silueta. Převzato z [14].

V případě sledování hráčů je třeba vhodně zakódovat vizuální informaci cíle. Pro zachycení barvy je použit HSV barevný histogram a pro zakódování tvaru je použit HOG (histogram orientovaných gradientů) [18].

Histogram

RGB model je náchylnější na náhlé změny osvětlení, takže by mohlo docházet k chybám při sledování objektu. Proto je použit HSV (Hue-Saturation-Value) model, který má tři složky – hue (tón), saturation (sytost) a value (jas). HSV tyto nedostatky odstraňuje a je pro tento účel vhodnější. Tento model odděluje intenzitu (jas) od barvy (odstín a sytost), takže je méně citlivý na světelné efekty a je výhodný, jelikož sportovci mají dresy různých barev.

Jak lze vidět na obrázku 2.7, bude model složen z 2D barevného histogramu (odstín a sytost) a 1D histogramu (jas). Oba tyto histogramy budou mít stejnou šířku binů ($N_h = N_s = N_v = 10$), takže vznikne stodeseti dimenzionální HSV histogram ($N_h \times N_s + N_v = 10 \times 10 + 10 = 110$).



Obrázek 2.7:Ukázka HSV histogramů hráčů. Převzato z [15].

HOG deskriptor

Pro vyjádření tvaru objektu je v této práci použit HOG deskriptor [18], protože je robustní vůči změně pohledu a osvětlení, vykazuje dobré výsledky a je efektivní. Podrobnější popis následuje v kapitole 3.

2.3.1 Porovnání histogramů

Abychom zjistili, jak moc se dva histogramy od sebe liší, je třeba je porovnat podle nějaké metriky. Metod pro porovnávání histogramů existuje celá řada, a proto budou popsány jen ty nejpoužívanější.

Průnik histogramů

$$H(x_1, x_2) = \sum_i \min(x_{1i}, x_{2i}) \tag{2.14}$$

Jedná se o velmi jednoduchou metody, kdy se prochází dva normalizované histogramy a do celkové sumy se započítává vždy menší hodnota z daného binu x_{1i} nebo x_{2i}.

Euklidovská vzdálenost

$$H(x_1, x_2) = \sum_i (x_{1i} - x_{2i})^2 \tag{2.15}$$

Euklidovská vzdálenost je jedna z nejzákladnějších metrik, při čemž platí, že čím je menší výsledná vzdálenost, tím jsou histogramy podobnější.

Nejmenší čtverce

$$H(x_1, x_2) = \sum_i \frac{(x_{1i} - x_{2i})^2}{x_{1i} + x_{2i}} \tag{2.16}$$

Tato metoda vykazuje lepší výsledky než předchozí dvě metody. Platí, že čím menší výsledná suma, tím jsou histogramy podobnější.

Bhattacharyyova vzdálenost

$$H(x_1, x_2) = \sum_i (x_{1i} \cdot x_{2i}) \tag{2.17}$$

Bhattacharyyova vzdálenost je velmi používaná metrika, která vyjadřuje míru podobnosti. V tomto případě platí, že čím je výsledná vzdálenost vyšší, tím jsou histogramy podobnější.

Ostatní metody

Kromě výše zmíněných metod existuje celá řada metrik vhodných pro porovnání dvou histogramů. Jsou to například Mahalanobisova vzdálenost, logaritmická pravděpodobnost nebo metoda korelace.

3 Analýza pohybu hráčů v záznamech sportovních utkání

Tato kapitola přibližuje základní přístupy pro detekci osob, podrobněji se věnuje nejvýznamnějším metodám a je zde také nastíněn postup pro detekci hřiště.

3.1 Detekce postav

Abychom byli schopni nějaký objekt (osobu) sledovat, je potřeba ho nejprve detekovat. K tomu existuje velké množství metod. Tři hlavní přístupy jsou:

- Na základě pohybujících se objektů v popředí.
- Na základě tvaru, siluety, kostry a dalších vlastností lidské postavy.
- Na základě detekce tváře.

Poslední přístup, který vychází z detekce tváře, je už velmi dobře řešený a generuje dobré výsledky. Velkou nevýhodou ovšem je, že obličej může být zakrytý, pootočený nebo ve špatné kvalitě, která je zapříčiněná třeba pohybem, vzdáleností nebo šumem. Z tohoto důvodu se těmto metodám práce nebude věnovat.

3.1.1 Na základě pohybujících se objektů v popředí

Tento přístup je nejjednodušší, velmi efektivní a snadno implementovatelný. Díky tomu je používán v řadě sledovacích aplikací. Jedná se o metodu odečítání pozadí, která je založena na vytvoření reprezentace scény nazývané model pozadí. Aktuální snímek se pak porovnává vždy s tímto modelem a rozdílné pixely vytvoří oblast, kde došlo k pohybu. V tabulce 3.1 lze vidět přehled metod, které jsou založeny na odečítání pozadí. Základní popisy metod lze nalézt v [16].

Tato metoda má ale celou řadu nevýhod a omezení, a proto není vhodná k účelům této práce. Uveďme nejdůležitější omezení:

- proměnlivé pozadí
- rychlé změny osvětlení
- problémy se stíny
- požadavek na statickou kameru

Paper	**Background subtraction**	**Human feature**
Wren et al. [26]	Color/Reference image	Color, contour
Beleznai et al. [27]	Color/Reference image	Region model

Haga et al. [28]	Color/Reference image	Image motion + human motion + motion continuity
Eng et al. [29]	Color/Reference image	Color
Elzein et al. [30]	Motion/Frame difference	Wavelets
Toth and Aach [31]	Motion/Frame difference	Fourier shape
Lee et al. [32]	Motion/Frame difference	Shape
Zhou and Hoang [33]	Motion/Frame difference	Shape
Yoon and Kim [34]	Motion + Color	Geometric pixel value
Xu and Fujimura [35]	Depth	Motion
Li et al. [36]	Depth	Shape
Han and Bhanu [37]	Infrared	IR + color
Jiang et al. [38]	Infrared	IR + color

Tabulka 3.1: Metody používající odečítání pozadí [16].

3.1.2 Na základě vlastností lidské postavy

Tento přístup už je složitější a není třeba ho používat jen ve videu (sekvence snímků), ale můžeme ho použít i pro detekce lidí v obrázcích. Tyto metody jsou založeny na modelování vlastností lidského těla pomocí různých příznaků a deskriptorů. Většinou je poté provedena klasifikace, která rozhodne, zda se jedné o člověka či nikoli. Někdy se tyto metody označují jako přímé, jelikož pracují přímo s danými snímky.

Nejčastěji používané příznaky, které můžeme vidět i v tabulce 3.2, jsou:

- Tvary. Nejčastěji kontury, hrany, různé deskriptory (HOG).
- Barvy. Většinou se používají k detekci kůže.
- Pohyb.
- Kombinace předešlých.

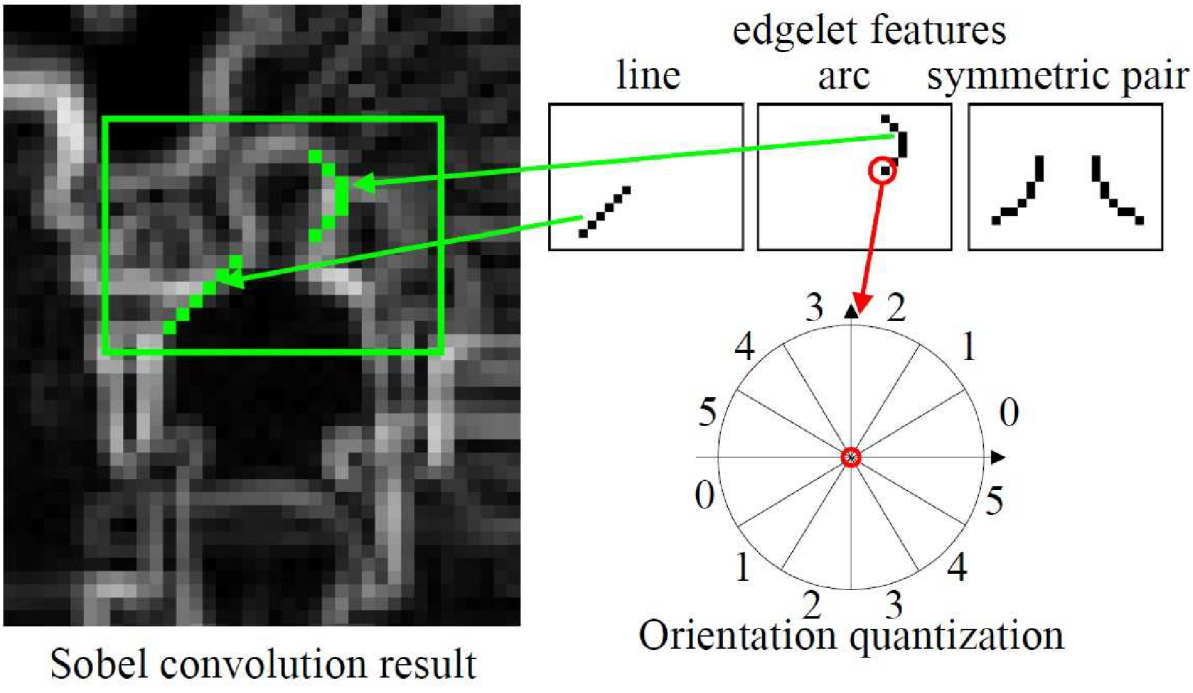
Paper	**Human model**	**Classifier**
Cutler and Davis [39]	Periodic Motion	Motion similarity
Utsumi and Tetsutani [40]	Geom. Pix. Val.	Distance
Gavrila and Giebel [23]	Shape template	Chamfer distance
Viola et al. [41]	Shape + motion	Adaboost cascade
Sidenbladh [42]	Optical flow	SVM (RBF)
Dalal and Triggs [18]	Hist. of gradients	SVM (Linear)
Papageorgiou et al. [43]	2D wavelet features	SVM
Mohan et al. [22]	Haar features	SVM
Mikolajczyk et al. [44]	Features based to orientation	Discrete AdaBoost
Wu and Nevatia [19]	Edgelet features	Real AdaBoost

Tabulka 3.2: Přehled metod přímé detekce [16, 17].

Nyní zde budou popsány některé zajímavé přístupy, ale největší pozornost bude věnována HOG – histogram orientovaných gradientů [18], protože se jedná o metodu, která vykazuje velmi dobré výsledky a hodí se pro účely této práce - k detekci lidí a jako reprezentace hráče (HOG deskriptor).

Pokud se zaměříme na porovnání metod, nejlépe si vedou metody HOG [18] a Edgelet features [19]. Smyslem porovnání je dosáhnout co nejvyšší úrovně detekce a také co nejmenšího množství falešných detekcí (false alarm rate). Pokud se zvýší počet detekovaných objektů, zvýší se i počet falešných detekcí, a proto je třeba oba tyto ukazatele brát v úvahu. V pracích [18,19] lze nalézt grafy, které srovnávají nejpoužívanější metody.

Wu a Nevatia [19] používají pro detekci lidí části lidského těla: hlavu a ramena, trup, nohy a celé tělo. Při detekci se nejprve detekují jednotlivé části těla a poté se kombinují dohromady. Jako příznaky jsou použity siluety (Edgelet features), což jsou čáry a křivky, které se nachází na hranách lidské postavy. Jedná se hlavně o části kružnic, čáry nebo křivky, které mohou tvořit i symetrické páry (obrázek 3.1). Pro výpočet velikosti a intenzity těchto hran je použit Sobelův operátor. Jako klasifikátor se využívá Reálný AdaBoost, kdy jako slabé klasifikátory jsou využity tyto příznaky, které tak vytváří silný klasifikátor.

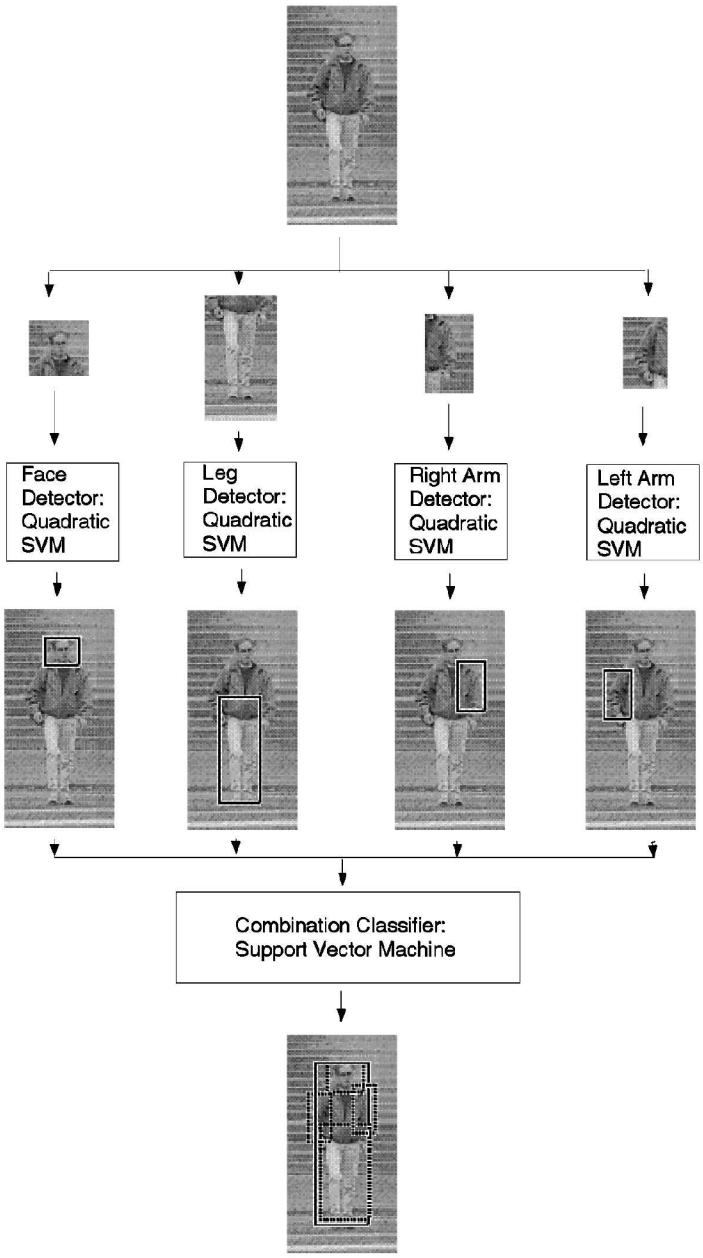


Obrázek 3.1:Ukázka příznaků. Převzato z [19].

Mohan et al. [22] využívají Haarovy příznaky (Haar features), jež jsou velmi známé díky použití při detekci tváře. Haarovy příznaky jsou jednoduché obdélníkové oblasti, které se aplikují na obrazová data. Získáme tak odezvu příznaků, která se skládá z rozdílu černé a bílé části.

V této práci se detekují jednotlivé části těla (obrázek 3.2). Konkrétně se jedná o hlavu a ramena, nohy, levou ruku a pravou ruku. Vzniká tedy kombinovaný detektor, který má lepší

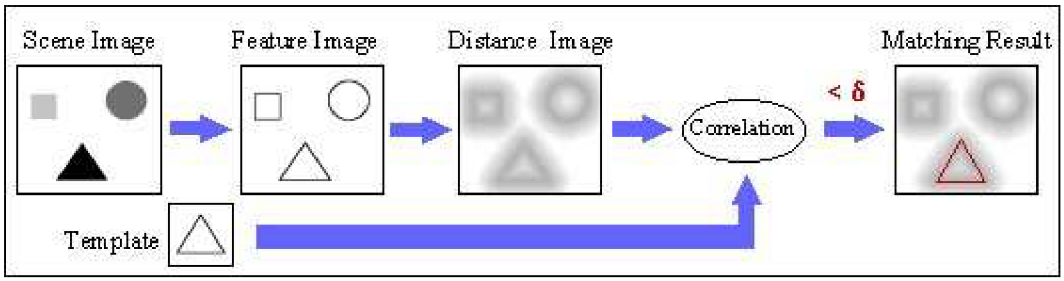
vlastnosti než jednotlivé detektory. Pro klasifikaci je použit SMV klasifikátor. Práce je zajímavá z hlediska použití detekce jednotlivých částí lidského těla, ale zároveň upozorňuje na nevhodnost použití Haarových příznaků, které mají dobré výsledky pouze pro detekci tváře.



Obrázek 3.2:Ukázka systému pro detekci osob využívající Haurovy příznaky. Převzato z [22].

Gavrila a Giebel [23, 24] ve své práci využívají kameru umístěnou na vozidle, kdy obraz z ní je použit pro detekci chodců. Metoda je založená na porovnávání šablon tvarů (obrázek 3.3) a vstupního obrázku převedeného na obrázek s příznaky a poté na „distanční“ obrázek. Příznaky se většinou získávají pomocí detekce hran a distanční obrázek je získán distanční transformací hran podle intenzity. Používá se zde hierarchický strom šablon, které umožňuje efektivní porovnávání. Proces porovnávání začíná od kořene stromu a postupuje přes uzly, jež mají nejlepší shodu s šablonou. Pro

porovnání se používá Chamferova vzdálenost. Pokud je vzdálenost větší než zvolený práh, hledání se nerozšíří pro podřízené uzly daného uzlu. Tímto je zajištěna efektivita porovnávání.



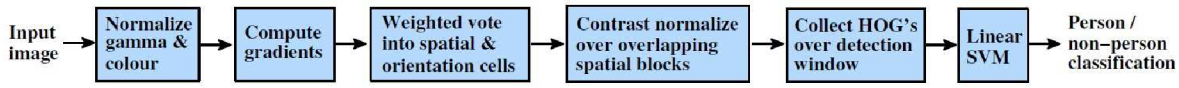
Obrázek 3.3:Ukázka procesu porovnávání. Převzato z [24].

3.2 Histogram orientovaných gradientů

Histogram orientovaných gradientů (anglicky Histogram of oriented gradient - HOG) představuje v současnosti jednu z nejlepších metod pro detekci lidí v obraze. V podstatě je tuto metodu možné použít pro klasifikaci a rozpoznávání libovolných objektů, které nemění své natočení v obraze. HOG byla představena v roce 2005 Dadalem a Triggsem [18], kdy zaznamenali výrazný pokrok v detekci osob.

Jednotlivé kroky při detekci jsou vidět na obrázku 3.4, kdy postup rozpoznání se skládá z následujících částí [21]:

- Výpočet gradientu (intenzita barevné změny) jednotlivých bodů.
- Rozdělení obrazu na malé části o dané velikosti.
- Výpočet histogramu pro každou část.
- Normalizace histogramů, kdy je obraz rozdělen na překrývající se části dané velikosti.
- SVM klasifikátoru je poslána kombinace těchto histogramů.



Obrázek 3.4: Postup klasifikace při použití HOG deskriptrů. Převzato z [18].

Gradient

Gradient obrazu neboli změna barevné intenzity sousedních pixelů v obraze se používá pro popis vlastnosti obrazu, většinou hran. Nejčastěji je gradient počítán z obrázku ve stupních šedi, kdy je získána jedna hodnota. U barevných obrázků se počítá každá barevná složka zvlášť. Nejčastěji se gradient získává pomocí Sobelova operátoru, diagonálního operátoru nebo jednoduchého vektoru, kdy se vypočítají složky ve směru osy x a y.

Konvoluční matice pro výpočet gradientu podle Sobelova filtru:

$$h_X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_Y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{3.1}$$

Použít lze i jednoduchou masku, kterou je možné použít v horizontálním i vertikálním směru:

$$h = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

Velikost gradientu:

$$|G| = \sqrt{G_X^2 + G_Y^2}, \tag{3.3}$$

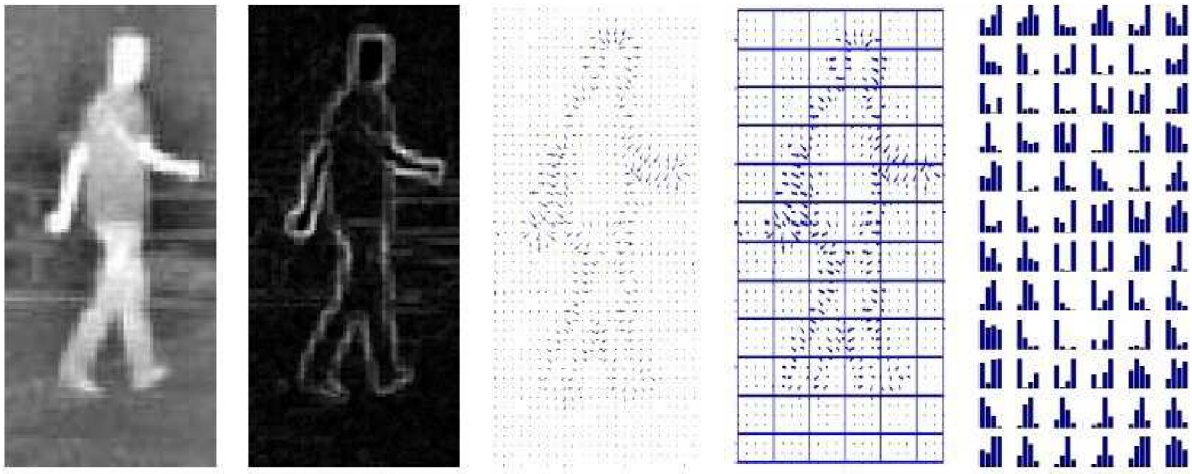
kde G_x a G_y jsou hodnoty gradientu ve směru x a y.

Směr gradientu:

$$\Theta(x, y) = \arctan\left(\frac{G_X}{G_Y}\right). \tag{3.4}$$

Výpočet histogramu

Obraz je nejprve rozdělen na bloky dané velikosti. Nejčastěji se jedná o bloky velikosti 4x4, 8x8 a 16x16. Pro každou takto rozdělenou buňku se zvlášť počítá lokální histogram, který má jeden rozměr (1D histogram). Pro každý pixel je tedy známa velikost a směr gradientu. Směr gradientu je nutné rozdělit do několika rozsahů (0° - 180° nebo 0° - 360°). Nejčastěji je tento rozsah rozdělen na 9 částí (binů). Velikost rozsahů ovlivňuje míru nezávislosti na možném natočení daného klasifikovaného objektu. Tyto rozsahy tvoří kanály histogramu a velikost těchto kanálů závisí na velikosti daných gradientů nebo jsou přímo úměrné této velikosti. Z toho vyplývá, že každý kanál poskytuje přehled zastoupení jednotlivých směrů gradientů. Příklad výpočtu histogramů je na obrázku 3.5.



Obrázek 3.5: Získání HOG deskriptorů z obrázku. Převzato z [20].

Deskriptor, normalizace a klasifikace

K docílení větší nezávislosti na okolních podmínkách, jako je osvětlení či stíny, je vhodné provést normalizaci oblastí o dané velikosti. Existuje celá řada normalizačních schémat. Nejpoužívanější metodou je seskupení buněk do bloků, kdy nad těmito bloky je provedena normalizace. Nejčastěji se používají bloky o velikosti 2x2 nebo 3x3 buňky, kdy se jednotlivé bloky překrývají. Pokud je tedy velikost jednoho bloku např. 8 pixelů, bude mít výsledná buňka velikost 16x16 nebo 24x24 pixelů. Samotná normalizace se provede tak, že se každá třída vydělí sumou všech hodnot v dané oblasti.

Normalizované histogramy pro daný blok nazýváme HOG deskriptor. Sada těchto deskriptorů, která reprezentuje obraz, se poté předává SVM klasifikátoru, který určí, zda se jedná či nejedná o osobu.

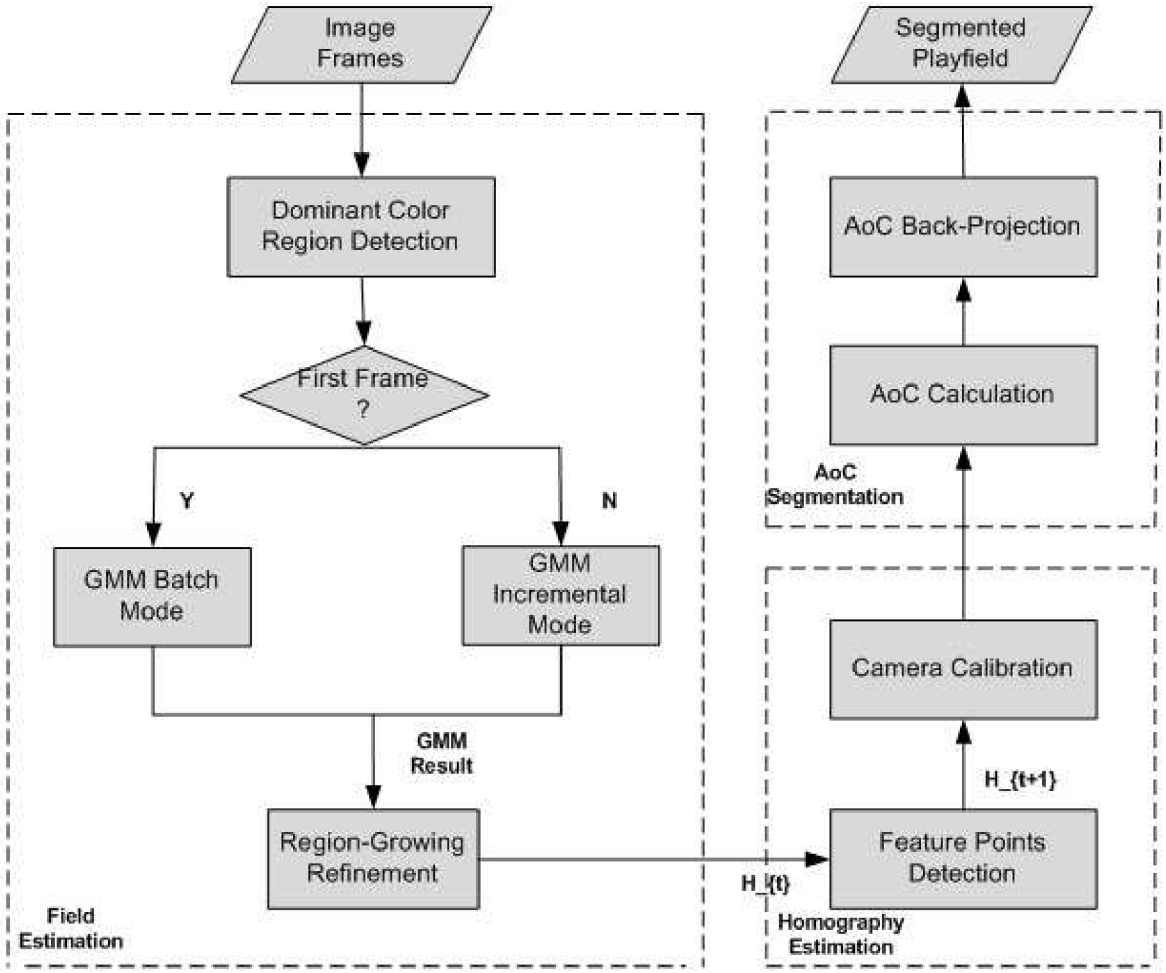
3.3 Detekce hřiště

Tato část práce ukazuje postup, jak lze detekovat fotbalové hřiště v záznamu sportovního utkání. Detekce hřiště je důležitá problematika, která spadá do oblasti analýzy sportovního utkání. Užitečná může být pro usnadnění detekce či sledování objektů (hráčů, rozhodčích nebo míče). Pro tuto práci by bylo vhodné metodu použít pro zjištění, na kterou oblast hřiště se kamera „dívá“. Díky těmto informacím lze už snadno vytvářet statistiky o pohybu, rychlosti či trajektorii hráče.

Postup detekce lze vidět na obrázku 3.6 a skládá se z následujících částí:

- Detekce regionu pomocí dominantní barvy.
- GMM (Gaussian Mixture Model) segmentace.
- Metoda narůstání oblastí (region growing).
- Houghova transformace pro detekci čar.
- Detekce význačných bodů (rohy hřiště, pokutové území atd.).
- Kalibrace kamery.
- Zjištění výsledné oblasti, na kterou je zaměřena kamera.

Pro více informací doporučuji článek [25].



Obrázek 3.6: Ukázka systému pro detekci hřiště. Převzato z [25].

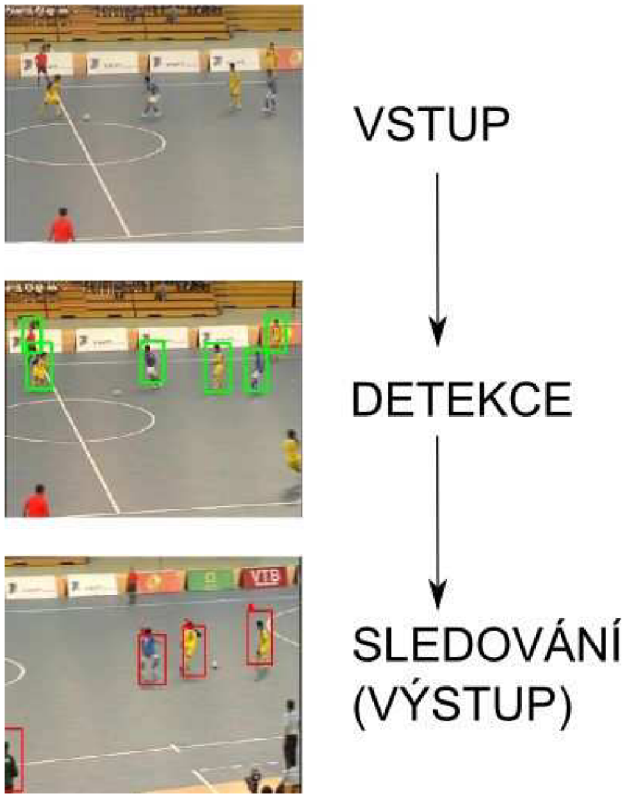
4 Návrh aplikace

Tato kapitola přibližuje základní cíle návrhu, popis vstupních i výstupních dat, volbu technologií pro detekci a sledování včetně odůvodnění nebo problémů, na které lze narazit při návrhu. Nechybí ani obecný popis navržené aplikace s vývojovým diagramem. Detailněji jsou také analyzovány jednotlivé body návrhu.

4.1 Cíle a funkce aplikace

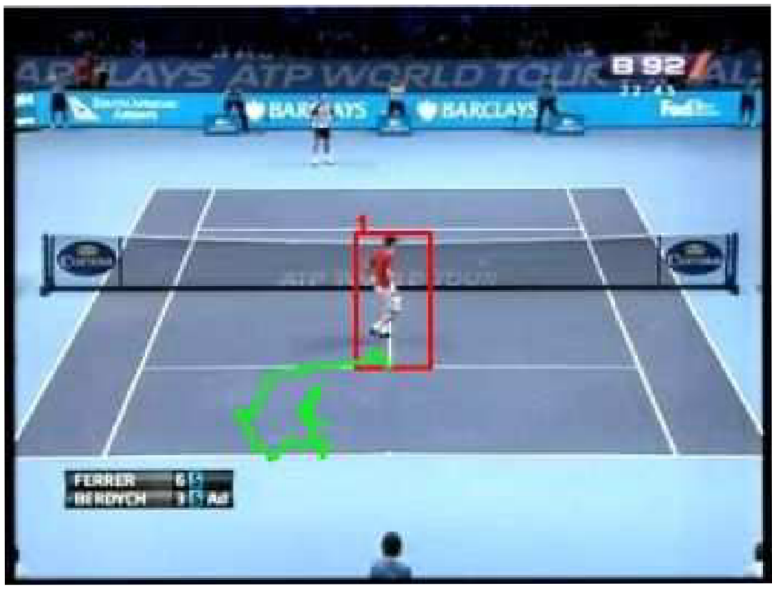
Cílem diplomové práce bylo navrhnout aplikaci pro detekci a sledování hráčů v záznamu sportovních utkání. Aplikace by měla detekovat hráče, kteří se vyskytují v průběhu celé video sekvence a následně provádět jejich sledování. To znamená, že se budou noví hráči přidávat, ale i odebírat ze systému.

Vstupními daty (obrázek 4.1) bude video sekvence se záznamem sportovního utkání (hokej, fotbal, futsal, tenis, basketbal atd.). Tyto záznamy se vyznačují tím, že mohou být pořízeny jak statickou, tak pohyblivou kamerou, a proto při nich může docházet k pohybu a natáčení kamery. Aplikace se bude umět vyrovnat i se zoomováním, menšími změnami osvětlení nebo stíny. Záznamy sportovních utkání mívají také různou kvalitu video sekvence a rozdílné rozlišení. Výsledná aplikace by měla analyzovat všechny tyto video sekvence bez ohledu na kvalitu obrazu.



Obrázek 4.1: Ukázka vstupu a výstupu programu.

Výstupem programu bude video sekvence obsahující detekované hráče, kteří jsou zvýrazněni pomocí rámečku s číslem, jež je unikátní a lze z něho poznat, v jakém pořadí byl daný hráč přidán do sledovacího systému. Ke každému hráči bude taky možné vykreslit trajektorii (obrázek 4.2), již při sledování vykonal.



Obrázek 4.2: Ukázka trajektorie, kterou tenista vykonal.

Výše popsané cíle lze shrnout do následujících bodů:

- Co nejlepší detekce hráčů bez falešných detekcí.
- Statická i pohyblivá kamera.
- Zoom kamery.
- Mírné změny osvětlení a stíny.
- Různé velikosti rozlišení vstupní video sekvence.
- Různá kvalita obrazu včetně rozmazaných hráčů.
- Přesné sledování detekovaných hráčů.
- Trajektorie pohybu každého hráče.
- Přidávání a odebírání hráčů.

4.2 Obecná idea a volba technologií

Aby bylo možné hráče sledovat, je třeba je nějakým způsobem detekovat. K detekci lze použít různé metody, které jsou popsány v kapitole 3. Pro tyto účely byl vybrán detektor, který je založen na SVM klasifikátoru využívající HOG deskriptory. Tato metoda vykazuje dobré výsledky a hodí se pro tuto aplikaci.

Při návrhu je také nutnost brát v úvahu výpočetní náročnost daného řešení detekce, které se pohybuje v řádech sekund na jeden snímek v závislosti na nastavení detektoru (krok pohybujícího se okénka, krok pyramidy atd.) a hlavně velikosti vstupního videa. Například u záznamu o velikosti 320x240 je doba detekce kolem jedné sekundy.

Detekovat hráče ale není třeba pro každý snímek videa, protože již detekovaní hráči jsou sledováni. Pro účely sledování je nutné vybrat vhodnou metodu popsanou v kapitole 2, která splňuje požadavky kladené na aplikaci. Především se jedná o pohyb a zoomování kamery, rychlý pohyb a prudké změny hráčů (protipohyb, vzájemné srážky) nebo překrývání hráčů mezi sebou. Nejvhodnější metoda je částicový filtr, jenž splňuje tyto požadavky.

Popis objektu (v tomto případě hráče) bude z největší části tvořen HSV histogramem, který je méně náchylný na změny osvětlení a světelné efekty než RGB histogram. Sledovaný objekt je reprezentován obdélníkem, který ho ohraničuje. Jeho histogram je ale počítán z elipsy, jež lépe eliminuje vliv pozadí. Pro popis objektu je použit také HOG deskriptor zachycující tvar objektu. Ten se časem mění, a proto je třeba tomuto popisu dát menší váhu.

4.3 Problémy při detekci a sledování

Mezi zajímavé problémy patří například zjištění, o jakého konkrétního hráče se jedná. Pro řešení této situace existují dva hlavní způsoby:

1. Pomocí čísla dresů.
2. Pomocí obličeje.

Bohužel ani jednu metodu není v tomto případě možné použít, jelikož hráči jsou v záznamech příliš malí nebo rozmazaní, video má malé rozlišení, obraz je „zašuměný“ nebo mohou být hráči špatně natočeni, takže jim nejde vidět číslo ani obličej.

Dalším problémem, kterému se při zápase nevyhneme, je překrývání objektů (hráčů) při sledování. Tato aplikace by si s touto překážkou měla poradit, protože pro sledování je použit částicový filtr, který pro reprezentaci objektů využívá dvou modelů – histogram a HOG deskriptor. Bohužel nastanou i situace, kdy se překrývají dva hráči stejného týmu, kdy mají i stejné dresy a jejich histogramy jsou velmi podobné. V tomto případě nelze správně rozhodnout, koho z nich si vybrat, a dochází k chybám při sledování.

Často také dochází ke zmizení sledovaného objektu ze scény. Tento stav se řeší tak, že se několik následujících snímků drží v systému informace o objektu, a pokud se po určité době objekt neobjeví, je označen za ztracený. Tuto metodu nelze použít vzhledem k výše zmíněné podobnosti histogramů u hráčů stejného týmu. Proto při zmizení hráče ze scény je hráč označen okamžitě jako ztracený. Při detekování nového hráče je zase tento hráč označen jako nový a přidán do systému.

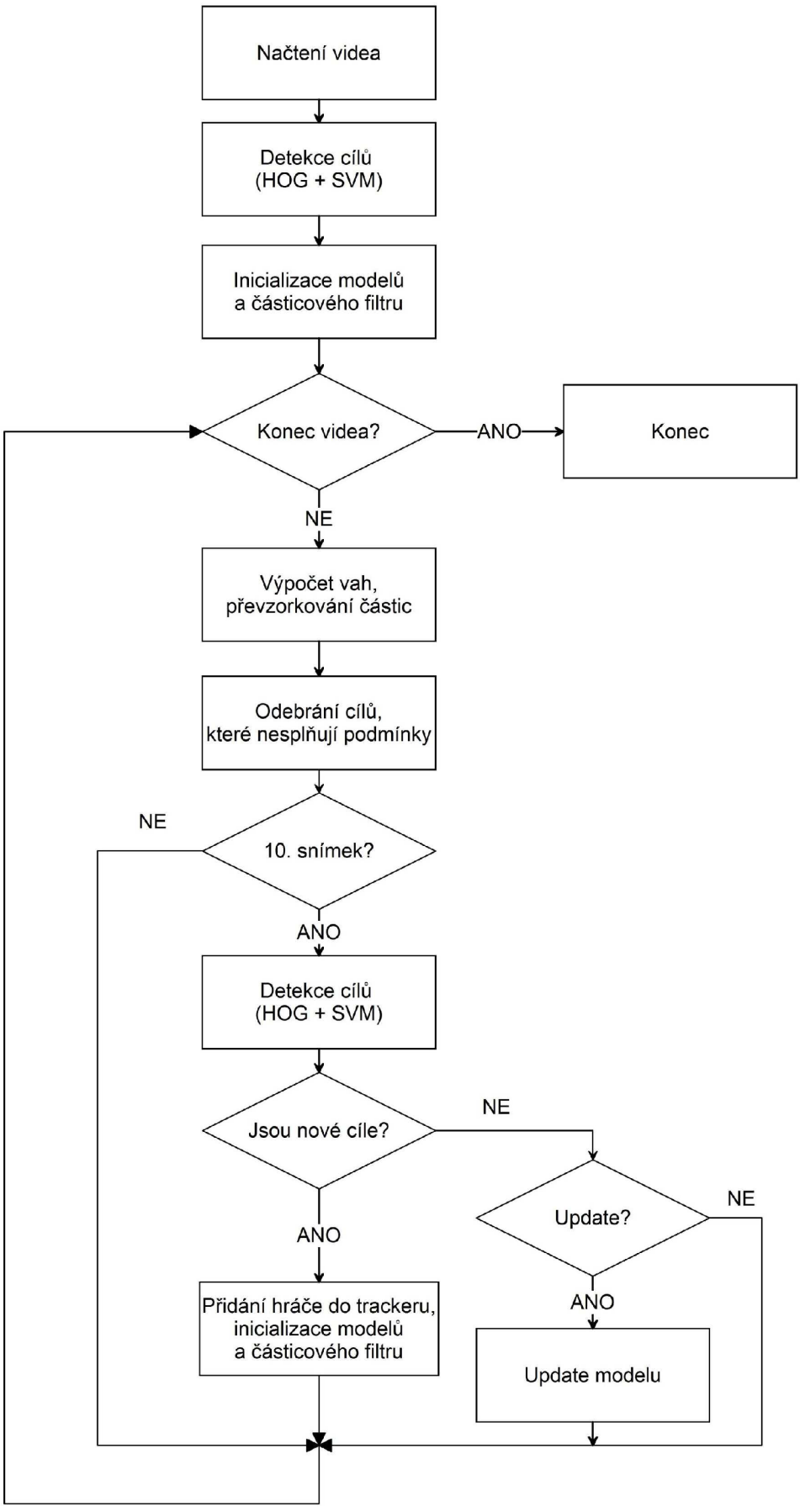
4.4 Podrobný návrh detekce

Vývojový diagram výsledné aplikace lze vidět na obrázku 4.3. Nejprve se v prvním snímku detekují hráči. Pro každého hráče se inicializuje jeho model a provede se konfigurace a inicializace částicového filtru. Poté se cyklicky bude načítat další snímek, dokud není konec videa. U každého snímku se pro všechny hráče vypočítají váhy jednotlivých částic a provede se převzorkování. Aby bylo možné do systému přidávat nové hráče, provádí se při každém desátém snímku detekce hráčů. Když jsou nové cíle detekovány, je třeba tyto hráče přidat do systému, inicializovat jejich model a vytvořit pro ně částicový filtr včetně jeho inicializace. Nakonec se kontroluje, zda jsou při sledování splněny podmínky, které kontrolují, zda hráč nezmizel z obrazu. Pokud hráč zmizel, je odebrán z programu.

S detekcí hřiště momentálně návrh nepočítá, ale do budoucna je to jedna z možností dalšího vývoje.

Nyní budou popsány hlavní části návrhu podrobněji:

- Detekce hráčů.
- Inicializace modelů a částicového filtru.
- Kontrola sledovaných hráčů.
- Výpočet vah a převzorkování částic.
- Zjištění nového cíle.
- Update modelů.



Obrázek 4.3: Vývojový diagram navrženého systému.

Detekce hráčů

Detekce je založena na detektoru využívající SVM klasifikátor a HOG deskriptor. Nejprve je třeba detektor inicializovat, což spočívá v nastavení parametrů a hlavně načtení vektoru ze souboru, který reprezentuje natrénovaný klasifikátor. V diagramu vidíme, že detekce proběhne pro 1. snímek a poté každý desátý. Po provedení samotné detekce je získán vektor obdélníků, které reprezentují nalezené hráče.

Inicializace modelů a částicového filtru

Všichni detekovaní hráči v prvním snímku jsou do systému přidáni. V dalších snímcích jsou přidáni už jen ti, kteří splňují podmínku, že se jedná o nového hráče. Přidání hráčů do systému spočívá v tom, že je třeba vytvořit a inicializovat jejich model (HSV histogram a HOG deskriptor) a částicový filtr.

Výpočet vah a převzorkování částic

Pro každou částici je třeba vypočítat váhu. Ta se určuje jako podobnost modelu a dané částice, kdy se počítá Bhattacharyyova vzdálenost v mezi histogramy a HOG deskriptory. Tyto dvě váhy je nutné normalizovat a poté vynásobit konstantou, která reprezentuje důležitost daného popisu. Nakonec se tyto čísla sečtou do výsledné váhy v. Například:

$$v = 0,8 \cdot v_{hist} + 0,2 \cdot v_{HOG} \,. \tag{4.1}$$

Pokud máme takto vypočtené váhy pro všechny vzorky (částice), můžeme provést převzorkování. Vzorek s nejvyšší vahou určuje sledovaný objekt.

Kontrola sledovaných hráčů

Hráče je třeba kontrolovat, zda jsou správně sledováni. Při situacích, kdy například hráč odejde z videa nebo je sledovacím algoritmem ztracen, je nutné tohoto hráče ze systému odebrat. Abychom zjistili, že taková situace nastala, musíme kontrolovat tyto podmínky:

- Rozdíl histogramů modelu a aktuálního stavu sledovaného hráče:

  Aby se předešlo zbytečnému odebírání, musí být v 2x po sobě jdoucích snímcích rozdíl histogramů větší než zadaný práh. Tato situace nastává nejčastěji.

- Poměr stran aktuálního stavu sledovaného hráče:

  Tato situace nastává ojediněle a zpravidla se projevuje tím, že je sledovaný hráč reprezentován moc úzkým obdélníkem.

- Dva obdélníky jsou na stejné pozici:

  Tato situace může vzniknout, pokud se střetnou dva hráči s podobným histogramem. Nastane tak, že sledovací obdélník přejde na spoluhráče a ten je poté sledován dvěma částicovými filtry.

Situaci lze detekovat tak, že oba obdélníky budou mít své středy vzdáleny blízko sebe. Pokud jsou tyto středy blízko 5x po sobě, odebere se hráč s větším identifikačním číslem.

Zjištění nového hráče

Vektor, který obsahuje obdélníky reprezentující detekované hráče, je třeba porovnat se sledovanými hráči v systému. Pro každý obdélník se zjišťuje, zda se nepřekrývá s nějakým hráčem v systému. Pokud je tato podmínka splněna, může být přidán do systému.

Update modelu

Jelikož při sledování není hráč přesně vycentrovaný v obdélníku, je obtížné provádět update modelu dle částice s nejlepší vahou pro každý snímek, protože by se původní model hráče „deformoval“ a po nějakém čase by byl model nepoužitelný. Proto je možné k aktualizaci využít obdélníky, které vzniknout při detekci, protože jsou přesněji vycentrované. V tomto případě lze použít jen obdélník, jehož střed je blízko středu sledovaného hráče. V opačném případě se může jednat o jiného hráče a docházelo by znovu k deformaci modelu. Samotný update proběhne tak, že se histogram i HOG deskriptor modelu nahradí nově vypočtenými hodnotami z detektoru.

Nejjednodušší variantou je ale situace, kdy je update vypnutý. To není na škodu, protože hráč má pořád stejný dres a histogram se mění minimálně. Také dochází k méně chybám, protože při aktualizaci se může více hráčů překrývat a histogram by tudíž byl nepřesný. Implicitně bude tedy update vypnutý, ale lze ho zapnout.

5 Implementace

V této kapitole budou popsány použité technologie a vytvořené třídy, které ukazují, jak je reálně implementovaná aplikace. Tyto třídy jsou zde podrobněji analyzovány. Nechybí ani postup tréninku detektoru.

5.1 Použité technologie

Výsledná aplikace je implementována v jazyce C++. Vývoj a testování probíhalo na operačním systému Windows 7 v prostředí Microsoft Visual Studio 2010. Při implementaci byly použity následující technologie.

OpenCV 2.2

OpenCV [45] je volně dostupná multiplatformní knihovna zaměřená na algoritmy počítačového vidění a zpracování obrazu využívaná po celém světě. Byla vyvinuta společností Intel a je přístupná pod licencí BSD. Obsahuje velké množství funkcí a algoritmů od jednoduchých pro práci s obrázky a videem až po složitější, jako je segmentace, rozpoznávání, detekce, sledování a mnoho dalších.

Pro tyto účely je využívána k usnadnění práce s obrazem, pro výpočet histogramu orientovaných gradientů nebo pro detekci lidí ve statickém obraze.

OpenCVX

OpenCVX [46] je volně dostupná knihovna pod licencí MIT. Tato knihovna se snaží implementovat některé algoritmy počítačového vidění, které v OpenCV chybí.

Jelikož jsou v OpenCVX implementovány funkce pro práci s částicovým filtrem, snažím se tyto funkce maximálně využít při implementaci algoritmu pro sledování objektů.

Visual Studio 2010

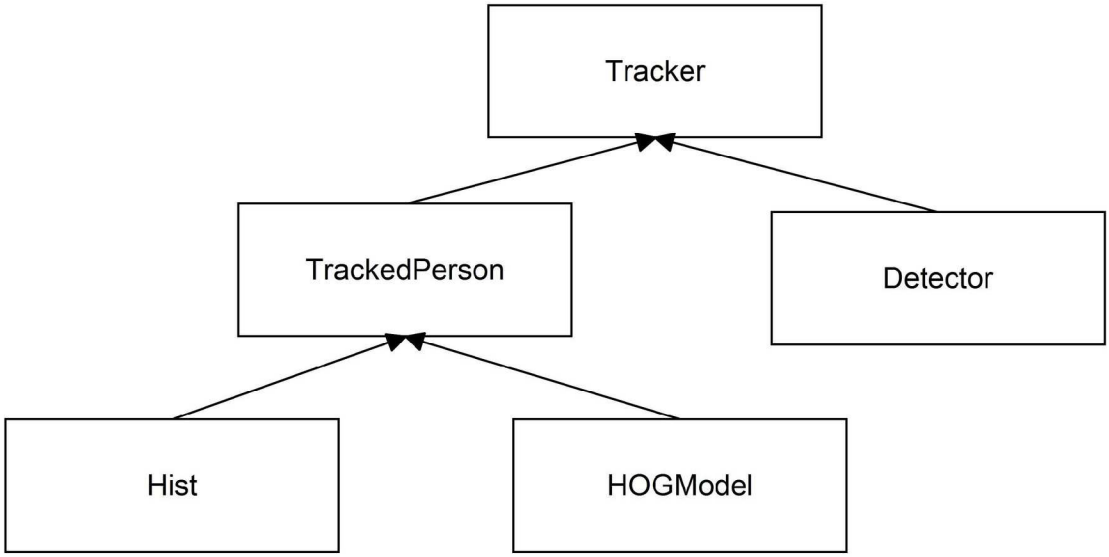
Jedná se o profesionální a velice používané vývojové prostředí od Microsoftu, které je velmi komplexní a usnadňuje práci uživateli.

SVMLight

Tato knihovna [47] slouží pro práci s SVM (support vector machine). Součásti této knihovny jsou i nástroje pro trénování vlastního SVM klasifikátoru, které jsem využil. Tyto nástroje lze stáhnout již přeložené.

5.2 Třídy aplikace

Strukturu programu lze vidět na obrázku 5.1. Uvnitř hlavního programu běží hlavní smyčka načítající jednotlivé snímky videa a provádějící kroky dle návrhu v kapitole 4. O většinu ostatních činností, jako je například sledování a detekce hráčů, se stará třída Tracker, která využívá zbylé třídy.



Obrázek 5.1: Struktura programu.

Tracker

Tato třída reprezentuje sledovací systém. Hlavním úkolem je uchovávat informace o aktuálně sledovaných osobách a provádět hlavní funkce systému, například kontrolu sledovaných osob z hlediska odebírání, přidání nové osoby do systému, provádí krok sledovacího algoritmu pro všechny osoby, stará se o detekci nových objektů nebo zkontroluje, zda se jedná o nový objekt či nikoli. K těmto činnostem využívá třídy TrackedPerson a Detector.

TrackedPerson

Třída TrackedPerson reprezentuje sledovanou osobu, u které je nutné uchovávat velké množství informací. Mezi nejdůležitější uchovávané informace patří aktuální pozice objektu, trajektorie, popis objektu HSV histogramem a HOG deskriptorem nebo struktura reprezentující částicový filtr, jež je použita z knihovny OpenCVX (konkrétně soubor `cvparticle.h`), kde jsou také implementovány základní funkce pro práci částicového filtru. Konkrétně se jedná o Condensation algoritmus (kapitola 2.1.4).

Ve třídě TrackedPerson je také implementováno velké množství nezbytných metod nutných k činnosti částicového filtru, ale knihovna `cvparticle.h` je neobsahuje. Jedná se například o funkci `measureParticle`, která přidá jednotlivým částicím váhu.

Detector

Tato třída se stará pouze o inicializaci a detekci hráčů ve snímku. Pro detekci je použita třída HOGDescriptor, která je implementovaná v OpenCV. Při vytváření objektu pomocí konstruktoru je nutné nastavit následující parametry (tabulka 5.1):

blockSize	16x16
blockStride	8x8
CellSize	8x8
derivAperture	0
gammaCorrection	0
histogramNormType	0
L2HysTheshold	0,2
nbins	9
winSigma	-1
winSize	64x128 (32x64)

Tabulka 5.1: Nastavení parametrů konstruktoru.

Je nutné dát pozor, aby při trénování vlastního klasifikátoru bylo použito stejné nastavení pro velikost detekčního okna.

Pro nastavení vlastního detektoru se použije metoda:
`setSVMDetector(const vector <float> & _svmdetector)`, kde parametr je vektor reprezentující natrénovaný detektor.

Pokud jde o samotnou detekci, je použita metoda `detectMultiScale`, u které jsou důležité tyto parametry (tabulka 5.2):

Práh	0
Práh pro vzájemné překrývání obdélníků	1
Koeficient zvětšování detekčního okna	1,15

Tabulka 5.2: Nastavení parametrů pro detekci.

Jelikož je detektor pyramidový, je možné zvolit koeficient, který udává o kolik se zvětší detekční okno mezi jednotlivými stupni pyramidy. Zbylé dva parametry spolu navzájem souvisí a je nutné je vhodně nastavit. Pro každý detektor se liší, a proto je žádoucí vytvořit ROC křivku, která ukáže nejvhodnější nastavení.

Hist

Jedná se o třídu reprezentující HSV histogram sledovaného objektu a stará se o veškerou práci s ním. Mezi nejdůležitější funkce patří transformace RGB obrazu na HSV reprezentaci, výpočet 1D histogramu, jeho normalizace a update histogramu.

HOGModel

HOGModel je třída představující popis objektu pomocí HOG deskriptoru. Stejně jako u třídy Hist, HOGModel se stará o výpočet HOG deskriptoru, normalizaci a update.

Pro výpočet deskriptoru se využívá metoda `HOGDescriptor::compute`, která je opět implementována v OpenCV.

5.3 Trénování SVM klasifikátoru

Kromě defaultního detektoru, který je obsažen v OpenCV, je možné použít vlastní natrénovaný detektor. V této kapitole bude popsán postup trénování a použité nástroje. Výsledné detektory a jejich trénovací sady jsou popsány v kapitole 6, kde lze najít i jejich vzájemné porovnání.

Postup trénování

Nejprve je potřeba přichystat dostatečně velkou trénovací sadu obsahující negativní i pozitivní obrázky. Je vhodné mít i testovací sadu, s jejíž pomocí se provede následná evaluace. Obecně platí, že čím větší sada, tím je výsledný detektor lepší. V některých případech může dojít k přetrénování. Tato situace se dá ošetřit použitím funkce „odebrání nekonzistentních vzorků a přetrénování“ v knihovně SVMLight. Jelikož obrovské trénovací sady nepoužívám, tuto funkci nechávám vypnutou.

Pro pozitivní datovou sadu je potřeba připravit obrázky o velikosti 64x128 (32x64). Pro tyto účely je vytvořena jednoduchá aplikace *createDataset*, která ze vstupního videa vyřezává obrázky pomocí myši. Ořezané obrázky se ukládají v několika rozlišeních, takže lze detektor natrénovat na různou velikost detekčního okna. Tato aplikace má také ještě jednu funkci, kterou lze využít. Umí prohledat složku s již existující datovou sadou a vytvořit z ní několik stejných sad o různých rozlišeních.

V tomto případě se velikost pozitivní sady pohybuje mezi 700-1000 položek. Negativní datová sada může obsahovat libovolně velké obrázky a měla by obsahovat více prvků než sada pozitivní. To je zajištěno tak, že z každého obrázku je náhodně vyříznuto 10 obrázků. Negativní sada použita v této práci má kolem 1300 prvků a po vynásobení 10 vznikne sada o velikosti 13000 prvků.

Pokud je vytvořena trénovací sada, je potřeba vypočítat HOG deskriptory pro všechny prvky a uložit do „trénovacího“ souboru. Stejná činnost se provede také pro testovací sadu a výsledky se uloží do „testovacího“ souboru.

Pro tento účel je vytvořena aplikace *HOGCompute*, která pro zadané složky s pozitivní, negativní a testovací sadou vytvoří příslušné soubory obsahující HOG deskriptory. Pro výpočet HOG deskriptorů se využije funkce `HOGDescriptor::compute`.

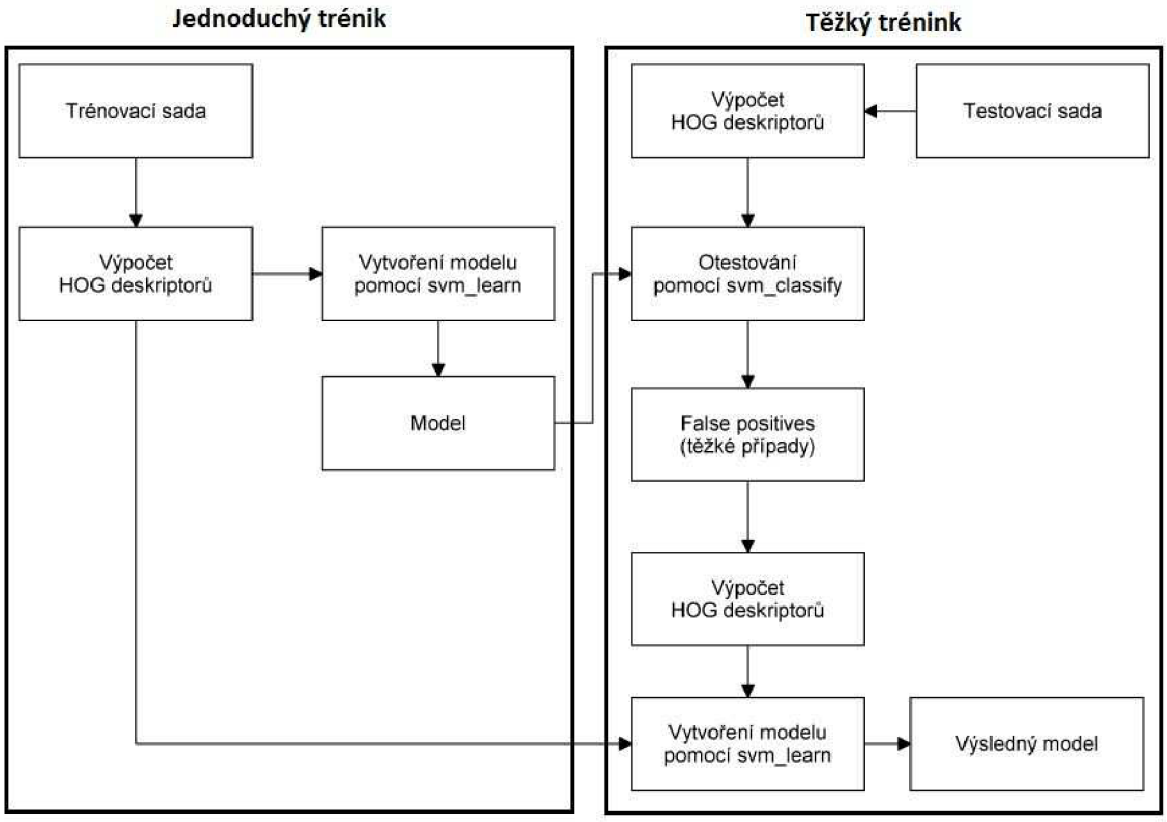
Vlastní trénování řeší program *svm_learn*, který na vstupu očekává soubor s HOG deskriptory popsaný v předchozím odstavci. Výstupem je soubor s modelem (model file). Dále je použit program *svm_classify*, jež otestuje klasifikátor pomocí testovací (evaluační) sady. Tento test spočívá v tom, že

se prochází testovací negativní sada (testovací soubor s HOG deskriptory) a aplikuje se na ni detektor. Výsledkem je soubor s příznaky, které určují, zda byl vyhodnocen obrázek z testovací sady jako kladný nebo záporný. Nás zajímá kladné vyhodnocení, které značí, že se jedná o falešnou detekci (false positive). Každý kladně vyhodnocený obrázek má svůj vektor reprezentující HOG deskriptor, který se přidá do trénovacího souboru jako tzv. „těžký“ příklad. Pokud spustíme znovu trénink, máme natrénovaný detektor. Konkrétní nastavení parametrů programů svm_learn a svm_classify lze vidět zde:

./svm_learn -v 1 -t 0 -c 0.01 FeatureTrain.txt svmLightModel.txt

./svm_classify FeatureTest.txt svmLightModel.txt FeaturePredict.txt

Výsledný soubor s modelem obsahuje velké množství vektorů, přičemž je potřeba pouze jeden. Z toho důvodu se použije skript *weightVector.pl*, který vytvoří jeden vážený vektor. Ten se pak načte pomocí metody `HOGDescriptor::setSVMDetector`. Celý tento postup trénování znázorňuje obrázek 5.2.



Obrázek 5.2: Postup trénování SVM.

6 Testování a výsledky

Tato kapitola je zaměřena na experimenty se systémem. Cílem testů je vyhodnotit různé vlastnosti jednotlivých částí systému a díky tomu nalézt taková nastavení, která nejlépe vyhovují požadavkům na rychlost a úspěšnost práce vytvořené aplikace. První část kapitoly je věnována trénovacím sadám použitých k natrénování jednotlivých detektorů. V dalších částech jsou popsány a vyhodnoceny jednotlivé testy a experimenty včetně výsledků ve formě grafů a tabulek. Tyto testy jsou zaměřené na úspěšnost detekce různých detektorů, přesnost sledování, práci aplikace jako celku, časovou náročnost jednotlivých detektorů a časovou náročnost sledovacího algoritmu.

Vzhledem k velkému množství naměřených hodnot nejsou všechny tabulky uvedeny v tomto dokumentu, ale jsou umístěné na přiloženém CD v souboru *Namerena_Data.pdf*. Výsledné grafy v této kapitole jsou vytvořené z těchto hodnot. Testovací videa jsou stažena ze serveru youtube.com a jsou rovněž umístěna na přiloženém CD.

6.1 Trénovací sady a natrénované detektory

Pro účely této práce byly vytvořeny dvě trénovací sady: *TaekwondoSet* a *FutsalSet*. Kromě nich byla použita také databáze chodců INRIA [48].

Datová sada INRIA (INRIA pedestrian dataset) je komplexní sada, která obsahuje data jak pro trénování, tak i pro následné otestování. Výhodou je, že obsahuje pozitivní i negativní vzorky pro natrénování. Pozitivní vzorky mají velikost 64x128 pixelů. U negativních vzorků je velikost vyšší a pohybuje se od rozlišení 320x240 až po 640x480 pixelů. Datová sada INRIA obsahuje velké množství snímků pro testování jak pozitivních, tak negativních. Stažená sada rovněž zahrnuje informace o anotování, které obsahují informace o poloze osoby. To se může hodit, pokud se na snímku vyskytuje více osob. Příklad obrázků nacházejících se v této sadě lze vidět na obrázku 6.1.

Obrázek 6.1: Ukázka datové sady INRIA.

Jak už z názvů vytvořených datových sad napovídá, obsahují obrázky sportovců. První sada se zaměřuje na taekwondo. Hlavní rozdíl oproti jiným sportovcům či chodcům je ten, že v tomto sportu

se nosí kimono, které je mnohem volnější než normální oděv. Druhá sada obsahuje hráče futsalu. Ukázku pozitivních snímků z obou databází lze vidět na obrázku 6.2.

Obrázek 6.2: Ukázka datové sady TaekwondoSet a FutsalSet.

Pro *TaekwondoSet* a *FutsalSet* se pozitivní sady vytvářely ručně pomocí programu *createDataset*. Jelikož databáze INRIA obsahuje velmi kvalitní negativní sadu, využila se jako negativní sada pro trénink všech detektorů s tím rozdílem, že se k nim přidalo několik desítek ručně vyřezaných negativních snímků, které byly získány z video sekvencí sportovních utkání. Testovací sada, která obsahuje negativní snímky, byla použita také z databáze INRIA a stejně jako v předchozím případě k ní byly přidány nově vytvořené snímky.

Tabulka 6.1 přehledně znázorňuje sady pro trénování včetně velikostí jednotlivých částí dané sady.

Datové sady	Počet vzorků v sadách:			
	pozitivních	negativních	negativních * 10	testovací
TaekwondoSet	1040	1329	13290	462
FutsalSet	698	1411	14110	462
INRIA	2416	1300	13000	453

Tabulka 6.1: Velikost sad pro trénování.

Negativní sada (obrázek 6.3) je se sadou testovací velmi podobná, protože se v obou případech jedná o sady negativní. Platí však, že žádné snímky nejsou totožné.

Obrázek 6.3: Negativní datová sada. Poslední dva obrázky jsou ručně vyřezané z video sekvence.

Pomocí výše zmíněných trénovacích sad bylo vytvořeno několik detektorů. V následujících testech budou tyto detektory využívány, a proto je vhodné je popsat. Přehled natrénovaných detektorů lze vidět v následující tabulce 6.2.

Detektory	Použitá sada
T64x128	TaekwondoSet
F32x64	FutsalSet
F48x96	FutsalSet
F64x128	FutsalSet
INRIA detektor	INRIA

Tabulka 6.2: Přehled natrénovaných detektorů a použitých sad.

Jak lze vidět, kromě různých trénovacích sad byly natrénovány detektory i pro různou velikost detekčního okna. To je důležité hlavně pro videa s menší velikostí hráčů, kdy větší detekční okno už není schopno menší hráče detekovat. V případě, že jsou hráči větší, nenastává žádný problém, protože detektor je pyramidový a detekční okno se postupně zvětšuje.

V tabulce jsou uvedeny pouze detektory, které vykazují rozumné výsledky a lze je použít. Při trénování byly vytvořeny i menší detektory (16x32 a 24x48), ale při otestování bylo zjištěno, že vykazují velmi špatné výsledky, a tudíž se jimi dále zabývat nebudu.

6.2 Přesnost sledování

Cílem experimentu bylo zjistit přesnost sledovacího algoritmu. Sledování objektů není vždy dokonalé, proto je třeba zvolit nějakou metriku, která bude udávat, jak moc se liší ideální sledování od skutečného.

6.2.1 Metriky pro vyhodnocení přesnosti sledování

Pro vyhodnocení přesnosti sledování se používají dva obdélníky reprezentující referenční box a box získaný pomocí sledovacího algoritmu. Pro zjištění, jak moc se boxy liší, se používají metriky F-measure a vzdálenost středů.

F-measure

F-measure je metrika, která je definována jako harmonický průměr přesnosti P (Precision) a úplnosti R (Recall) [49].

Pro odvození použijeme obecný vzorec harmonického průběhu *H*:

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \tag{6.1}$$

$$H = \frac{n}{\frac{1}{x_1} + \ldots + \frac{1}{x_n}}, \tag{6.2}$$

kde $x_1 = P$ *a* $x_2 = R$. Potom *H* bude:

$$H = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{6.3}$$

$$H = \frac{2 \cdot P \cdot R}{P + R}. \tag{6.4}$$

Nakonec zbývá vyjádřit přesnost *P* a úplnost *R*:

$$P = \frac{S_{ref\_box} \cap S_{track\_box}}{S_{track\_box}} \tag{6.5}$$

$$P = \frac{S_{ref\_box} \cap S_{track\_box}}{S_{ref\_box}}. \tag{6.6}$$

Ze vzorců lze vidět, že je nutné vypočítat obsah referenčního obdélníku S_{ref_box}, sledovaného obdélníku S_{track_box} a obsah jejich průniku.

Při ideální situaci nastává, že referenční boxy jsou stejné (stejně překryté) jako boxy vzniklé při sledování a F-measure je potom rovna jedné. Z toho lze odvodit, že čím blíže je výsledek jedné, tím je sledování přesnější.

Pokud jsou přesnost *P* a úplnost *R* podobné hodnoty, výsledek F-measure se blíží aritmetickému průměru. Jestliže jsou hodnoty více odlišné, výsledek F-measure směřuje k nižší z hodnot *P* a *R*.

Vzdálenost středů

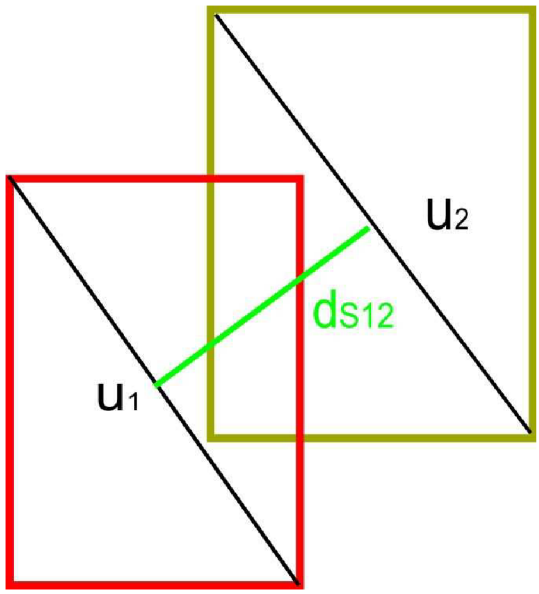
Tato metrika udává poměrnou vzdálenost středů sledovaného a referenčního obdélníku. Podstatné je, že do svého výpočtu zahrnuje velikosti obou obdélníků. Důležitost této metriky spočívá v situacích, kdy se liší velikost referenčního a sledovaného obdélníku, ale jejich středy (těžiště) jsou blízko sebe. V tomto případě dochází ke správnému sledování, ale například metrika F-measure vykazuje horší výsledky.

Vzorec pro výpočet:

$$D = \frac{1}{2} \cdot \left(\frac{d_{S12}}{u_1} + \frac{d_{S12}}{u_2} \right), \tag{6.7}$$

kde u_1 a u_2 jsou úhlopříčky obdélníků a d_{S12} je vzdálenost jejich středů. Tuto situaci ilustruje obrázek 6.4.

Může nastat situace, kdy výsledek vyjde větší než 1. To je třeba ošetřit a započítávat maximální hodnotu 1. Při ideální situaci, kdy jsou středy blízko sebe, se výsledná hodnota blíží nule.



Obrázek 6.4: Ilustrační obrázek znázorňující proměnné pro výpočet.

Přesnost sledování

Tato metrika udává poměr správně sledovaných vzorků oproti všem vzorkům.

$$P = \frac{pocet\_spravne\_sledovanych}{celkovy\_pocet} \tag{6.8}$$

Správně sledovaný vzorek je takový vzorek, který má hodnotu F-measure větší než zvolené číslo (nejčastěji 0,5 nebo 0,7).

6.2.2 Testování

Aby bylo možné výše popsané metriky vyhodnotit, je nutné nejprve anotovat referenční data. K tomuto účelu slouží program *Anotate*, který pomocí myši ukládá rozměry referenčních obdélníků. Výsledný soubor obsahuje na každém řádku číslo snímku a rozměry boxu. Každý snímek může obsahovat i více objektů, a tudíž i více boxů. V tomto případě je myší anotován jeden sledovaný objekt.

K získání informací (souřadnic obdélníku) o reálně sledovaném objektu se využije hlavní aplikace, kdy se pro každý snímek ukládají do souboru souřadnice pro zvolený sledovaný objekt. Formát výsledného souboru je stejný jako v předchozím případě.

Pokud jsou k dispozici referenční data a data obsahující reálné sledování, provede se výpočet jednotlivých metrik pomocí programu *PrecisionEval*, který už jednoduše tyto vstupní soubory zpracuje a vypočítá výsledky.

Tabulka 6.3 obsahuje výsledky testování pro několik různých video sekvencí. Řádky ukazují jednotlivé výsledky pro určité video. Sloupce udávají výsledné metriky F-measure a vzdálenost středů. Poslední dva sloupce určují poměr správně sledovaných vzorků k celkovému počtu vzorků. Všechny tyto metriky jsou popsány výše.

Při testování byly nastaveny tyto parametry:

- Počet částic: 100
- Velikost částic: 16x32 pixelů

video	F-measure	vzdál. středů	přesnost > 0,5	přesnost > 0,7
TENIS4	0,7341	0,1425	0,9911	0,6106
JAP1 – ID 5	0,7469	0,1334	0,9736	0,7631
JAP1 – ID 7	0,8052	0,0763	1,0000	0,9250
FIFA1 – ID 2	0,8022	0,0882	0,9750	0,9000
průměr	0,7721	0,1101	0,9849	0,7997

Tabulka 6.3: Výsledky obsahující jednotlivé metriky.

Tabulka 6.4 je obdobná jako předchozí, ale je naměřená pro jednu video sekvenci *TENIS4.avi* (obrázek 6.5), kdy se pro každý řádek změnily parametry sledovacího algoritmu: počet částic a velikost částic.

video: TENIS4					
Poč. částic	Vel. částice [px]	F-measure	vzdál. středů	přesnost > 0,5	přesnost > 0,7
100	16x32	0,7341	0,1425	0,9911	0,6106
50	16x32	0,6649	0,1780	0,8938	0,3805
100	24x48	0,7614	0,1268	0,9646	0,7610
50	24x48	0,7477	0,1404	0,9557	0,6902
100	32x64	0,7778	0,1190	0,9734	0,7433

Tabulka 6.4: Výsledky metrik v závislosti na změně parametrů.

Vyhodnocení

Při testování nedošlo ke ztrátě ani záměně sledovaného objektu, a proto byly vypočítány „rozumné“ hodnoty. Pokud by došlo ke ztrátě nebo záměně sledovaného objektu, výsledky by byly horší. Tato situace by vadila z toho důvodu, že pokud by například došlo ke ztrátě v prvním snímku, výsledná

přesnost by se blížila nule. Jakmile by takováto obdobná situace nastala, ovlivnila by výsledky celého testu a ten by neměl vypovídající hodnotu.

Výsledky metriky F-measure se pohybují v průměru kolem 0,77, což značí slušnou přesnost. To dokazuje i metrika vzdálenosti středů, u které je průměrná vzdálenost mezi středy při referenčním a reálném sledování 0,11.

Pokud se mění parametry u sledovacího algoritmu, mění se i jednotlivé metriky. Jelikož byly zvoleny rozumné parametry, nemění se metriky nějak zásadně. Z hodnot je ale patrný logický předpoklad, že čím větší velikost a počet částic, tím je sledování přesnější. Ukázka sledování je vidět na obrázku 6.5.

Obrázek 6.5: Ukázka sledování ve video sekvenci TENIS4.avi. U snímku číslo 82 je sledování poměrně přesné. Ukázku méně přesného sledování vidíme na snímku 44 a 107.

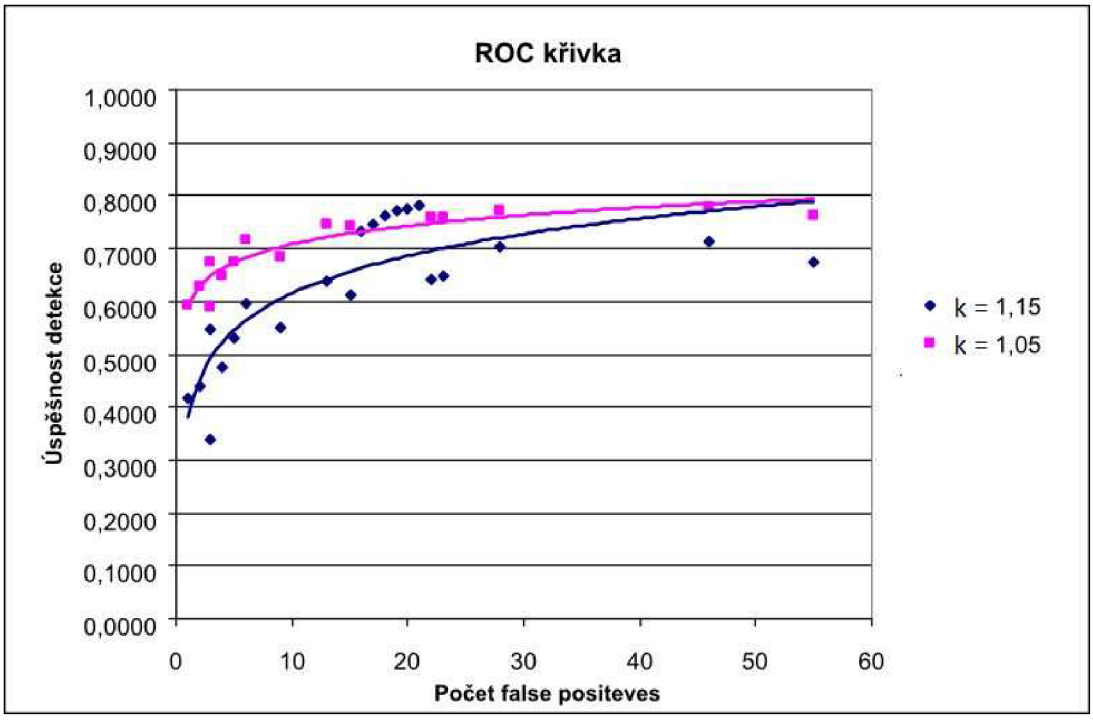
6.3 Úspěšnost detekce

Cílem tohoto testu je zjistit detekční schopnost různých detektorů. Nejčastějším výsledkem tohoto typu testů je ROC křivka, která udává úspěšnost detekce vůči falešným detekcím (false positives). Při samotné detekci lze nastavovat více parametrů, a proto bylo získáno větší množství hodnot, ze kterých je tato křivka sestavena.

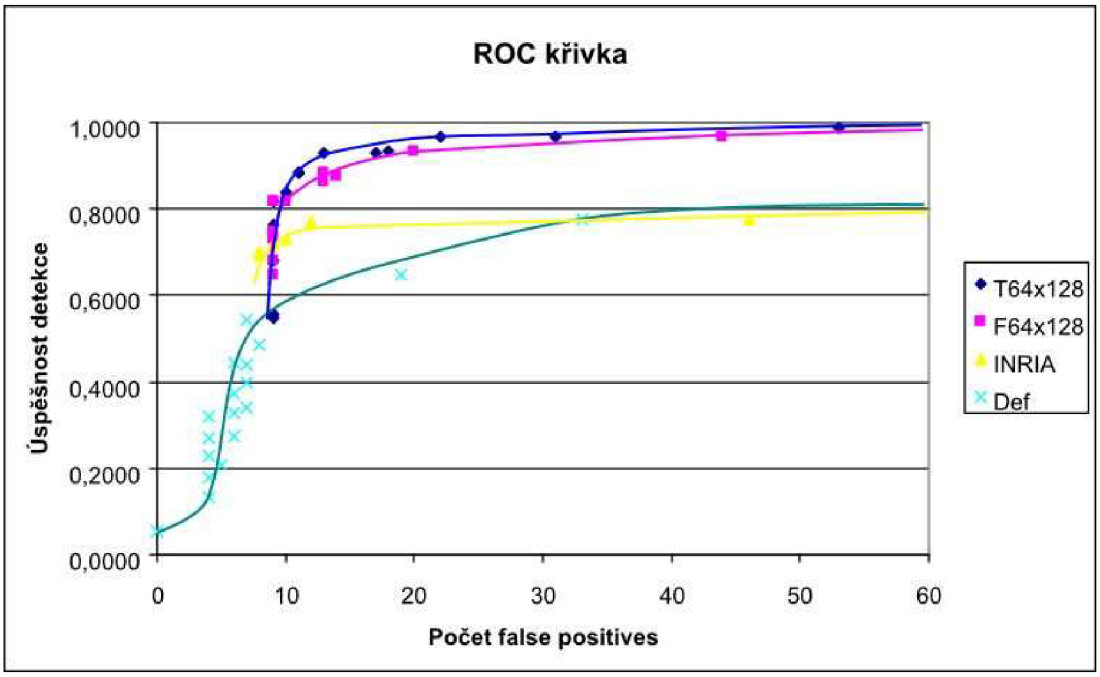
6.3.1 Testování

Jelikož se při tomto testování získává velké množství dat, je nutné tento proces řešit automatizovaně. Pro anotování referenčních obdélníků, které reprezentují detekované objekty, se opět použije program *Anotate* (popsaný výše). Samotné vyhodnocení úspěšnosti řeší program *DetectionEval*, který jako vstup očekává soubor s referenčními daty a video sekvencí, pro kterou bude vyhodnocovat úspěšnost detekce. Program pracuje tak, že porovnává referenční obdélníky s obdélníky, které jsou detekované detektorem. Tuto činnost provede pro každý snímek videa a na výstup vrátí celkovou sumu správně detekovaných hráčů, celkový počet nedetekovaných hráčů a výsledný počet falešných detekcí. Tento program je nutné spustit několikrát, aby se získaly hodnoty pro různé parametry. V tomto experimentu se mění parametry: práh, práh pro vzájemné překrývání obdélníků a koeficient zvětšování detekčního okna.

Výsledné grafy (obrázek 6.6 a 6.7) vygenerováné z naměřených hodnot, znázorňují ROC křivky. Na osu y se vynáší úspěšnost detekce a na osu x se vynáší počet falešných detekcí. Pro první graf je použit detektor F32x64 a každá ROC křivka je vypočítána pro různý koeficient zvětšení detekčního okna *k*. Pro testování byla použita video sekvence *JAP1.avi* (futsal). Druhý graf slouží k porovnání různých detektorů mezi sebou při použití video sekvence *T6_4.avi* (taekwondo).



Obrázek 6.6: ROC křivky, které jsou vygenerovány pro různý koeficient zvětšení detekčního okna k.



Obrázek 6.7: Vygenerované ROC křivky pro různé detektory.

Vyhodnocení

Z prvního grafu je patrné, že změna koeficientu zvětšení detekčního okna má vliv na úspěšnost detekce. To není překvapivé, protože čím vyšší bude tento koeficient, tím více skokově se bude měnit detekční okno a častěji budou nastávat situace, kdy detekční okno nebude souhlasit s detekovaným objektem. Další hodnoty kroku změny velikosti detekčního okna už nejsou zajímavé, protože prudce klesá detekční schopnost.

Při prohlédnutí druhého grafu, kdy se porovnávají detektory mezi sebou, bylo zjištěno, že nejlepší výsledky má detektor T64x128 (natrénováno pomocí *TaekwondoSet*). Jelikož je vstupní video pořízeno ze zápasu Taekwonda, byl tento výsledek očekáván. Na druhém místě je detektor F64x128 (*FutsalSet*), který vykazuje velmi podobné výsledky jako T64x128. Další v pořadí je detektor, který je natrénován pomocí datové sady INRIA (obrázky chodců). Výsledky jsou horší než u předchozích dvou, ale detektor je stále bez problémů použitelný. Tyto tři detektory vykazují podobné počty falešných detekcí. Pro srovnání je zde umístěn také defaultní detektor z OpenCV. Ten má detekční schopnost nižší, ale vykazuje menší počet falešných detekcí. To je důležitá vlastnost, a proto v některých případech může být vhodnější než předchozí detektory.

Na obrázku 6.8 a 6.9 lze vidět příklady správně detekovaných hráčů. Obrázek 6.10 ukazuje některé falešné či nepřesné detekce. Zelená barva značí detekovanou osobu a bílá barva označuje osobu, která byla detekována detektorem, ale jelikož leží mimo obraz, tak se zahazuje. Tyto případy mohou ovlivňovat výsledek, protože například na snímku 32 (obrázek 6.10) je detekovaný rozhodčí, který leží mimo obraz, ale v referenčních datech anotovaný není.

Obrázek 6.8: Ukázka činnosti detektoru F32x66.

Obrázek 6.9: Ukázka činnosti detektoru T64x128.

Obrázek 6.10: Ukázka falešných či nepřesných detekcí. Na snímku 6 a 11 lze vidět příklad falešné detekce, která je při testování započítána. Snímek 32 ukazuje nepřesnost detektoru, kdy je detekované okno znatelně větší než by mělo být.

6.4 Práce aplikace jako celku

Cílem tohoto testování je ukázat, jak pracuje aplikace jako celek. Nejprve dochází k rozpoznání hráče a poté následuje sledování až do doby, kdy hráč opustí scénu. Test se tedy zaměřuje na správné sledování hráče od jeho prvního výskytu ve videu.

Kromě výše zmíněného testu, zde budou ukázány a popsány zajímavé případy, které nastávají v průběhu činnosti aplikace.

6.4.1 Testování

Tato činnost bohužel nešla automatizovat, a proto bylo testování provedeno ručně. Výsledkem sledování každé osoby jsou tyto případy:

- Sledování proběhlo v pořádku a hráč byl správně sledován od začátku až do konce.
- Sledování skončilo neúspěchem, protože hráč byl ztracen.
- Sledování bylo neúspěšné, protože došlo k „přeskočení“ sledování na jiný podobný objekt.

Při tomto testu byly použity videa *JAP7.avi* a *JAP6.avi*. Tyto video sekvence jsou pořízené z futsalového utkání a vyznačují se tím, že obsahují poměrně velké množství hráčů, kteří se neustále pohybují, tudíž dochází k častému přicházení a odcházení ze scény. Výsledné hodnoty se nacházejí v tabulce 6.5. Tabulka obsahuje sloupec práh, který udává, jak velký může být rozdíl mezi referenčním histogramem a histogramem sledovaného objektu. Pokud je tento rozdíl několik snímků po sobě větší než nastavený práh, dojde k odebrání objektu. Parametr práh má rozsah od nuly do jedné.

Při testování byly nastaveny tyto parametry sledovacího algoritmu:

- Počet částic: 100.
- Velikost částic: 16x32 pixelů.

video	práh	ztraceno	převzato	odsledováno	úspěšnost
JAP7	0,15	22	2	19	44,1860
JAP7	0,25	7	5	10	45,4545
JAP6	0,15	4	3	8	53,3333

Tabulka 6.5: Naměřené výsledky.

Vyhodnocení

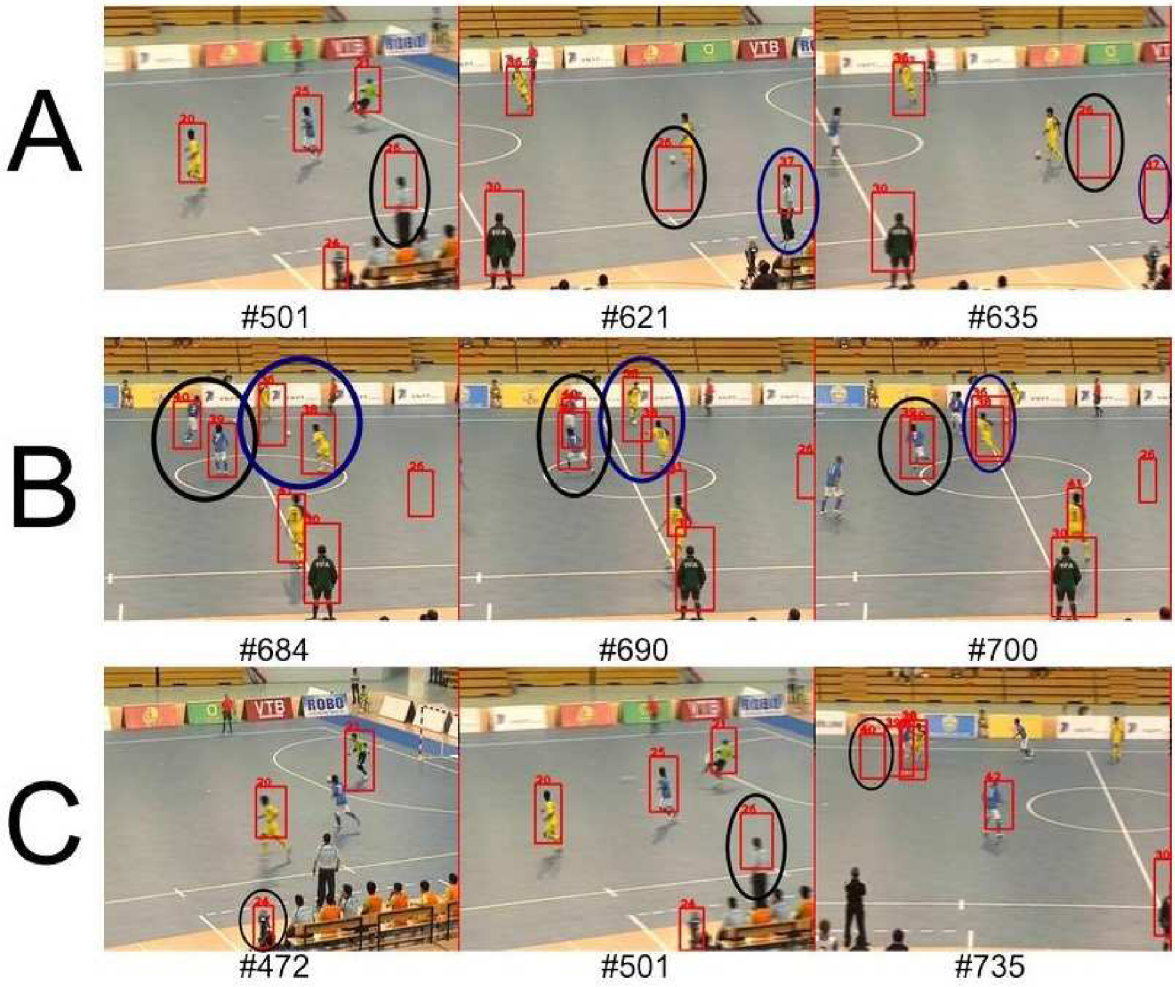
Výsledné hodnoty značně závisí na použitém videu. Pokud nedochází k rozmazání obrazu a osoby se vzájemně nepřekrývají, není problém získat výsledky, které se blíží ke 100%. K tomuto účelu je použito futsalové utkání, a proto jsou výsledky úspěšnosti sledování kolem 50%.

Pokud se zaměříme na nastavení velikosti prahu, vidíme, že při jeho zvýšení dochází k nižším ztrátám. Bohužel tento práh už je poměrně velký a někdy se sledovaný objekt neodebere, i když už odešel ze scény. Vznikají pak případy, kdy sledovací okno zůstává v obraze delší dobu, což vyvolává dojem, že program pracuje s velkou chybovostí.

Na obrázku 6.11 jsou ukázány hlavní situace, při kterých vznikají chyby. Řádek A ukazuje výše popsaný problém s neodebíráním sledovacích oken, i když už sledovaný objekt opustil scénu. Sledovací okna jsou zvýrazněna barevně (černě a modře). Tento případ lze odstranit pomocí snížení prahu pro odebírání. Snížení prahu znamená, že se budou častěji odebírat i dobře sledované osoby. Rozumným kompromisem je nastavit práh na standardní hodnotu 0,15. Problémy nastávají tehdy, když má detekovaný objekt velmi podobný histogram jako „palubovka“. Tento stav je obtížně řešitelný.

Řádek B znázorňuje stav, kdy dochází k přechodu sledovacího okna z jedné osoby na druhou (okno „přeskočí“). K této situaci dochází pouze tehdy, pokud se překryjí dva spoluhráči (mají stejné dresy), protože jejich histogramy jsou velmi podobné.

Poslední řádek, který je označen písmenem C, ukazuje falešné a nepřesné detekce. Na posledním snímku vidíme důsledek chybné detekce, kdy okno nesleduje žádný objekt.

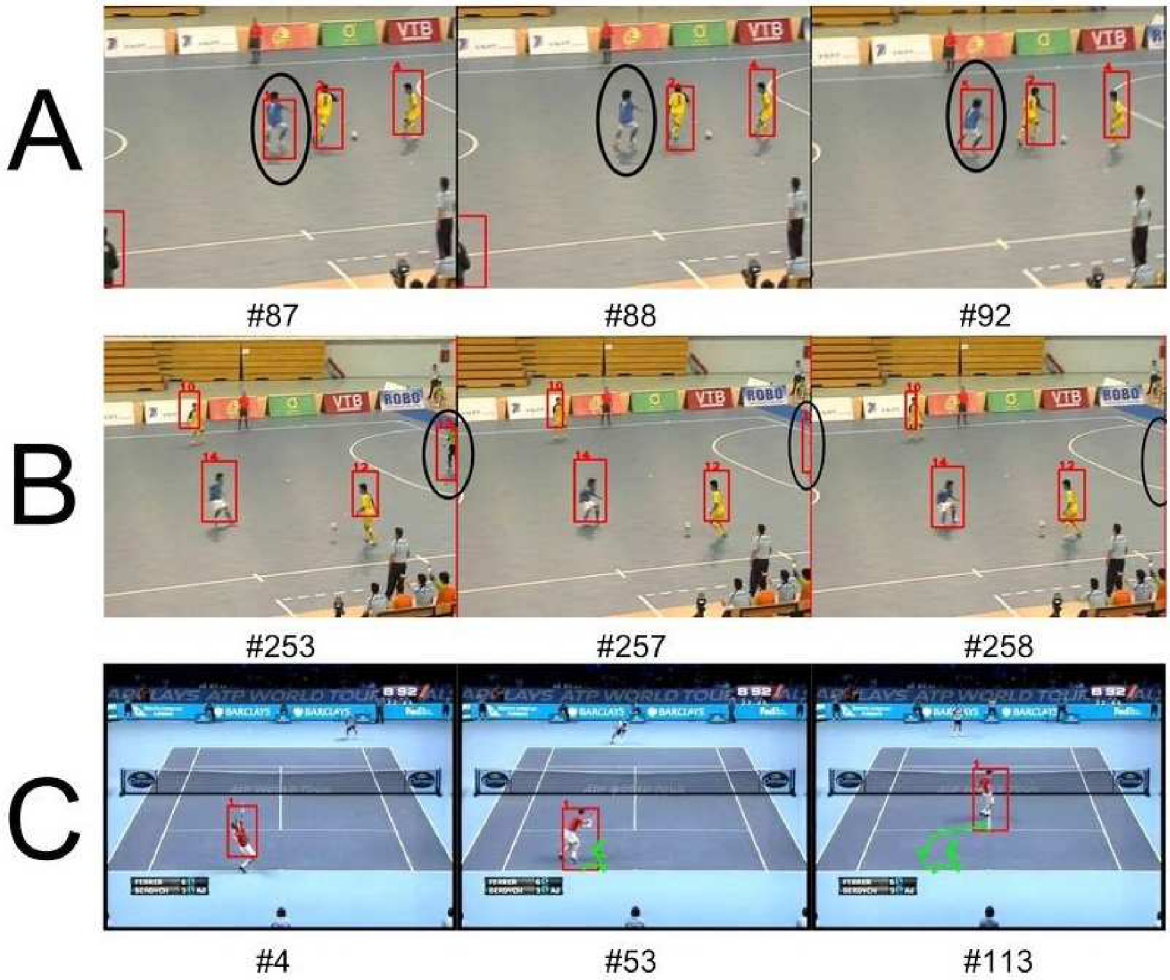


Obrázek 6.11: Ukázka činnosti programu. Obrázek se snaží ukázat chyby, které mohou při zpracování videa nastat. Jedná se o: neodebrání sledovacího okna (A), „přeskočení“ okna z jedné osoby na druhou (B) a chybná nebo nepřesná detekce (C).

Obrázek 6.12 ukazuje správnou funkci programu. První řádek s obrázky ilustruje případy, kdy došlo k chybnému odebrání objektu. Nejčastější příčinou bývá rozmazání obrazu vlivem rychlého pohybu kamery nebo hráče. Aplikace se s tímto stavem vyrovná tak, že za několik málo snímků detekuje objekt znovu a aplikace pokračuje ve své činnosti bez dalších chyb.

Správné odebrání hráče, který opouští scénu, vidíme na řádku B. Pokud je hráč dobře detekován detektorem (má histogram, který odpovídá jeho tělu), odebírání funguje bez problémů a k žádným chybám nedochází.

Poslední řádek C ukazuje příklad video sekvence natočená statickou kamerou. Tenista je správně detekován, a tudíž i jeho model je velmi kvalitní. Dále nedochází k překrývání s jinými objekty, takže nenastává žádná obtížná situace, jež by mohla způsobit chybu. Jelikož je video sekvence pořízená statickou kamerou, je zapnutá funkce trajektorie, která je postupně vykreslována zelenou barvou a značí pohyb, který hráč vykonal během sledování.



Obrázek 6.12: Ukázka správné činnosti programu hlavně při odebírání hráčů. Jedná se o: chybné odebrání a následná nová detekce (A), odebrání hráče opouštějícího scénu (B) a ukázka vykreslení trajektorie (C).

6.5 Časová náročnost

Toto testování má za úkol zjistit časovou náročnost výsledného systému. Nejnáročnější části aplikace jsou detekce a sledování. Kromě efektivního návrhu algoritmů se dá časová náročnost snížit pomocí vhodného nastavení různých parametrů. Ve většině případů je ale nutné volit kompromis mezi rychlostí a kvalitou detekce či sledování.

Faktory ovlivňující výpočetní náročnost:

- Rozlišení vstupního videa.
- Nastavení parametrů při detekci a sledování.
- Výkon počítače.

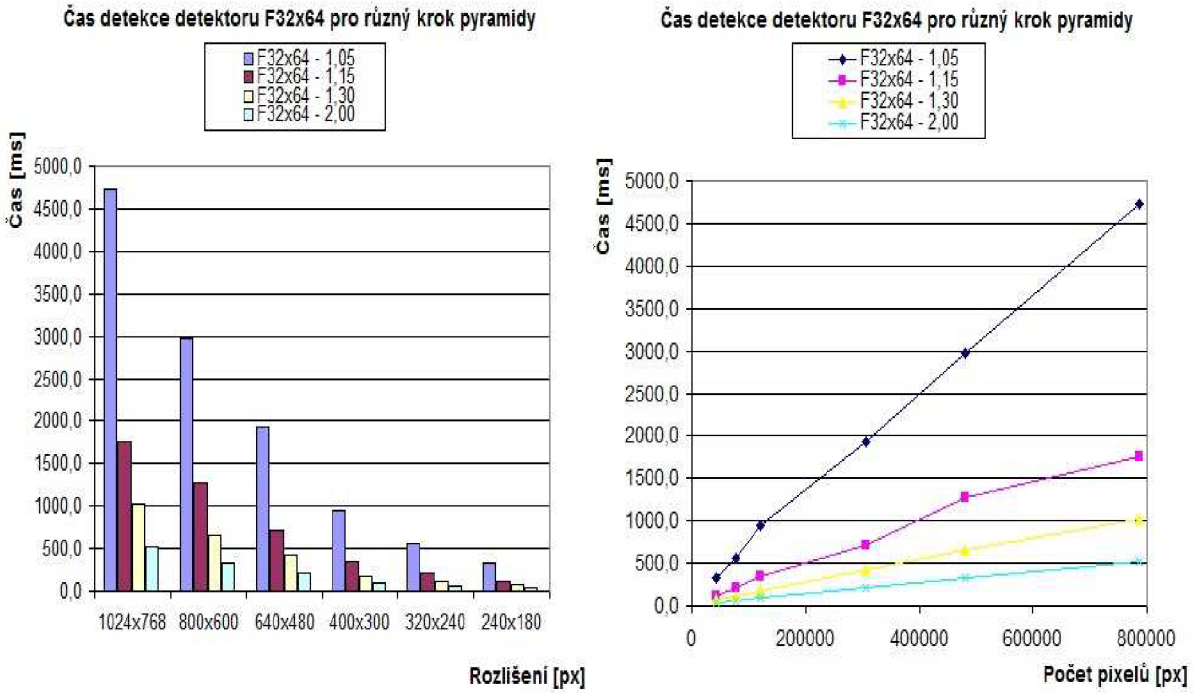
Všechna testování byla provedena na počítači AMD Athlon 64 X2 Dual-Core 1,8 GHz, 3 GB RAM.

6.5.1 Testování

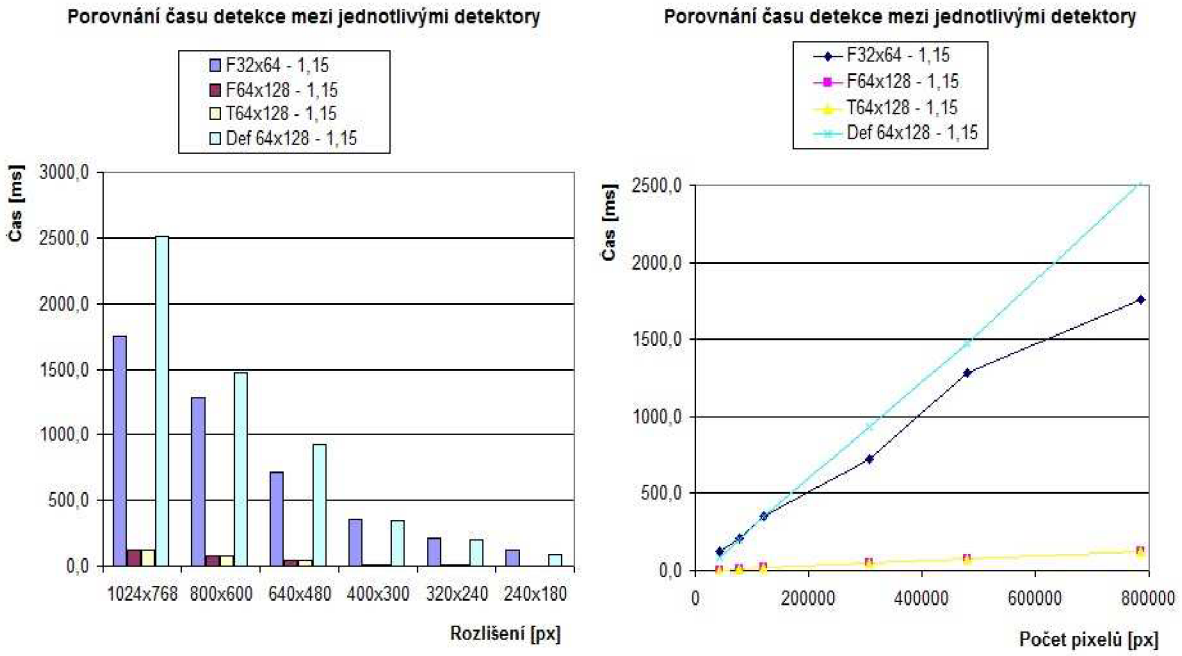
Testování bylo provedeno jednoduše tak, že se ve vhodných částech programu vložila funkce pro zachycení aktuálního času. Tuto funkci vložíme před a za část vykonávající detekci nebo sledování. Takto získané časy stačí od sebe odečíst a tím je získán výsledek. Pro zjištění času detekce byl použit jednoduchý program *TimeEval*, který pro zadaný vstupní obrázek vypočítá čas detekce pro odlišné detektory s různým nastavením. Čas sledování byl naměřen přímo v hlavní aplikaci vždy pro jeden krok sledování a výsledky byly zprůměrovány.

Časová náročnost detektoru

Obrázek 6.13 obsahuje dva grafy, které znázorňují čas detekce pro různé rozlišení respektive pro různý počet pixelů v obraze. Tato měření byla provedena pro jednotlivé koeficienty zvětšení detekčního okna *k*. Na dalších grafech (obrázek 6.14) je toto měření provedeno pro jednotlivé detektory, kdy krok zvětšení *k* je stejný.



Obrázek 6.13: Ukázka času detekce pro různý krok pyramidy k detektoru F32x64.



Obrázek 6.14: Porovnání detektorů mezi sebou dle času detekce.

Časová náročnost sledování

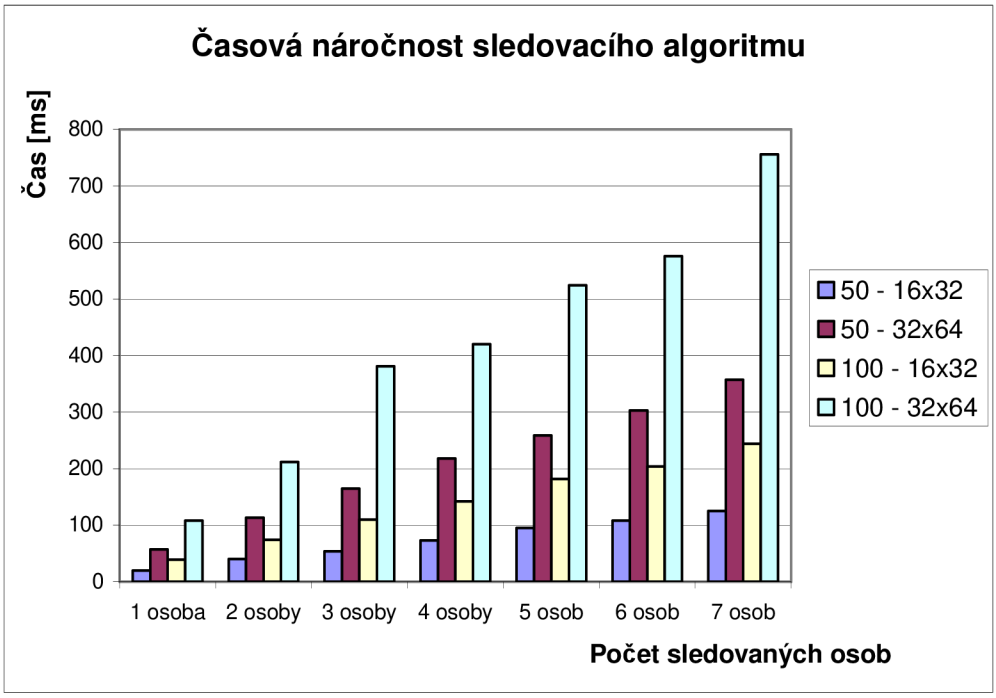
Toto měření bylo prováděno pro různý počet částic a různě velké okénko reprezentující částici. Ve videu se vyskytoval proměnlivý počet hráčů, a proto byly zaznamenány časy v závislosti na tomto počtu. Výsledné hodnoty obsahuje tabulka 6.6 a pro přehlednější porovnání je zde také graf (obrázek 6.15), který je vygenerován z těchto hodnot. Pro zjištění, kolik snímků za sekundu (FPS) je schopný sledovací algoritmus zpracovat, slouží tabulka 6.7.

Počet částic	Velikost okénka	Čas potřebný pro sledování daného počtu hráčů pro jeden snímek videa [ms]						
		1 osoba	2 osoby	3 osoby	4 osoby	5 osob	6 osob	7 osob
50	16x32	20	40	54	73	95	108	125
50	32x64	57	113	165	218	259	303	357
100	16x32	39	74	110	142	182	204	244
100	32x64	108	212	381	420	524	576	756

Tabulka 6.6: Naměřené časy potřebné pro sledování.

Počet částic	Velikost okénka	Počet snímků za sekundu [FPS] při sledování daného počtu osob						
		1 osoba	2 osoby	3 osoby	4 osoby	5 osob	6 osob	7 osob
50	16x32	50,0	25,0	18,5	13,7	10,5	9,3	8,0
50	32x64	17,5	8,8	6,1	4,6	3,9	3,3	2,8
100	16x32	25,6	13,5	9,1	7,0	5,5	4,9	4,1
100	32x64	9,3	4,7	2,6	2,4	1,9	1,7	1,3

Tabulka 6.7: Náročnost sledování vyjádřená v FPS (počet snímků za sekundu).



Obrázek 6.15: Časová náročnost sledování pro různá nastavení.

Výsledky

Nejprve se zaměříme na časovou náročnost detektoru. Z grafů vyplývá, že čas potřebný pro detekci roste lineárně se vzrůstajícím počtem pixelů. Dále lze vidět, že čím menší je koeficient zvětšení detekčního okna, tím je výpočetní doba delší. Číslo 1,15 je rozumnou volbou, při které má detektor pořád slušnou úspěšnost.

Dalším testem bylo porovnání časů detekce mezi jednotlivými detektory. Lze vidět, že nejrychleji dochází k detekci u detektorů F64x128 a T64x128. Oba detektory mají stejné nastavení, takže naměřené časy jsou velmi podobné. Mnohem více času je potřeba u detektoru F32x64. To je logické, protože počáteční velikost detekčního okna je menší, a proto se vykonává větší počet stupňů pyramidy než v předchozích případech. Nejdelší dobu detekce zaznamenal defaultní detektor. Tato informace je překvapivá, protože velikost detekčního okna je stejná jako v prvních dvou případech. Rozdíl bude pravděpodobně ve vnitřním nastavení, protože u tohoto typu se nenastavují žádné počáteční parametry. Detektor tedy používá implicitní nastavení.

Při testování časových nároků sledování bylo provedeno měření pro různé nastavení parametrů. Z výsledků vyplývá, že se vzrůstajícím počtem osob lineárně roste i potřebný čas pro sledování. Čas potřebný pro sledování vzrůstá také v závislosti na počtu a velikosti částic. Sledování v reálném čase by se dalo provádět pouze s omezeným počtem osob a také s horším nastavením parametrů.

Závěr

Cílem této diplomové práce bylo prozkoumat metody analýzy pohybu v obraze a detekce pohybujících se objektů. S využitím těchto metod byl navržen a implementován systém, který provádí detekci a následné sledování hráčů záznamu sportovního utkání.

Konkrétním zaměřením této práce byla analýza pohybu hráčů v záznamech sportovních utkání. Pro tento účel bylo nutné vybrat nejvhodnější metody. Pro detekci hráčů v obraze je použit histogram orientovaných gradientů využívající SVM klasifikátor. Sledování jednotlivých hráčů se provádí pomocí částicového filtru.

Se systémem bylo provedeno velké množství testů a experimentů. Výsledky těchto testů jsou ukázány ve formě grafů a tabulek včetně slovního popisu a vyhodnocení. V rámci práce byly vytvořeny trénovací sady pro trénink detektoru. Pomocí těchto sad bylo možné natrénovat a porovnat jednotlivé detektory mezi sebou.

Aplikace jako celek pracuje dobře. Chyby, které mohou v průběhu činnosti nastat jsou diskutovány a odůvodněny. Co se týče přesnosti sledovacího algoritmu, metrika F-measure dává průměrně výsledek 0,77, který značí slušnou přesnost. Úspěšnost a počet falešných detekcí detektoru lze vidět na ROC křivkách. Samotná detekce osob ve video sekvenci je časově náročná činnost, a proto nelze zpracovávat vstupní video v reálném čase. Pokud se zaměříme na samotné sledování pomocí částicového filtru, algoritmus je schopen sledovat až dvě osoby v reálném čase v závislosti na konkrétním nastavení.

Další vývoj práce by měl směřovat ke snížení výpočetní doby celého systému. Časově nejnáročnější je detekce a sledování. Obě tyto úlohy lze dobře paralelizovat, a proto by bylo vhodné použít technologie OpenCL či CUDA. OpenCV 2.3 již obsahuje v modulu GPU implementaci metod pro detekci objektů, které využívají architekturu CUDA. Dále by bylo možno zvýšit robustnost aplikace jako celku. Vytvoření grafického uživatelského rozhraní umožní snadné nastavení parametrů a zjednoduší práci uživateli. Další vývoj může spočívat v detekci hřiště, což umožní vytvářet statistiky o pohybu, rychlosti či trajektorii hráče.

Literatura

[1] Comaniciu D., Ramesh V., Meer P.: Real-time tracking of non-rigid objects using mean shift. In CVPR, 2000.

[2] Comaniciu D., Ramesh V., Meer P.: Kernel-Based Object Tracking. IEEE Transaction on Pattern Analysis and Machine Intelligence, May 2003.

[3] Comanciu D., Meer P.: Mean Shift: A Robust Approach, Toward Feature Space Analysis. 2002.

[4] cvr.yorku.ca [online]. c2010 [cit. 2012-01-01]. Computer Vision Related Notes. Dostupné z WWW: <http://cvr.yorku.ca/members/gradstudents/kosta/compvis>.

[5] Bouguet Y.: Pyramidal implementation of the lucas kanade feature tracker, Intel Corporation, Microprocessor Research Labs, Tech. Rep., 2000.

[6] Welch G., Bishop G.: An Introduction to the Kalman Filter. Chapel Hill, NC, USA, University of North Carolina at Chapel Hill, 2006.

[7] Hongling W., Bo Y., Guodong T., Aidong T.: Object tracking by applying mean-shift algorithm into particle filtering. 2nd IEEE International Conference on Broadband Network Multimedia Technology, 2009. IC-BNMT '09.

[8] Maskell S., Gordon N.: A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking. IEEE Transactions on Signal Processing, 2001.

[9] Juránek R.: Rozpoznávání vzorů v obraze pomoci klasifikátorů. Diplomová práce, Brno, FIT VUT v Brně, 2007.

[10] cc.gatech.edu [online]. c2009 [cit. 2012-01-01]. Computer Vision Final Project. Dostupné z WWW: <http://www.cc.gatech.edu/~kihwan23/imageCV/Final2005/FinalProject_KH.htm>.

[11] Ruiz-del-Solar J.; VERSCHAE R.: Object detection using cascades of boosted classifiers. [online]. 2006, [cit. 2012-01-22]. Dostupný z WWW: <http://www.die.uchile.cl/ieee-cis/files/RuizdelSolar_T9.pdf>.

[12] Webb A. R.: Statistical Pattern Recognition, second edition. Butterworth Heinemann, 2002.

[13] Spáčil P.: Detekce a rozpoznávání dopravních značek v obraze, diplomová práce, Brno, FIT VUT v Brně, 2011.

[14] Yilmaz A., Javed O., Shah M.: Object tracking: A Survey. ACM Computing Surveys, s. 1-45, 2006.

[15] Lu W.: Tracking and recognizing actions of multiple hockey players using the boosted particle filter. University of British Columbia, Department of Computer Science, 2008.

[16] Ogale A. N.: A survey of techniques for human detection from video. University of Maryland, Department of Computer Science, 2006.

[17] Matečný A.: Detekcia postáv na obrázkoch. Bakalářská práce, Brno, FI MU v Brně, 2011.

[18] Dalan N.; Triggs B.: Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005.

[19] Wu B., Nevatia R.: Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. In ICCV '05 Proceedings of the Tenth IEEE International Conference on Computer Vision, 2005.

[20] Bertozzi M., Broggi A., Del Rose M.: A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. 2007.

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

[22] Mohan A., Papageorgiou C., Poggio T.: Example-Based Object Detection in Images by Components. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001.

[23] Gavrila D., Giebel J.: Shape-based pedestrian detection and tracking. IEEE Intelligent Vehicle Symposium, 2002.

[24] gavrila.net [online]. c2006 [cit. 2012-01-03]. The Chamfer System. Dostupné z WWW: <http://www.gavrila.net/Research/Chamfer_System/chamfer_system.html>.

[25] Ngo V., Yang W., Cai J.: Accurate Playfield Detection Using Area-of-Coverage. Centre for Multimedia and Network Technology, Nanyang Technological University, 2006.

[26] Wren C., Azarbayejani A., Darrell T., Pentland A.: Pfinder: real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.

[27] Beleznai C., Fruhstuck B., Bischof H.: Human detection in groups using a fast mean shift procedure. International Conference on Image Processing, 2004.

[28] Haga T., Sumi K., Yagi Y.: Human detection in outdoor scene using spatio-temporal motion analysis. International Konference on Pattern Recognition, 2004.

[29] Eng H., Wang J., Kam A., Yau W.: A bayesian framework for robust human detection and occlusion handling using a human shape model. International Conference on Pattern Recognition, 2004.

[30] Elzein H., Lakshmanan S., Watta P.: A motion and shapebased pedestrian detection algorithm. IEEE Intelligent Vehicles Symposium, s. 500–504, 2003.

[31] Toth D., Aach T.: Detection and recognition of moving objevte using statistical motion detection and fourier descriptors. International Conference on Image Analysis and Processing, s. 430–435, 2003.

[32] Lee D., Zhan P., Thomas A., Schoenberger R.: Shape-based human intrusion detection. SPIE International Symposium on Defense and Security, Visual Information Processing XIII, 2004.

[33] Zhou J., Hoang J.: Real time robust human detection and tracking system. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[34] Yoon S., Hyunwoo K.: Real-time multiple people detection using skin color, motion and appearance information. International Workshop on Robot and Human Interactive Communication, s. 331–334, 2004.

[35] Xu F., Fujimura K.: Human detection using depth and gray images. EEE Conference on Advanced Video and Signal Based Surveillance, s. 115–121, 2003.

[36] Li L., Ge S.: Object-oriented scale-adaptive filtering for human detection from stereo images. IEEE Conference on Cybernetics and Intelligent Systems, 2004.

[37] Han J., Bhanu B.: Detecting moving humans using color and infrared video. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2003.

[38] Jiang L., Tian F.: Perceptual-based fusion of ir and visual images for human detection. International Symposium on Intelligent Multimedia, Video and Speech Processing, s. 514-517, 2004.

[39] Cutler R., Davis S. L.: Robust real-time periodic motion detection, analysis, and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

[40] Utsumi A., Tetsutani N.: Human detection using geometrical pixel value structures. Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 39 s., 2002.

[41] Viola P., Jones J. M., Snow D.: Detecting pedestrians using patterns of motion and appearance. IEEE International Conference on Computer Vision, 2003.

[42] Sidenbladh H.: Detecting human motion with support vector machines. Proceedings of the 17th International Conference on Pattern Recognition, 2004.

[43] Papageorgiou C., Evgeniou T., Poggio T.: A Trainable Pedestrian Detection System. In Proceedings of Intelligent Vehicles, s. 241-246, 1998.

[44] Mikolajczyk K., Schmid C., Zisserman A.: Human Detection Based on a Probabilistic Assembly of Robust Part Detectors. In Computer Vision - ECCV 2004, s. 69-82, 2004.

[45] opencv.willowgarage.com [online]. c2012 [cit. 2012-04-01]. OpenCV. Dostupné z WWW: < http://opencv.willowgarage.com/wiki>.

[46] code.google.com [online]. c2010 [cit. 2012-04-02]. opencvx. Dostupné z WWW: <http://code.google.com/p/opencvx>.

[47] svmlight.joachims.org [online]. c2010 [cit. 2012-03-01]. Support Vector Machine. Dostupné z WWW: <http://svmlight.joachims.org>.

[48] pascal.inrialpes.fr [online]. c2005 [cit. 2012-02-02]. INRIA Person Dataset. Dostupné z WWW: <http://pascal.inrialpes.fr/data/human>.

[49] Sasaki Y.: The truth of the F-measure. School of Computer Science, University of Manchester, 2007.

[50] Karlsson R., Bergman N.: Auxiliary Particle Filters for Tracking a Maneuvering Target, Department of Electrical Engineering, Linkoping University, 2000.

[51] Duan L., Farmer C., Moroz I.: Regularized Particle Filter with Langevin Resampling Step, In International Conference of Numerical Analysis and Applied Mathematics - ICNAAM 2010, s. 1080-1083, 2010.

[52] Kotecha J., Djuric P.: Gaussian particle filtering, IEEE Transactions on Signal Processing, s. 2592 – 2601, 2003.

[53] Merwe R., Doucet A., Freitas N., Wan E.: The Unscented Particle Filter, Engineering Department, Cambridge University, 2000.

[54] Doucet A., Godsill S., Andrieu C.: On sequential Monte Carlo sampling methods for Bayesian filtering, Statistics and Computing, s.197-208, 2000.

[55] Yuan Z., Zheng N., Jia X.: The Gauss-Hermite Particle Filter, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, 2003.

[56] Bugallo M., Maiz S., Miguez J., Djuric P.: Cost-Reference Particle Filters and Fusion of Information, Department of Electrical and Computer Engineering, Stony Brook University, 2009.

[57] Sarkka S., Vehtari A., Lampinen J.: Rao-Blackwellized Particle Filter for Multiple Target Tracking, Helsinki University of Technology, 2005.

[58] Isard M., Blake A.: CONDENSATION—Conditional Density Propagation for Visual Tracking, Department of Engineering Science, University of Oxford, 1997.

Seznam příloh

Příloha 1. CD/DVD obsahující:

- Zdrojové soubory.
- Soubor Návod.pdf obsahující návod k instalaci a použití programů.
- Technickou zprávu bakalářské práce ve formátu PDF v souboru xnemec26_dp.pdf.
- Technickou zprávu bakalářské práce ve formátu DOC v souboru xnemec26_dp.doc.
- Plakát.