VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SLEDOVÁNÍ LIDSKÉ POSTAVY VE VIDEOSEKVENCI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

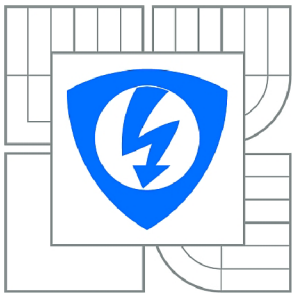
AUTOR PRÁCE
AUTHOR

Bc. MICHAL PLAČKO

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SLEDOVÁNÍ LIDSKÉ POSTAVY VE VIDEOSEKVENCI

MONITORING OF HUMAN BODY IN VIDEOSEQUENCE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

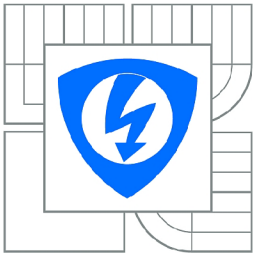
AUTOR PRÁCE — Bc. MICHAL PLAČKO

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR ČÍKA, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Michal Plačko ***ID:*** 115257
Ročník: 2 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Sledování lidské postavy ve videosekvenci

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte současné algoritmy pro sledování lidské postavy a gest ve videosekvenci. Vyberte vhodný způsob detekce a navrhněte, jakým způsobem ho budete implementovat v programovacím jazyce JAVA. Vytvořte aplikaci, ve které Vámi navržený algoritmus implementujte.

DOPORUČENÁ LITERATURA:

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

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 28.5.2014

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

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

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práca sa zaoberá problematikou detekcie ľudskej postavy a sledovania gest vo videosekvencii. Po obecnom popise spracovania videosekcencie sú bližšie popísané metódy detekcie ľudskej postavy reprezentované významnými prácami. Najviac pozornosti je upriamenej na detekciu pomocou reálneho AdaBoostu s využitím Haar-like čŕt a detekciu pomocou reálneho AdaBoostu s využitím Edgelet čŕt.
Praktická časť diplomovej práce sa najprv venuje výberu metódy. Metóda implementovaná v tejto práci je detekcia pomocou reálneho AdaBoostu založenom na Haar-like črtách. Ďalej sú popísané možnosti spracovania videosekvencie v programovacom jazyku JAVA a odôvodnenie výberu knižnice OpenCV spolu s wrapperom JavaCV, ktorý je v práci použitý. Nakoniec je popísaný samotný program, popis grafického rozhrania, ako i popis a funkcionalita jednotlivých tried.

KĽÚČOVÉ SLOVÁ

Detekcia ľudskej postavy, Haar-like features, Reálny AdaBoost, JavaFX, videosekcencia, Vitruviov muž

ABSTRACT

This thesis deals with human body detection and gestures tracking in videosequences. First, processing of videosequences in general is described. Further, different methods of human body detection are described and represented by significant papers. The most of the attention is focused on detection by real AdaBoost algorithm based on Haar-like features and Edgelet features.
The practical part starts with selection of method that is implemented in this thesis. This method is detection by real AdaBoost based on Haar-like features. Further, different options of videosequence processing in JAVA are researched with justification of choice OpenCV library with JavaCV wrapper, which is used in this thesis. In the end, application itself is described, including description of GUI and description of each class and its functionality.

KEYWORDS

Human body detection, Haar-like features, Real AdaBoost, JavaFX, videosequence, Vitruvian man

PLAČKO, Michal *Sledování lidské postavy ve videosekvenci*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 55 s. Vedúci práce bol Ing. Petr Číka, PhD.

PREHLÁSENIE

Prehlasujem, že som svoju diplomovú prácu na tému „Sledování lidské postavy ve videosekvenci“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právech súvisejících s právem autorským a o změně některých zákonů (autorský zákon), v znení neskorších predpisů, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. díl 4 Trestního zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

// PODĎAKOVANIE

Ďakujem vedúcemu diplomovej práce Ing. Petrovi Číkovi, PhD. za metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady, ktoré som využil pri tvorbe diplomovej práce. Ďalej ďakujem mojej sestre Andrei, ktorá mi pomáhala pri finálnej korektúre gramatických chýb a preklepov. V neposlednom rade ďakujem mojím rodičom za podporu počas celého štúdia.

Brno . . . .

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Technicka 12, CZ-61600 Brno
Czech Republic

http://www.six.feec.vutbr.cz

Výskum popísaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených z projektu SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Cieľom diplomovej práce bolo oboznámiť sa s dostupnými metódami rozpoznávania ľudskej postavy vo videosekvencii, vybrať a preštudovať jednu metódu a následne implementovať túto metódu v programovacom jazyku Java.

Človek si dnešný svet snáď ani nevie predstaviť bez výpočtovej techniky, ktorá je všade okolo nás a ľudia ju využívajú na zjednodušenie svojich životov. Ešte donedávna boli počítače len stroje, ktorých najväčšou výhodou bola rýchlosť, čím dokázali ušetriť ľudom kopec času. Nastal však moment, kedy len rýchlosť nestačila a od výpočtovej techniky sa začala vyžadovať aj istá dávka samostatnosti. To viedlo k vytvoreniu umelej inteligencie, ktorá má umožniť výpočtovej technike dáta nielen spracovávať, ale naučiť sa vnímať a analyzovať rozličné situácie a rozhodovať sa na základe rôznych podnetov. Jedná sa o simuláciu ľudského správania.

Teória umelej inteligencie je rozsiahly vedný odbor, ktorý obsahuje mnoho podoblastí. Vývoj v týchto podoblastiach nie je ani zďaleka rovnaký. Bežnou súčasťou sofistikovaných textových editorov je korekcia textu, ktorá dokáže užívateľa upozorňovať nielen na gramatické, ale aj na štylistické chyby, a v prípade zistenia chyby navrhuje aj možné riešenia. Rovnako je na tom aj spracovanie zvuku vďaka metódam spracovania, ktoré dokážu z audio záznamu zdetekovať emócie v hlase, či prepisovať hovorený text do elektronickej podoby. Spracovanie textu a zvuku je teda na pomerne vysokej úrovni a nastal čas zamerať sa na časť umelej inteligencie známu ako „počítačové videnie“. Jedná sa o spracovanie obrazu, či už statického, v podobe obrázkov, alebo dynamického, v podobe videosekvencií.

Aktuálne je spracovanie obrazu na vzostupe a je to kľúčový krok k dosiahnutiu komfortnej interakcie medzi počítačom a človekom. Na trhu už sú prvé inteligentné televízory, ktoré môžu byť ovládané gestami. Počítačové videnie má však omnoho viac využití, ako len zvýšenie komfortu bežných ľudí. Rozpoznávanie ľudskej postavy vo videosekvencii sa môže využívať na zvýšenie bezpečnosti, či už na letiskách, alebo vo väzení, či napríklad na inteligentné riadenie dopravy, kedy sa sleduje počet osôb na priechode pre chodcov.

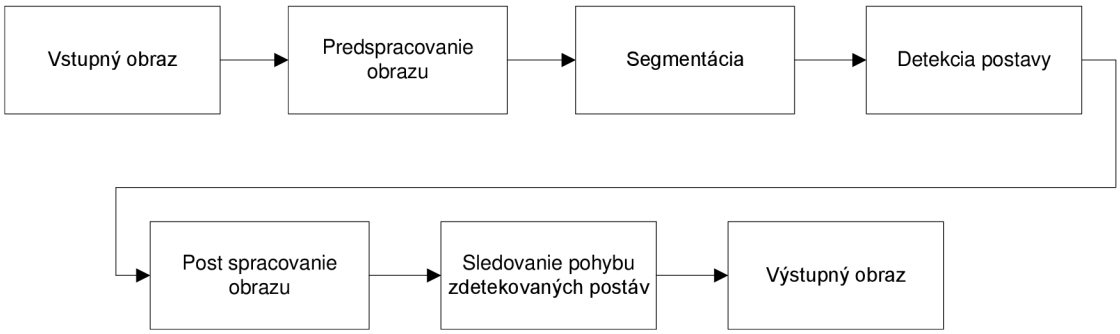
1 VIDEO SEKVENCIA

Počítačové spracovanie dynamického obrazu (videa), bolo dlhšiu dobu pozadu oproti spracovaniu obrazov statických (obrázkov). Bolo totiž komplikované v reálnom čase spracovávať veľké množstvo obrázkov. Postupom času a zvyšovaním výkonnosti výpočtovej techniky však tento problém zanikol a nastal priestor na analýzu a realizáciu problematiky „počítačového videnia“. Počítačovo nie je možné vnímať video ako nepretržitý sled obrázkov. Využíva sa tu, rovnako ako vo filmovom priemysle, nedokonalosť ľudského oka, konkrétne jeho zotrvačnosť. Pri dopade svetla o určitej intenzite na ľudské oko sú vybudené nervové zakončenia, zvané tyčinky, ktoré určitú dobu po tomto osvietení nevnímajú razantný pokles intenzity dopadajúceho svetla. Ak teda na oko dopadajú svetelné lúče v pravidelných intervaloch, mozog to nevníma ako skokové zmeny jasu, ale ako svetlo s ustálenou intenzitou. Obdobným spôsobom je realizovaná ilúzia videa pomocou počítača, či televízie. Jedná sa o video sekvenciu, teda sled podobných obrázkov, ktoré sú zobrazované dostatočne rýchlo na to, aby ich ľudský mozog vnímal vďaka nedokonalosti ľudského zraku ako jednoliate video. Aby v ľudskom mozgu vznikol dojem, že sleduje jednoliate video, a nie sériu obrázkov, je nutné vysielať minimálne 25 snímkov za sekundu. Táto hodnota sa využíva aj v televíznom priemysle.

1.1 Spracovanie video sekvencie

Ak teda uvažujeme, že bude spracovávaná video sekvencia, pozostávajúca zo sledu obrázkov, môžeme na ich spracovanie uplatniť metódy spracovania statických obrazov. Celé rozpoznávanie ľudskej postavy vo video sekvencii je potom možné popísať pomocou blokovej schémy na obrázku 1.1.

Na začiatku je pochopiteľne vstupný obraz. Jedná sa o video súbor načítaný z pevného disku, alebo o videosekvenciu prijímanú z webkamery. Po získaní video sekvencie sú jej jednotlivé snímky upravené. Jedná sa napríklad o odstránenie šumu, zmenu veľkosti snímkov, či rozpitie pixelov z dôvodu dosiahnutia kvalitnejšej segmentácie. Segmentácia má za úlohu rozdeliť snímok ako celok na skupinu segmentov, pričom sa bude jednať o pohybujúce sa objekty a nepohyblivú scénu. Pohyblivé objekty sa odovzdajú detektoru ľudskej postavy, ktorý rozhodne, či tieto



Obr. 1.1: Bloková schéma detekcie ľudskej postavy vo video sekvencii.

objekty sú, alebo nie sú ľudské postavy. Na základe tohoto rozhodnutia sa zrealizuje post spracovanie snímkov, kde dôjde k označeniu zdetekovaných postáv v pôvodnej video sekvencii. Pohyb jednotlivých zdetekovaných postáv je zaznamenávaný z dôvodu ďalšieho spracovania, napríklad rozpoznania gest. Na výstupe je vstupný obraz, ktorý je doplnený o označenie ľudskej postavy vo video sekvencii. Všetky bloky medzi vstupným a výstupným obrazom sú v práci postupne popísané. Bloku Detekcie je venovaná samostatná kapitola.

1.1.1 Predspracovanie obrazu

Obraz, ktorý človek dokáže vnímať ako súbor rôznych predmetov, je počítačom vnímaný len ako zhluk pixelov. Predspracovanie obrazu je potrebný krok k tomu, aby mohol počítač „vnímať“ obraz podobne ako ho vníma ľudské oko. Jedná sa o celoplošné úpravy obrazu, medzi ktoré patrí napríklad odstránenie šumu, rozpitie pixelov, či aplikácia rôznych filtrov. Táto operácia umožní následné rozdelenie obrazu na zaujímavé časti, teda pohyblivé objekty, o ktorých chceme zistiť, či sa jedná o ľudské bytosti, a na časti nezaujímavé, teda na statickú scénu.

1.1.2 Segmentácia obrazu

Obecne je segmentácia definovaná ako proces delenia obrazu do častí, ktoré korešpondujú s objektmi nachádzajúcimi sa v obraze. Jedná sa teda o zoskupovanie jednotlivých pixelov do segmentov, pričom každý segment reprezentuje jeden objekt v obraze. Segmentácia je jedna z najdôležitejších operácií pri spracovaní obrazu. Výsledky segmentácie sú následne predkladané klasifikátoru, ktorý rozhodne, či daný

objekt, reprezentovaný segmentom, je alebo nie je ľudská postava.

V súčasnosti existuje mnoho segmentačných algoritmov, pričom vhodnosť použitia jednotlivej techniky určuje ďalšie spracovanie. Delenie segmentačných techník, uvedené v práci [19], je následovné:

- **Metódy detekcie hrán** - metódy orientované na detekciu významných hrán v obraze. Na detekciu lokálnych hrán sa používajú hranové detektory, ktoré pracujú na základe rozdielu hodnôt v okolí pixelu.
- **Metódy nárastu oblastí** - v princípe sú rovnaké ako metódy detekcie hrán. Ak je možné zdetekovať hrany, tak by teoreticky mali tieto hrany ohraničovať oblasti, ktoré sú nájdené pomocou metódy nárastu oblastí. V praxi však pri použití metód nárastu oblastí dochádza k odlišným výsledkom segmentácie ako pri použití metód detekcie hrán, pretože kontúry objektov môžu byť porušené, alebo nemusia ohraničovať celý objekt.
- **Statické metódy**- v prípade týchto metód sa využívajú statické informácie o obraze, najčastejšie hodnoty jednotlivých pixelov. Informácie o štruktúre obrazu sa zvyčajne zanedbávajú.
- **Hybridné metódy** - V prípade hybridných metód segmentácie sa využívajú prvky viacerých vyššie uvedených metód, a preto nie je možné ich zaradiť do jednej z týchto kategórií. Medzi hybridné metódy patria taktiež metódy založené na matematickej morfológii, kedy sa pre segmentáciu využívajú matematické charakteristiky obrazu, ako napríklad priebeh gradientu.
- **Vedomostné (knowledge-based) metódy** - v prípade týchto metód sa využíva báza vedomostí o objektoch získaná z predošlých spracovávaných obrazov, ako je napríklad tvar, farba, či štruktúra. Tieto informácie o objektoch môžu segmentáciu značne uľahčiť. Pri týchto metódach sa využívajú predlohy segmentovaných objektov. Predlohy vznikajú pri procese trénovania, prípadne môžu byť vkladané ručne, na základe ľudských skúseností. Segmentácia prebieha transformáciou známych objektov a ich porovnávaním s objektmi nájdenými v obraze.

1.1.3 Post spracovanie obrazu

Tento blok má za úlohu označiť zdetekované ľudské postavy v pôvodnej video sekvencii na základe výsledkov detekcie, čím vznikne výstupná video sekvencia.

1.1.4 Sledovanie zdetekovaných postáv

V prípade, že sa vo video sekvencii zdetekuje ľudská postava, zaznamená sa poloha významných bodov, teda hlavy, končatín a trupu pre každú zdetekovanú postavu zvlášť. Tým vznikne postupnosť, ktorá nesie informáciu o pohybe postavy. Porovnaním tejto postupnosti so súborom známych postupností je možné určiť, či postava vykonáva nejakú akciu alebo gesto.

2 METÓDY DETEKCIE ĽUDSKEJ POSTAVY

V súčasnosti existuje množstvo metód, ktoré sa používajú na detekciu ľudskej postavy. Všetky tieto metódy je v princípe možné rozdeliť podľa prístupu na

- **prístup založený na škvrnách** - Tento prístup je výpočetne jednoduchý a veľmi efektívny. Je založený na segmentácii objektov v popredí snímanej scény a následné oddelenie od pozadia scény [25]. Výhodou metód založených na tomto prístupe je využitie v zaľudnených situáciách, v ktorých často dochádza k prekrývaniu ľudských postáv. Na druhej strane nevýhodou sú scény s premenlivým pozadím, či rýchlymi zmenami svetelných podmienok.
- **prístup založený na vlastnostiach ľudskej postavy** - Na rozdiel od prístupu založenom na škvrnách, tento prístup je výpočetne omnoho náročnejší. Jeho využitie však nie je zamerané len na detekciu ľudských postáv vo video sekvencii. Často sa využíva na detekciu aj v statických obrázkoch. Základom prístupu je model ľudského tela vytvorený jeho výraznými črtmi.
- **prístup založený na detekcii tváre** - Tento prístup zakladá detekciu celej ľudskej postavy len na detekcii tváre, pretože tvár je charakteristickou črtou ľudskej postavy. Neriešia sa prípady, kedy tvár nie je viditeľná.

2.1 Detekcie založené na škvrnách v popredí

Tieto detekcie využívajú extrakciu objektov v popredí snímku od pozadia. Jednou z výrazných prác, zaoberajúcou sa týmto typom detekcie je [25]. V tejto práci bol predstavený optimalizovaný model segmentácie človeka v popredí snímku. Navyše sa v okolí každého segmentu hľadá silueta ľudskej hlavy, čo značne zvyšuje už aj tak výkonné detekcie založené na škvrnách v popredí. Pre sledovanie postáv je nutné pridanie detektorov pre ďalšie časti ľudského tela, ako bolo realizované v práci [24]. Metódy založené na extrakcii škvŕn v popredí je vhodné využívať na sledovanie viacerých osôb, najčastejšie vo verejných, zaľudnených priestoroch, pričom priestor je snímaný statickou kamerou. Vhodnejšie je použitie vo vnútorných priestoroch.

2.2 Detekcie založené na vlastnostiach postavy

Detekcie založené na vlastnostiach ľudskej postavy obsahujú priame detekčné metódy. Priame metódy operujú nad jednotlivými snímkami získanými z video sekvencie, pričom každý snímok klasifikujú ako obsahujúci/neobsahujíci ľudskú postavu. Na základe čŕt a funkcií, ktoré sú použité pri klasifikácii jednotlivých snímkov, môžeme tieto techniky rozdeliť na techniky

- využívajúce tvar (vo forme kontúr, či iných popisovačov),
- využívajúce farbu (detekcia farby ľudskej pokožky),
- využívajúce pohyb,
- využívajúce kombinácie vyššie uvedených.

V tab. 2.1 môžeme vidieť prehľad použitých modelov a klasifikátorov pre významné práce v tomto odvetví. Podrobnejší popis jednotlivých prác sa nachádza v [13].

Autori, práca	**Použitý model**	**klasifikátor**
Cutler, R.; Davis, S., L., [2]	Periodiský pohyb	pohybová podobnosť
Utsumi, A.; Tetsutani, N., [20]	Geometrická hodnota pixelu	vzdialonostný
Gavrila, M., D.; Giebel, J., [7]	predloha tvaru	chamferová vzdialenosť
Viola, P.; Jones, J., M.; Snow, D., [22]	tvar + pohyb	AdaBoost kaskádny
Sidenbladh, H., [17]	Optický tok	SVM (RBF)
Dalal, N.; Triggs, B., [3]	Histogram gradientov	SVM lineárny

Tab. 2.1: Prehľad metód detekcie vo vybraných prácach [13].

Pre detekciu ľudskej postavy sa vo vyššie uvedených prácach používajú najmä tvary ľudskej postavy, či jej jednotlivých častí. Pre detekciu vo video sekvencii je možné použiť aj informácie o pohybe v kombinácii s tvarmi ľudskej postavy, či tvarmi jednotlivých končatín. Niektoré detektory, ktoré boli v minulosti prezentované, sa snažili na detekciu ľudskej postavy používať farebnosť, avšak tieto detektory nedosahovali dostatočne vysokú úspešnosť. V súčasnosti sa najviac využívajú Support Vector Machine (SVM) detektory, či rôzne varianty AdaBoost detektorov.

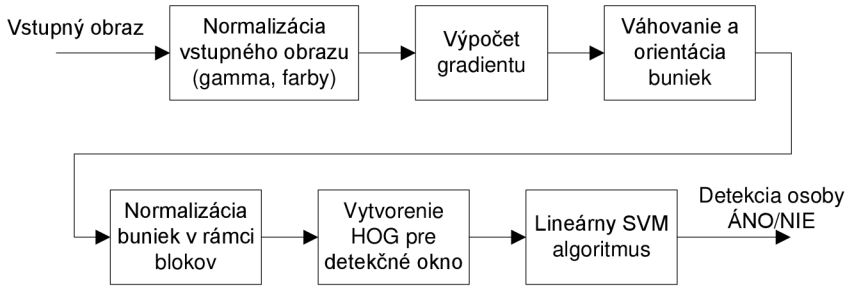
2.2.1 Support Vector Machine (SVM) detektory

Jedná sa o metódu strojového učenia. Úlohou SVM je nájsť nadrovinu, ktorá optimálne rozdeľuje trénovacie data. Optimálnou nadrovinou rozumieme, že body ležia v opačných polpriestoroch a hodnota minimálnych vzdialeností bodov od roviny je čo najväčšia. Okolo nadroviny je teda čo najširší pruh bez bodov. Na popis nadroviny teda stačia najbližšie body, ktorých je obvykle málo. Tieto body sa nazývajú „podporné vektory“, v angličtine „support vectors“ [18].

Dôležitou súčasťou detektorov SVM je jadrová transformácia (kernel transformation) priestoru príznakov do priestoru transformačných príznakov, typicky vyššej dimenzie. Táto jadrová transformácia umožňuje previesť pôvodne lineárne neseparovateľnú úlohu na lineárne separovateľnú, na ktorú je možné aplikovať optimalizačný algoritmus pre nájdenie ďalšej nadroviny [18].

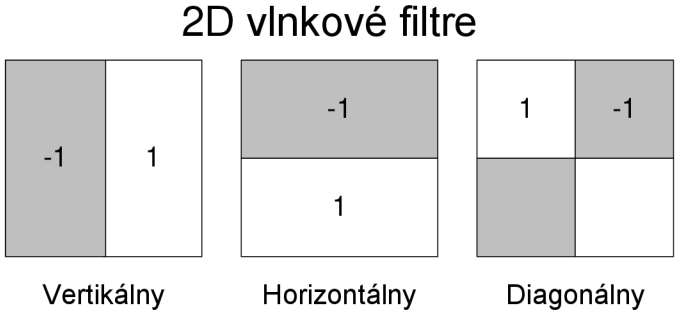
Dôležitou je práca [3], v ktorej sa prvýkrát podarilo pomocou histogramu orientovaných prechodov, v skratke HOG, z anglického *Histogram of Oriented Gradients*, zaznamenať výrazný pokrok pri detekcii ľudskej postavy. HOG od tej doby zaznamenal omnoho vyššiu úspešnosť detekcie (detection rate) ako ďalšie známe metódy. HOG metóda spočíva vo výpočte histogramu prechodov medzi prekrývajúcimi sa blokmi v okne detektora. Následne sa vypočíta vážený rozdiel medzi intenzitami dvoch prekrývajúcich sa blokov. Výsledkom je výsledná hodnota v histograme. Po normalizácii je histogram spracovaný pomocou SVM algoritmu, ktorý klasifikuje vstup dvoma možnými výstupmi, konkrétne *pozitívny* a *negatívny*. Celý postup spravovania snímku je uvedený na obrázku 2.1. Jedná sa o celkom zdĺhavý proces spracovania, a preto vzniklo niekoľko pokusov zvýšenia rýchlosti detekcie. Medzi podstatné pokusy patrí určite práca [26]. Aj napriek úspešnému pokusu, ktorý je uvedený v tejto práci, vznikali a vznikajú stále nové modifikácie detektorov [12, 1] využívajúce HOG.

SVM algoritmus je využívaný aj v práci [14]. V práci sa ďalej využívajú 2D vlnkové črty v posuvnom okne detektora o počiatočnej veľkosti 64 x 128 pixelov. Filtre troch základných smerov sú na obrázku 2.2. Tieto filtre sa aplikujú na jasovú zložku snímku, kde hľadajú rôzne krivky a vlnky. Následne sa SVM algoritmu na klasifikáciu dodávajú nájdené vlnky, ktoré sú ovplyvnené znalosťami autorov tak, aby boli vybrané vlnky s najväčšou viditeľnosťou. Tým sa dosiahne, že budú vybrané najpodstatnejšie črty v snímku.



Obr. 2.1: Postup spracovania pomocou HOG metódy [3].

Ďalšou prácou, ktorá využíva SVM algoritmus, v spolupráci s funkciami Haarovej vlnky je práca [11]. Táto práca je významná, pretože tu nebol použitý detektor na nájdenie ľudskej postavy ako celku, ale viacero detektorov pre jednotlivé časti ľudského tela. Jedná sa o detektor pre hlavu a ramená, detektor pre nohy a po jednom detektore pre každú ruku. Výsledným spojením týchto detektorov vzniká kombinovaný detektor, u ktorého je podstatne menší pomer nesprávnej detekcie, ako



Obr. 2.2: Základné filtre 2D wave [15].

u jednotlivých dielčich detektorov. Problémom v tejto práci boli rozmanitejšie tvary ľudského tela, pri ktorých bolo použitie statických Haarových funkcií nie príliš vhodnou voľbou pre detekciu ľudskej postavy.

2.2.2 Porovnávanie šablón

Metódy porovnávajúce šablóny porovnávajú vstupné dáta so sadou šablón pomocou transformačnej techniky, ktorá prepočítava vzdialenosť vstupu a šablóny v priestore

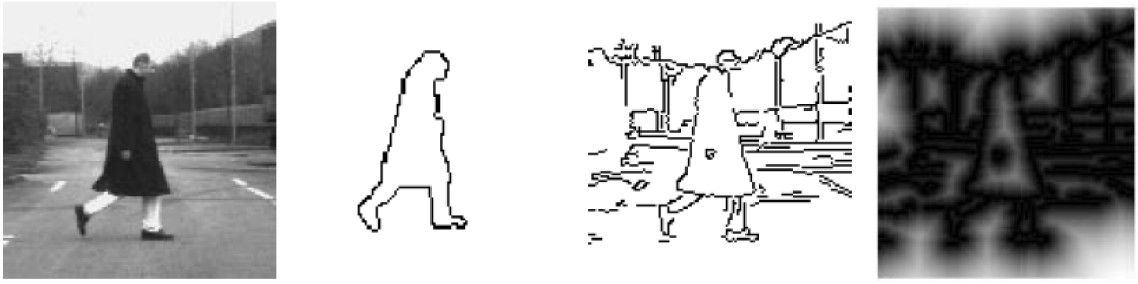
čŕt či popisovačov. V práci [8] je metóda založená na porovnávaní šablón tvarov a vstupného obrázka, ktorý je prevedený do binárneho priestoru čŕt tvoreného detekovanými hranami. Porovnávanie pozostáva z výpočtu Chamferovej vzdialenosti medzi šablónou a transformovaným obrázkom. Transformovaný obrázok je získaný pomocou distančnej transformácie [4]. Chamferova vzdialenosť je definovaná vzťahom

$$D_{chamfer}(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t), \tag{2.1}$$

kde T je priestor čŕt šablón, $|T|$ je počet všetkých čŕt v T a $d_I(t)$ označuje vzdialenosť medzi črtou t y T a k nej najbližšia črtou transformovaného obrázka I. Výsledkom je priemerná vzdialenosť čŕt šablóny a obrázka [8].

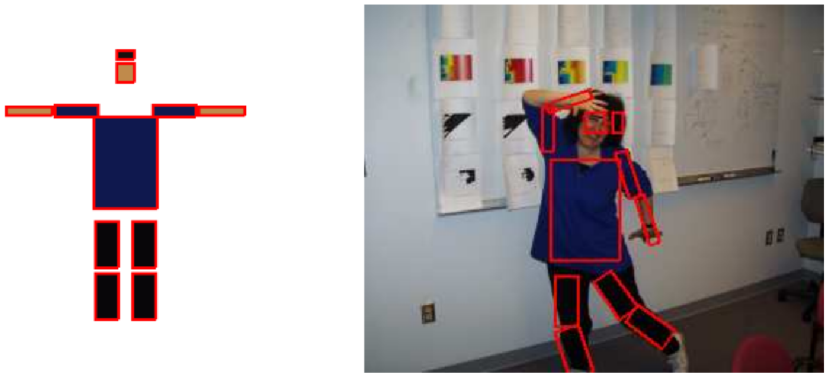
Každá šablóna zo sady šablón je transformovaná, čo znamená, že môže byť posunutá, pootočená, či ináč škálovaná a je považovaná za nájdenú v snímku, ak výsledná hodnota Chamferovej vzdialenosti nepresiahne vopred stanovený prah.

Na obrázku 2.3 je naznačený priebeh metódy pri typickom použití detekcie hrán. Úplne vpravo je na obrázku výsledok distančnej transformácie hrán. Svetlejšie pixle reprezentujú väčšiu vzdialenosť od najbližšej hrany. Metóda používa transformovaný obrázok namiesto detekovaných hrán. K dispozícii je teda hladšia funkcia prechodov farieb, ktorá umožňuje väčšiu variabilitu šablóny. To je spôsob, ako uľahčiť hľadanie vhodných transformačných parametrov pre šablónu.



Obr. 2.3: Obrázky použité pri výpočte Chamferovej vzdialenosti. Zľava: Originálny obrázok, šablóna, hrany obrázka, transformovaný obrázok [3].

Felzenszwalb a Huttenlocher v práci [5] používajú detekciu založenú na obrazovom modely človeka, ktorý sa skladá z obrazových modelov jednotlivých častí ľudského tela. Tieto modely častí tela sú následne kombinované do obrazových štruktúr pomocou definície možných spojení všetkých konfigurácií. Autori použili na vyjadre-

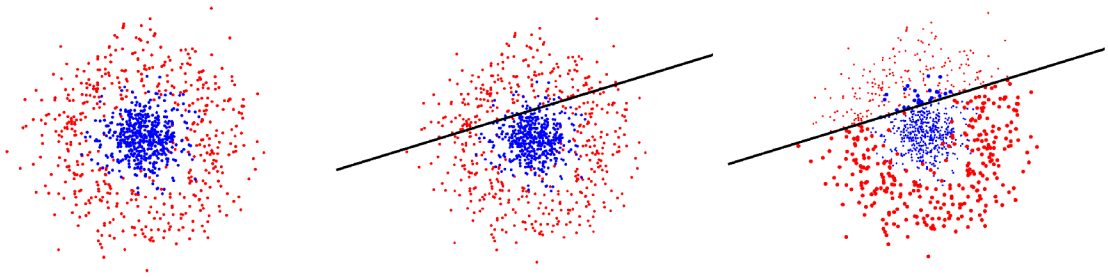


Obr. 2.4: Obrazový model človeka. Vľavo počiatočná konfigurácia, vpravo najlepšie odpovedajúca konfigurácia [5].

nie konfigurácie modelov častí tela graf a následne definovali problém výberu najlepšej konfigurácie ako problém minimalizácie tejto konfigurácie a maximalizácie miery podobnosti konfigurácie k testovanému obrázku. Na obrázku 2.4 je vidieť princíp funcionality. Táto metóda je jednoduchá, no pre jej realizáciu je nutné mať veľkú sadu obrazových modelov pre všetky časti ľudského tela, ktoré majú byť detekované [5].

2.2.3 Využívanie AdaBoost

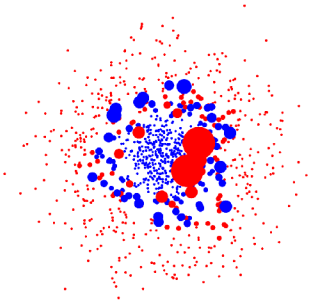
AdaBoost je algoritmus strojového učenia, ktorý bol predstavený Fraundom a Shapireom v roku 1996. Jedná sa o meta-algoritmus, ktorý je možné použiť s inými učiacimi algoritmami. Názov AdaBoost je skratkou pre Adaptive Boosting [6]. Boosting v názve hovorí o tom, že sa jedná o vytvorenie silného klasifikátora kombináciou väčšieho množstva slabých klasifikátorov. Slabé klasifikátory sú založené práve na jednej črte. AdaBoost si kladie za úlohu nájsť tie klasifikátory, ktoré rozdeľujú tréningovú sadu dát s čo najmenším váženým pomerom správne a nesprávne klasifikovaných príkladov. Po nájdení všetkých nesprávne klasifikovaných príkladov z tréningovej sady sa zvýši váhový koeficient o určitú hodnotu, ktorá ovplyvňuje hľadanie nasledujúcich hypotéz. Príklad rozdelenia tréningovej sady je uvedený na obrázku 2.5. Zvýšením váhového koeficientu sa hypotézy viac sústredia na klasifikovanie príkladov, ktoré boli predchádzajúcimi hypotézami klasifikované ako nesprávne. Kombináciou slabých hypotéz adaptívnou metódou do silného klasifikátora je možné klasifikovať



Obr. 2.5: Zobrazenie tréningovej sady v podobe bodov v jednotlivých iteráciách hľadania slabých hypotéz. Vľavo počiatočný stav, uprostred moment nájdenia prvej slabej hypotézy, vpravo stav po prevážení nesprávne klasifikovaných príkladov. Väčšie body označujú väčšiu váhu[9].

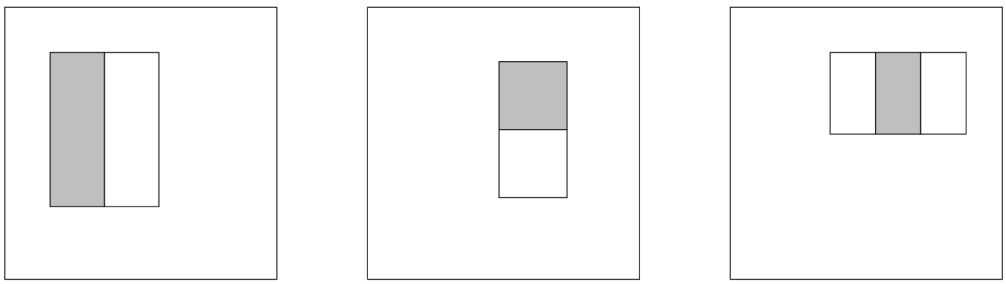
omnoho zložitejšie problémy ako pomocou jednotlivých slabých hypotéz.

Silný klasifikátor je konštruovaný lineárnou kombináciou najlepších slabých hypotéz. V každej iterácii sa najlepšia hypotéza pridá do silnejšieho klasifikátora. Počet iterácií pre hľadanie slabých hypotéz pre silné klasifikátory môže byť až v radoch stoviek, či tisícov, a je závislý na zložitosti predmetu detekcie. Tréningovú sadu po niekoľkých desiatkach iterácií môžeme vidieť na obrázku 2.6. Na tomto obrázku je vidieť, že niektoré body tréningovej sady sú stále nesprávne klasifikované a teda ich váha rastie. V priebehu ďalšej iterácie bude vybraná tá hypotéza, ktorá bude správne klasifikovať tréningové príklady s väčšími váhami, čo má za následok rýchlejšie dosiahnutie potrebného detekčného pomeru a fakt, že jednotlivé hypotézy sa vzájomne dopĺňajú.



Obr. 2.6: Rozloženie váh v tréningovej sade po niekoľkých iteráciách[9].

V roku 2001 bol prvýkrát predstavený framework detekcie objektov, ktorý pracuje v reálnom čase. Jeho tvorcami sú páni Viola a Jones. Je síce deklarované, že môže byť použitý na detekciu akýchkoľvek objektov, no primárne sa využíva na detekciu tváre. Okrem použitia AdaBoost sa v [21] využíva integrálny obraz, kaskáda silných klasifikátorov a ako črty pre slabé hypotézy sú použité Haar-like features. Integrálny obraz je algoritmus, ktorý umožňuje rýchly výpočet súčtu hodnôt pixelov v obrázku, prípadne v časti obrázku. Pre každý pixel sa spočíta súčet všetkých pixelov, ktorých súradnice nie sú väčšie ako súradnice daného pixelu.

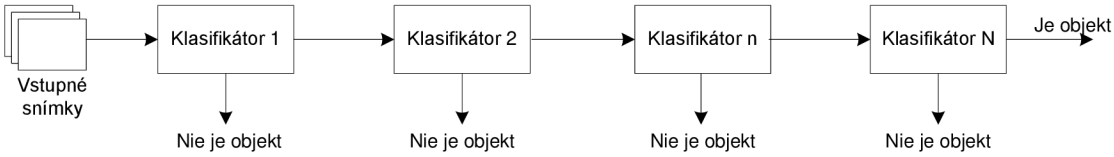


Obr. 2.7: Príklady Haar-like features [22].

Pre účely tréningu a detekcie sú použité Haar-like features. Haar-like features (obr. 2.7) sú črty obdĺžnikového tvaru, ktoré sú podobné Haarovým vlnkám. Ich výhodou je, že v spolupráci s integrálnym obrazom je výpočet hodnôt konštantný pre každú veľkosť.

V ďalšej fáze sú silné klasifikátory postupne zoradené do kaskády (obr. 2.8). Účelom tejto kaskády je znížiť pomer nesprávnych detekcií na minimum a udržať relatívne vysokú úspešnosť detekcie. Pri detekcii je kaskáda v podstate retazec klasifikátorov, pričom klasifikátory na začiatku tohoto reťazca riešia jednoduché problémy a ich úlohou je rýchlo odfiltrovať nevhodné objekty. Zložitosť problémov sa v nasledujúcich klasifikátoroch zväčšuje. Objekty, ktoré sú posledným klasifikátorom označené ako pozitívna detekcia, sú považované za nájdené objekty.

Táto metóda sa stala obľúbenou, no napriek tomu, že by mala slúžiť na detekciu ľubovoľných obejktov, jej primárnym využitím ostala detekcia tváre. Žiaľ, samotní autori priznali, že Haar-like features nie sú vhodné na detekciu ľudskej postavy, pretože je príliš obtiažné pomocou nich popísať rozmanitosť tvarov ľudského tela.

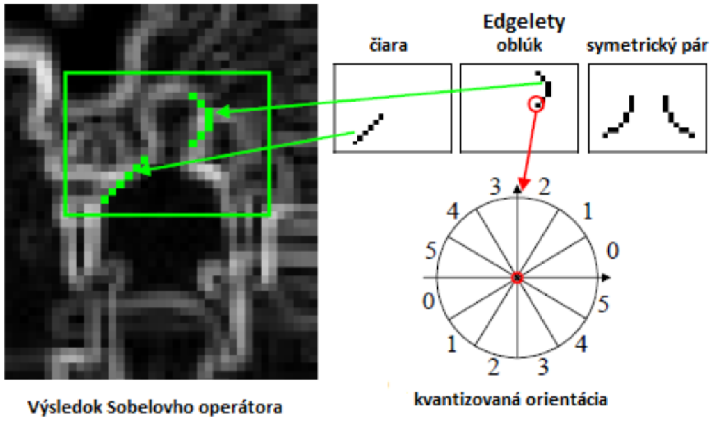


Obr. 2.8: Klasifikátory usporiadané do kaskády [22].

S vylepšením prišli Wu a Nevartia v práci [23], keď predstavili metódu, kde kombinujú viacero dobrých vlastností z rôznych dovtedy známych metód. Pri detekcii využívajú reprezentáciu človeka nie ako celku, ale ako jednotlivé časti: hlava, ramená, trup, nohy, celé telo. Metóda je tvorená dvoma častami. Prvá je detekcia častí ľudského tela a druhá je ich kombinácia. Ako črty bola použitá nová sada čŕt orientovaných na detekciu siluet. Tieto črty sa nazývajú Edgelet features, alebo edglety. Edgelety sú krátke čiarky, alebo krivky. Možu byť tvorené jednoduchými čiarami, osminami kružníc, štvrťkružnicami, polkružnicami a ich symetrickými pármi. Symetrický pár je spojenie jedného edgeletu s jeho zrkadlovým obrazom. Použitím reálneho AdaBoostu [16] sú na základe edgeletov vytvorené slabé hypotézy, ktoré sú kombinované do silných klasifikátorov. Reálny AdaBoost je vylepšením pôvodného diskrétneho AdaBoostu. Hlavnou zmenou je upravenie slabých hypotéz tak, že počítajú aj mieru spoľahlivosti, s akou klasifikujú dané dáta a objekty. Rozdiel pri tvorbe kaskády z klasifikátorov je v tom, že každý klasifikátor okrem prvého prevezme slabé hypotézy predchadzádzajúceho klasifikátora a pri detekcii najprv použije tieto hypotézy. Následne použije svoje vlastné hypotézy.

S ďalším zaujímavým prístupom prišli v roku 2005 páni Micilotta, Ong a Bowden v práci [10]. V práci bol implementovaný AdaBoost objektový detektor z práce [21], pomocou ktorého boli natrénované dielčie detektory pre tvár, torzo, nohy a ruky. Tieto dielčie detektory sú aplikované na celý obraz a následne sa pomocou algoritmu RANCAS v reálnom čase tvorí výsledná detekcia ľudskej postavy vhodnou kombináciou výsledkov jednotlivých dielčích detektorov [10].

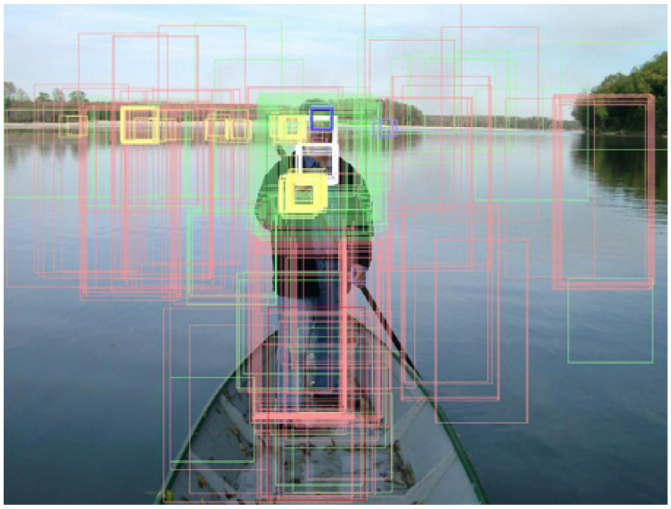
Táto práca ma zaujala hlavne kvôli využitiu svetoznámej kresby daVinciho Vitruviovho muža, ktorý bol v tejto práci využitý na približné určenie vzťahov medzi jednotlivými zdetekovanými časťami. Princíp rozmiestnenia bude ďalej v tejto práci popísaný, nakoľko som princíp Vitruviovho muža pre určenie rozmiestnenia zdete-



Obr. 2.9: Príklad nájdených odpovedajúcich edgeletov [23].

kovaných častí využil aj vo svojej práci.

Za nevýhodu tejto práce považujem aplikáciu dielčích detektorov na celý skúmaný obraz, čo viedlo pri určitých detektoroch k veľkému počtu falošne označených pozitívnych detekcií, ako je možné vidieť na následujúcom obrázku. Tieto nesprávne označené detekcie boli redukované pomocou podobnosti farby kože a pravdepodobnosťou výskytu jednotlivých častí [10].



Obr. 2.10: Nesprávne označené detekcie [10].

3 REALIZÁCIA

Pred samotnou implementáciou bolo nutné rozumne zvoliť metódu, ktorá bude implementovaná. Premyslieť, ako má aplikácia vo výsledku vyzerať, na čo má slúžiť a čo by od nej užívateľ mal očakávať. Výsledná aplikácia je tvorená pre statickú scénu, v ktorej bude jedna ľudská postava. Program dokáže detekovať aj viac, ako jednu postavu, rovnako tak vyhodnocovať gestá viacerých zdetekovaných postáv, no nie je primárne stavaný na detekciu prekrývajúcich sa postáv, či spracovávanie scén zo zaľudnených priestorov. Pri pozitívnej detekcii ľudskej postavy (zdetekovaná tvár a torzo vo vzájomnom vzťahu popísanom ďalej v tejto práci) dôjde pri prípadnom zdetekovaní rúk k sledovaniu ich pohybu. Zaznamenané postupnosti pohybov jednotlivých rúk sa pravidelne vyhodnocujú a pri pozitívnej detekcii gesta sa vykoná vopred stanovená akcia.

3.1 Výber metódy

Pri výbere metódy, ktorá bola implementovaná v programovacom jazyku Java som zvažoval dve možnosti. Metódu založenú na Edgelet features, ktorá je najlepšie popísaná v práci [23], alebo rokmi overenú metódu, ktorá sa dodnes úspešne využíva pri detekcii tváre, teda metódu využívajúcu Haar-like features, ktorej autormi sú páni Viola a Jones.

Po načrtnutí finálnej podoby aplikácie som si uvedomil jednu skutočnosť, ktorá bola kľúčová pri výbere metódy. Vychádzam z predpokladu, že človek, ktorý bude pred kamerou stáť a bude chcieť zadať gesto, bude ku kamere otočený tvárou. Z tohoto dôvôdu som sa rozhodol implementovať metódu rozpoznávania tváre pomocou Haar-like features. Autori síce uvádzajú, že táto metóda nie je vhodná pre detekciu ľudskej postavy, no je tým myslené detekciu ľudskej postavy ako celku. Rozhodol som sa využiť túto metódu a program postaviť na detektore tváre, ktorý bude základom pre zostavovanie ľudskej postavy z výsledkov dielčích detektorov za pomoci pomocných algoritmov, ktoré sú v práci podrobnejšie popísané.

3.2 Výber prostredia

Pre prácu s programovacím jazykom Java bolo vybrané prostredie Eclipse verzie Kepler. Na rozdiel od predchádzajúceho semestrálneho projektu došlo k zmene knižnice pre prácu s videosekvenciou. knižnica Xuggle bola nahradená implementáciou OpenCV do programovacieho jazyka Java. Táto knižnica je známa ako JavaCV. Dôvodom tejto zmeny bola hlavne omnoho lepšia dostupnosť k akýmkoľvek informáciám, či už na rôzných diskusných fórach, alebo priamo na stránkach OpenCV, kde je dostupná prehľadná a zrozumiteľná dokumentácia s ukážkami jednotlivých funkcií.

3.3 Kolekcie obrázkov

Pre vytvorenie klasifikátorov bolo nutné vytvoriť jednotlivé tréningové kolekcie obrázkov. Každá kolekcia musí obsahovať pozitívne obrázky, na ktorých sa nachádza predmet záujmu (v tomto prípade časť ľudského tela, ktorá má byť detekovaná) a negatívne obrázky, na ktorých pre zmenu objekt záujmu byť nesmie. V súčasnej dobe existuje niekoľko kolekcií, ktoré sú určené na účely strojového učenia. V niektorých kolekciách sa nachádzajú obrázky celých ľudských postáv, v niektorých len jednotlivé časti tela. Záleží od toho, na aký účel bola tá ktorá kolekcia vytvorená. Rovnako sú dostupné obrázky s jednou osobou, ako aj obrázky zo zaľudnených priestorov. Najznámejšie a najviac využívané kolekcie obrázkov sú v tabuľke 3.1.

V tejto práci boli využité kolekcie Dailmer, MIT chodci, MIT tvár a Cambrigde kolekcia rúk. Ako negatívna trénovacia kolekcia pre všetky klasifikátory bola vybraná kolekcia negatívnych obrázkov z Dailmer databázy obsahujúca 3600 negatívnych obrázkov vytvorených z kamery namontovanej na automobile. Pozitívna sada pre tréning tvárového klasifikátora pozostávala z kolekcie MIT tvár. Táto kolekcia obsahuje 2429 obrázkov tváre, doplnených o 400 obrázkov tváre z kolekcie Cambrigde tvár. Pozitívnou kolekciou pre torzo a nohy sa stala MIT kolekcia chodcov, obsahujúca 924 obrázkov ľudí. Pre trénovanie ruky výborne poslúžila kolekcia Cambridge ruky, ktorá obsahuje okolo 50 000 obrázkov ruky, v rôznych polohách, pri rôznom osvetlení. Z tejto kolekcie bolo vybraných 8906 obrázkov ruky.

Kolekcia	Obsah kolekcie	URL adresa
CAVIAR	videozáznamy ľudí z rôznych pohľadov	http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/
Daimler	pozitívne obrázky chodcov, negatívne obrázky	http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/daimler_pedestrian_benchmark_d.html
ETH	pozitívne obrázky chodcov	http://www.vision.ee.ethz.ch/~aess/dataset/
INRIA	pozitívne a negatívne obrázky ľudí	http://pascal.inrialpes.fr/data/human/
MIT chodci	pozitívne obrázky, čelné pohľady	http://cbcl.mit.edu/software-datasets/PedestrianData.html
MIT tvár	pozitívne obrázky, čelné pohľady	http://cbcl.mit.edu/software-datasets/FaceData2.html
Cambridge ruky	pozitívne obrázky rúk, rôzne osvetlenie	http://www.iis.ee.ic.ac.uk/icvl/ges_db.htm
Cambridge tvár	pozitívne obrázky tváre	http://www.cl.cam.ac.uk/research/dtg/attarchive/pub/data/

Tab. 3.1: Kolekcie obrázkov.

3.4 Trénovanie a testovanie detektora

Nakoľko sa v práci používa wrapper OpenCV do programovacieho jazyka Java, využíval som OpenCV aj pri trénovaní klasifikátorov. Žiaľ, wrapper JavaCV neobsahuje všetko to, čo OpenCV, nebolo teda možné priamo v Jave využit trénovacie aplikácie, ktoré sú v OpenCV dostupné. Na tieto účely boly teda využité priamo aplikáce z OpenCV. Jedná sa o aplikácie haartraining a traincascade. Aplikácia haartraining slúžila na vytvorenie trénovacích a testovacích sád obrázkov, aplikácia traincascade na samotné natrénovanie klasifikátorov. Pri trénovaní klasifikátorov mi bol najväčšiou oporou tutorial, ktorého tvorcom je pán Naotoshi Seo. Tento turorial je dostupný na adrese http://note.sonots.com/SciSoftware/haartraining.html.

V tomto tutoriale sa podrobne popisuje, ako sa majú vytvárať kolekcie pozitívnych obrázkov z jedného obrázku alebo z kolekcie obrázkov, ako má byť zapísaná kolekcia negatívnych obrázkov. Rovnako je to rozoberaná voľba veľkosti okna, či počet pozitívnych a negatívnych obrázkov. Po preštudovaní tohoto tutorialu som sa pustil do samotnej prípravy trénovacích množín z pripravených kolekcií obrázkov. Pre tvár a torzo bola zvolená veľkosť detekčného okna na 20 x 20 pixelov, pre nohy to bolo 20 x 40 pixelov a pre ruku bola veľkosť okna zvolená na 26 x 20 pixelov. Ukážky jednotlivých pozitívnych obrázkov sú uvedené na obrázku 3.1.

U všetkých klasifikátorov boli zhodne nastavené následujúce parametre:

- maximálna hĺbka stromu: 2
- minimálna úspešnosť správne označených detekcií: 0,999
- počet silných klasifikátorov v kaskáde: 20
- minimálna úspešnosť správne označených detekcií: 0,999
- typ boost algoritmu: Reálny AdaBoost

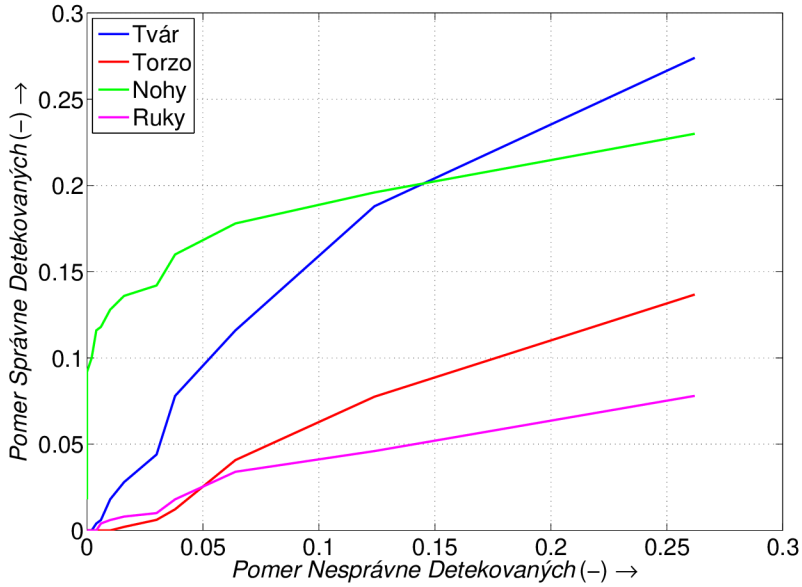
Parametre, ktoré nie sú uvedené, mali nastavené východzie hodnoty.

Obr. 3.1: Ukážka pozitívnych obrázkov pre tvár, torzo, nohy, ruky.

Trénovanie klasifikátorov bolo časovo náročné. Najmenej trvalo natrénovanie klasifikátora pre torzo, boli to zhruba 4 hodiny. Natrénovanie klasifikátora tváre trvalo zhruba 22 hodín, natrénovanie klasifikátora nôh zhruba 22 hodín a najviac, 64 hodín, zabralo natrénovanie klasifikátora rúk. Všetky klasifikátory boli trénované na notebooku s procesorom Intel i5-3230M s frekvenciou $2,6GHz$ a $8GB$ operačnej pamäte.

Niektoré klasifikátory aj napriek dostatočne veľkej trénovacej množine nedosahovali dostatočné výsledky. ROC krivky zobrazené na nasledujúcom grafe zodpovedajú

detekčným výsledkom na testovacích množinách pre jednotlivé klasifikátory. Tieto testovacie množiny sú súčasťou DVD nosiča priloženého k práci.

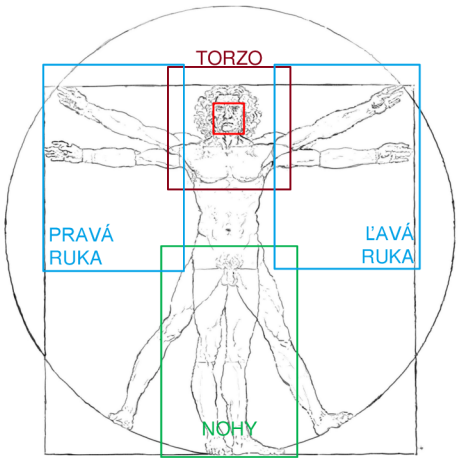


Obr. 3.2: ROC krivky jednotlivých klasifikátorov.

3.5 Detekcia postáv

Detekcia postáv je realizovaná postupným aplikovaním dielčích detektorov. Základným detektorom, ktorý sa aplikuje na celý obrázok, je detektor tváre. Dôvodom tohoto rozhodnutia je, že ak bude osoba chcieť zadať nejaké gesto, predpokladá sa, že bude tvárou otočená na kameru, ako pri medziľudskej komunikácii. Následne sa pre všetky zdetekované tváre aplikuje detektor torza len pre oblasť záujmu, ktorá sa odvíja od umiestnenia a veľkosti zdetekovanej tváre. Ak je v tejto oblasti zájmu zdetekované torzo, je detekcia postavy označená za úspešnú. Nohy nemajú vplyv na vyhodnocovanie zdetekovanej ľudskej postavy, pretože nohy nemusia byť vôbec súčasťou skúmaného obrázku. V prípade detekcie ľudskej postavy sa znovu určí oblasť záujmu, na ktorú sa aplikuje detektor nôh, aby došlo k správnemu označeniu postavy v prípade, že sa na obrázku nachádza celá postava. Ako posledný sa aplikuje

detektor rúk, ktorý sa aplikuje dvakrát, raz pre pravú a raz pre ľavú ruku. Oblasti záujmu pre jednotlivé časti sú znázornené na následujúcom obrázku.



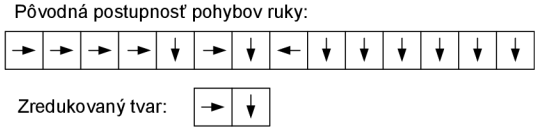
Obr. 3.3: Oblasti záujmu v závislosti k zdetekovanej tvári.

Dôvod, prečo je detekcia realizovaná takýmto spôsobom, je vcelku jednoduchý. Snaha docieliť čo najlepšie časové výsledky. V prípade, že po aplikácii detektora tváre na celý obrázok nie je nájdená žiadna zdetekovaná tvár, nedochádza k detekcii ďalších dielčích detektorov. V prípade detekcie sa tieto dielčie detektory aplikujú len na presne definovanú časť obrázka. Táto metóda vyžaduje aplikáciu robusných detektorov, ktoré dosahujú vysokú úspešnosť detekcie.

3.6 Detekcia gest

Po pozitívnej detekcii postavy sa poloha rúk pridá do zoznamu, ktorý v sebe nesie informáciu o pohybe pravej a ľavej ruky pre zdetekovanú osobu. Tento zoznam obsahuje polohy pravej a ľavej ruky a polohu tváre. Poloha tváre je tu pre identifikáciu, či ruky patria do tohoto zoznamu pre prípad detekcie viacerých osôb. Nakoľko aplikácia bola navrhnutá pre detekciu práve jednej osoby vo videosekvencii, nebolo by nutné prípad detekcie viacerých osôb riešiť, no to by mohlo viesť k destabilizácii aplikácie. Po každom pridaní polohy rúk do zoznamu sa spustí rozpoznávač gest pre postupnosť zatiaľ zaznamenaného pohybu jednotlivých rúk, ktorý zaznamenané postupnosti pohybov prevedie na redukovaný tvar, ktorý sa porovnáva s pevne danou množinou gest. Ku každému gestu z tejto množiny je priradený práve jeden filter,

ktorý sa na výstupný obraz aplikuje v prípade rozpoznania gesta, ako priamy dôkaz, že gesto bolo rozpoznané. Detekcia gest podporuje len pohyby hore, dole, doprava, doľava, stojí (bez pohybu) a „nie je určený“ v prípade, že detekcia neobsahuje polohy rúk. Pohyb zdetekovanej ruky určitým smerom je do zoznamu pohybu zaznamenaný na základe rozdielu medzi aktuálnou a predchádzajúcou polohou. Do zoznamu sa nevkladajú súradnice, ale priamo hodnoty zodpovedajúce jednotlivým pohybom. Po aplikácii detekčného algoritmu, ešte pred vložením snímku a výsledkov detekcie do medzipamäte, dochádza k vyhodnocovaniu všetkých zaznamenaných postupností. Vyhodnocovaniu predchádza prevod postupností do redukovaného tvaru, ktorý sa priamo porovnáva s vopred danou množinou známych gest. Príklad redukovaného tvaru je uvedený na obrázku 3.4. Táto postupnosť je reprezentovaná reťazcom znakov, pričom každý pohyb je reprezentovaný jedným písmenom (začiatočné písmená z anglického výrazu pre daný pohyb). Do redukovaného tvaru sa dostanú len tie pohyby, ktoré sa v zozname vyskytujú minimálne trikrát za sebou. Definícia gest je určená pomocou postupnosti smerov. Zoznam pohybov rúk má obmedzenú veľkosť na počet snímkov odpovedajúcich dobe pätnástich sekúnd. Jedná sa o zásobník typu FIFO, takže po naplnení sú odstraňované najstaršie pohyby rúk. Po pozitívnej detekcii gesta sa záznam pohybu okamžite vymaže.



Obr. 3.4: Zaznamenaná postupnosť pohybu ruky výsledný redukovaný tvar.

V aplikácii sú implementované len štyri gestá uvedené v nasledujúcej tabuľke.

Definícia gesta	**Zmena výstupného snímku**
hore, vľavo	šedotónový snímok
hore, vpravo	rozostrený šedotónový snímok
dole, vľavo	farebný snímok
dole, vpravo	rozostrený farebný snímok

Tab. 3.2: Tabuľka implementovaných gest.

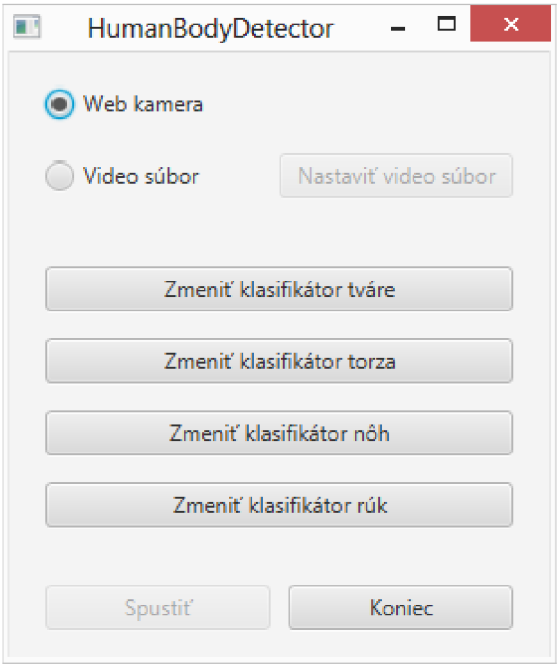
4 NÁVRH APLIKÁCIE

Pri návrhu aplikáce som už zo skúseností z predchádzajúceho semestrálneho projektu vedel, že sa bude musieť jednať o viac vláknovú aplikáciu, nakoľko by nebolo možné zrealizovať v jednom vlákne vyčítavanie snímkov z videosekvencie a zároveň realizovať detekciu tak, aby sa stále jednalo o aplikáciu pracujúcu v reálnom čase. Nakoľko aplikácia v semestrálnom projekte využívala knižnicu Xuggle, ktorá bola v diplomovej práci vymenená za knižnicu JavaCV, rozhodol som sa neupravovať predchádzajúcu aplikáciu. Z vlastných skúseností viem, že úprava existujúcej aplikácie môže zabrať viac času, ako napísať aplikáciu odznova podľa aktuálnych predstáv. Po spustení aplikácie sa z hlavného vlákna spustí jedno vlákno pre vyčítavanie snímkov z videosekvencie a zobrazovanie upravených snímkov z medzipamäte. Po vyčítaní snímkov dochádza k spúšťaniu ďalších vlákien, v ktorých sa realizuje samotná detekcia, následné vyhodnocovanie zaznamenaného pohybu rúk a na záver vloženie snímku a výsledkov detekcie do medzipamäte. Nikdy nedochádza k spracovávaniu každého snímku. Pri videosekvenciách s nižšou snímkovou frekvenciou (do 15 snímkov za sekundu) dochádza k spracovávaniu každého štvrtého snímku. Pri videosekvenciách s vyššiou snímkovou frekvenciou dochádza k spracovávaniu každého šiesteho snímku. Prvý snímok je spracovávaný vždy, pretože snímky, ktoré nie sú spracovávané, sú rovno uložené do medzipamäte s výsledkami detekcie z posledného spracovávaného snímku, aby v prípade pozitívnej detekcie videl užívateľ označenú zdetekovanú postavu stále a nie len na každom štvrtom, či šiestom snímku.

4.1 Grafické užívateľské rozhranie (GUI)

Grafické rozhranie aplikácie je pomerne minimalistické, ako je možné vidieť na obrázku 4.1.

Behom vývoja aplikácie som prešiel cez niekoľko rôznych návrhov. Okno malo pôvodne obsahovať aj sekciu, kde by sa zaznamenávali zdetekované gestá, no nakoniec som sa rozhodol, že indikácia detekcie sa bude realizovať priamo v zobrazovanom obraze. Hlavné okno aplikácie teda obsahuje len výber, či má byť spracovávaná videosekvencia z web kamery, alebo video súbor z pevného disku. Pri výbere možnosti **Web kamera** sa začne realizovať spracovávanie z defaultnej web kamery pripojenej



Obr. 4.1: Grafické uživateľské rozhranie.

k počítaču. Pri výbere možnosti **Video súbor** je nutné vybrať video súbor pomocou tlačítka **Nastaviť video súbor**, v opačnom prípade je tlačítko **Spustiť** zašednuté. Po nastavení vstupnej videosekvencie je možné zmeniť súbory obsahujúce kaskáru dielčích klasifikátorov. Možnosť nastaviť súbory s rôznymi klasifikátormi som sa rozhodol pridať z dôvodu lepšej možnosti testovania rôznych klasifikátorov v prípade, žeby niektorý klasifikátor nebol dostatočne robusný a jeho výsledky by neboli dostačujúce. Stlačením tlačítka **Spustiť** sa spustí spracovanie videosekvencie podľa predchádzajúceho výberu. Tlačítko **Koniec** slúži na ukončenie spustených vlákien a následné ukončenie aplikácie.

4.2 Trieda `Main`

Jedná sa o hlavnú triedu celej aplikácie, ktorá je prepojená s grafickým rozhraním. Úlohou tejto triedy je inicializovať grafické prostredie a následne obsluhovať príkazy z grafického prostredia pomocou nasledujúcich funkcií

- `isRunning()` - funkcia, ktorá slúži na zistenie hodnoty uloženej v globálnej premennej `running` typy `boolean`. Pokiaľ je táto premenná nastavená na hodnotu `true`, vlákno pre vyčítavanie snímkov z videosekvencie je zacyklené v

nekonečnej slučke, ináč sa vlákno ukončí. Slúži hlavne na korektné ukončenie vlákna pri vypnutí aplikácie.

- `getBufferSize()` - funkcia, ktorá vracia celočíselnú hodnotu obsahujúcu počet upravených snímkov vložených do medzipamäte. Táto funkcia sa využíva v triede `Process` k dosiahnutiu oneskorenia medzi aktuálne spracovávaným snímkom a aktuálne zobrazovaným snímkom. Toto oneskorenie je nutné na spracovanie snímku a dosiahnutie plynulého chodu videosekvencie.
- `flushBuffer()` - funkcia využívaná pri ukončení aplikácie. Nakoľko v momente ukončenia môžu byť v medzipamäti nejaké upravené snímky, ktoré už nebudú zobrazené, dôjde k ich odstráneniu a vyprázdneniu medzipamäte.
- `switchInputOption()` - táto funkcia je volaná pri zmene možnosti pre spracovávanie videosekvencie z web kamery, alebo z video súboru. Výber sa zaznamená do globálnej premennej typu `boolean`, na základe ktorej sa vo funkcii `init()` realizuje odlišné vytváranie objektu s videosekvenciou.
- `setInputFile()` - pri výbere možnosti spracovania videosekvencie zo súboru je nutné vybrať konkrétny súbor z disku. Na to slúži táto funkcia, ktorá je volaná stlačením tlačítka **Načítať video súbor**. Táto funkcia otvorí prehliadač súborov s prednastaveným filtrom na video súbory. Po výbere súboru je do globálnej premennej uložená absolútna cesta k súboru.
- `setFaceCascade()` - nastavenie súboru obsahujúceho klasifikátor tváre.
- `setTorsoCascade()` - nastavenie súboru obsahujúceho klasifikátor torza.
- `setLegsCascade()` - nastavenie súboru obsahujúceho klasifikátor nôh.
- `setHandCascade()` - nastavenie súboru obsahujúceho klasifikátor rúk.
- `init()` - funkcia sa zavolá po stlačení tlačítka **Spustiť**. Na základe výberu v grafickom prostredí dôjde k vytvoreniu premennej typu `CvCapture` z video súboru, alebo z web kamery. Táto premenná sa odovzdá do konštruktora triedy `Process`. Po vytvorení inštancie tejto triedy je funkciou `start()` spustená funkcia `run()` definovaná v tejto triede v samostatnom vlákne.
- `close()` - táto funkcia sa volá pri stlačení tlačítka **Koniec**. V tejto funkcii dôjde k nastaveniu globálnej premennej `boolean running` na hodnotu `false`. Po nastavení tejto premennej sa čaká na ukončenie vlákna s inštanciou triedy `Process`. Po ukončení tohoto vlákna dôjde k vyprázdneniu medzipamäte zavolaním funkcie `flushBuffer()` a samotnému ukončeniu aplikácie.

Trieda `Main` ďalej obsahuje dôležité globálne premenné, s ktorými sa pracuje v jednotlivých vláknach aplikácie. Medzi tieto premenné patria názvy súborov obsahujúce kaskády jednotlivých dielčích klasifikátorov, rozlíšenie spracovávanej videosekvencie, pomer šírky vybranej videosekvencie k vopred určenej šírke, názov súboru pre spracovanie videosekvencie z pevného disku, hodnotu minimálneho oneskorenia, ktoré je požadované pri zobrazovaní upravenej videosekvencie, či medzipamäť upravených snímkov, synchronizačný zámok na prístup k tejto medzipamäti, zoznam zaznamenaných postupností pohybu rúk, synchronizačný zámok na prístup k tomuto zoznamu, informáciu o tom, aké gesto bolo naposledy zdetekované.

4.3 Trieda `Process`

Táto trieda slúži na priamy prístup k videosekvencii. Jedná sa o rozšírenie triedy `Thread`. Trieda sa inicializuje z triedy Main zavolaním konštruktora, ktorého hlavička je nasledovná

```
public Process(CvCapture capture).
```

Parameter predávaný do tohoto konštruktora obsahuje súbor z pevného disku, alebo videosekvenciu prijímanú z web kamery. Pri vytváraní inštancie tejto triedy sa vytvorí aj zoznam časových známok, ktorý sa využíva pri zaručení správneho poradia pri zobrazovaní upravených snímkov. Po vytvorení inštancie tejto triedy je funkciou `start()` spustené samostatné vlákno, ktoré vykonáva operácie obsiahnuté vo fukncii `run()`. V tejto funkcii dochádza postupne k zisteniu rozlíšenia videosekvencie a prípadnému nastaveniu škálovacieho pomeru, ktorý sa bude využívať pri zmenšovaní snímku pre účely detekcie. Tento pomer sa nastaví v prípade, že šírka videosekvencie je väčšia ako $768px$ ($1.2 * 640px$). V takom prípade dôjde pred samotnou detekciou k zmenšeniu videosekvencie na šírku obrazu $640px$ pri zachovaní pomeru strán. Toto zmenšenie sa realizuje z dôvodu eliminácie veľkosti rozlíšenia videosekvencie na čas potrebný na detekciu. Pomer šírky pôvodnej a zmenšenej videosekvencie je uložený do globálnej premennej `rescaleFactor` z triedy `Main`. Táto premenná sa využíva taktiež pri nastavení správnych rozmerov na označenie zdetekovaných postáv vo výslednej videosekvencii.

Na základe snímkovej frekvencie videosekvencie sa nastaví, koľko snímkov sa nemá spracovávať medzi dvoma spracovávanými snímkami. V prípade, že je spracovávaný video súbor, snímkovú frekvenciu je možné priamo vyčítať, v prípade spracovávania videosekvencie z web kamery dôjde k výpočtu priemernej snímkovej frekvencie z desiatich snímkov.

Následne sa nastaví veľkosť medzipamäte pre postupnosti pohybu rúk reprezentujúca dobu $15s$ a veľkosť medzipamäte upravených snímkov reprezentujúca čas $300ms$. Tento čas je oneskorenie, ku ktorému by malo dochádzať pri zobrazovaní upravenej videosekvencie.

Po týchto inicializačných operáciách dochádza k samotnému spracovaniu videosekvencie v nekonečnej slučke, ktorá môže byť prerušená ukončením programu, chybou čítania z videosekvencie, alebo vyčítaním celej videosekvencie v prípade video súboru z pevného disku. V tejto slučke dochádza k vyčítaniu snímku z videosekvencie a kontrole, či tento snímok obsahuje nejaké dáta. Pokiaľ je snímok v poriadku, dochádza k vytvoreniu inštancie typu `Detect`, ktorej sa odovzdá časová známka, pre neskoršie správne vloženie a vyčítanie z medzipamäte, vyčítaný snímok, vyššie spomínaná premenná `rescaleFactor` a príznak, či má, alebo nemá byť realizovaná detekcia ľudskej postavy v v tomto snímku.

V závere tejto funkcie dochádza k vyčítavaniu snímkov z medzipamäte, pokiaľ už obsahuje dostatočné množstvo. Pri vyčítaní sa kontroluje, či snímok má správnu časovú známku. Na záver sú v obraze označené zdetekované osoby s vyznačením polohy rúk. Po týchto operáciách dôjde k zobrazeniu upraveného snímku v okne **Result**.

4.4 Trieda `Detect`

Trieda `Detect` slúži na aplikáciu detekčného algoritmu na snímok, pokiaľ na ňom má byť realizovaná detekcia. Po vykonaní detekcie dochádza k vyhodnocovaniu zaznamenaného pohybu rúk, v prípadne pozitívnej detekcie aj k vloženiu polohy rúk do postupnosti pohybov. V závere dochádza k vloženiu snímku spolu s údajmi o výsledku detekcie do medzipamäte upravených snímkov. Jedná sa o triedu rozširujúcu triedu `Thread`, takže po vytvorení inštancie tejto triedy a zavolaním metódy `start()` dôjde k vykonávaniu operácií v tejto triede v samostatnom vlákne. Funkcia

sa inicializuje z triedy `Process`. Hlavička konštruktora tejto triedy je nasledovná

```
public Detect(long stamp, IplImage originalImage,
        double scaleFactor, boolean skip).
```

Prvým parametrom je časová známka, ktorá je pre každý snímok získaná pomocou funkcie `System.currentTimeMillis();`, druhým parametrom je snímok vyčítaný z videosekvencie, tretím parametrom je pomer šírky originálneho snímku a čísla 640 a posledným parametrom je príznak nesúci informáciu o tom, či má dôjsť k aplikácii detekcie, alebo len k vloženiu snímku do medzipamäte. V konštruktore dôjde k odovzdaniu časovej známky a originálneho snímku do privátnych premenných triedy `Detect`. Rovnako dochádza k vytvoreniu šedotónového snímku, na ktorý sa následne aplikujú jednotlivé dielčie klasifikátory. Pokiaľ je premenná `scaleFactor` väčšia ako $1,2$, vytvára sa šedotónový snímok zo zmenšeného snímku, ktorý má šírku $640px$ a zachovaný pomer strán originálneho snímku. V opačnom prípade sa vytvára šedotónový snímok priamo z originálneho snímku.

V metóde `run()` dochádza k vytvoreniu premennej typu `DetectAll` v príapde že má dôjsť k detekcii (premenná `skip` predaná v konštruktore triedy má hodnotu `false`). Následne je zavolaná funkcia `doWork()` tejto premennej, ktorá vráti zoznam zdetekovaných osôb spolu s polohou stredných bodov tváre a rúk v snímku. Po tejto funkcii dochádza k uvoľňovaniu pamäte šedotónového snímku a zmenšeného snímku, ak bol vytvorený, pretože ďalej nie sú potrebné. Po zrealizovaní detekcie dochádza k vyhodnocovaniu zaznamenaného pohybu rúk. Vo východzom stave je nastavené zobrazovanie pôvodného snímku. V aplikácii sú implementované len štyri gestá, pričom je každému gestu priradené iné zobrazenie výstupného snímku. V prípade rozpoznania niektorého z gest dôjde k zmene výstupného snímku podľa tabuľky 3.2. Takto zmenené ostávajú všetky následujúce snímky až pokiaľ nedôjde k rozpoznaniu iného gesta.

V prípade, že čas detekcie a rozpoznávania gest nepresiahne $300ms$ (táto hodnota je vyčítaná z hodnoty nastavenej v triede `Main`), je vlákno uspané na čas potrebný na dosiahnutie tejto doby. Uspanie vlákna sa deje z dôvodu dosiahnutia približne rovnakého času spracovania pre snímky, na ktoré je aplikovaná detekcia a snímky, ktoré sú len vkladané do medzipamäte. Rovnako sa uspanie vlákna realizuje z dôvodu vloženia snímku na správne miesto v medzipamäti. Na záver dôjde k vloženiu časovej známky, pôvodného snímku a výsledkov detekcie do medzipamäte. Pred samotným

vkladaním sa ešte realizuje kontrola, či je snímok vkladaný na správne miesto v medzipamäti. Táto kontrola je založená na hodnote časovej známky.

4.5 Trieda `DetectorAll`

Táto trieda je v podstate grom celej aplikácie. Práve v tejto triede dochádza k aplikácii dielčích klasifikátorov a je tu implementovaná metóda, ktorá je popísaná v časti 3.5. Inštancia tejto funkcie sa vytvára v triede `Detect` nasledujúcim príkazom

```
DetectorAll detector = new DetectorAll(grayScaleFrame,
rescaleFactor);.
```

Táto trieda obsahuje sedem privátnych premenných. Prvými dvoma sú premenné, ktoré sa predávajú v konštruktore triedy. Je to šedotónový snímok a pomer pôvodnej a zmenšenej šírky snímky. Ak nedošlo k zmenšeniu snímku, tento pomer je rovný jednej. Ďalšie štyri podstatné premenné sú typu `CvHaarClassifierCascade`. Každá jedna premenná v sebe obsahuje jeden z dielčích klasifikátorov pre tvár, torzo, nohy a ruky. Poslednou privátnou premennou je `rectList`. Jedná sa o zoznam výsledkov detekcie, ktorý obsahuje objekty typu `HumanBodyRect`. Tento typ je presnejšie popísaný v kapitole 4.6. Čo sa týka funkcií, táto trieda obsahuje len jednu funkciu, `doWork()`. V tejto funkcii dochádza k aplikácii tvárového kalsifikátora na celý šedotónový snímok. Následne dochádza pre všetky zdetekované tváre k nastaveniu regiónu záujmu. Rozmery regióna záujmu sú následovné

- šírka je 6-násobkom šírky zdetekovanej tváre,
- výška je 6-násobkom výšky zdetekovanej tváre,
- región záujmu je vertikálne vycentrovaný s vertikálnou osou zdetekovanej tváre,
- vrchný okraj regiónu záujmu sa nachádza o $1,5$ násobok výšky zdetekovanej tváre nad vrchným okrajom zdetekovanej tváre.

Na takto nastavený región záujmu sa aplikuje klasifikátor torza. Aj napriek predpokladu, že bude zdetekované žiadne, alebo práve jedno torzo, sa tu realizuje operácia pre všetky zdetekované torzá pomocou `for` cyklu. V tomto cykle sa najprv vyhodnocuje poloha a rozmery zdetekovaného torza. Podmienky sú nasledovné

- šírka zdetekovaného torza je 4-násobkom šírky tváre (tolerancia je šírka zdetekovanej tváre),
- výška torza je 4-násobkom výšky tváre (tolerancia je výška zdetekovanej tváre),

- zdetekovaná tvár sa nachádza vo vrchnej časti zdetekovaného torza (minimálne pol výšky tváre pod vrchným okrajom torza)
- zdetekované torzo je vertikálne vycentrované so zdetekovanou tvárou (tolerancia 20% šírky tváre).

Pri splnení všetkých podmienok dochádza k označeniu torza a tváre ako zdetekovanej postavy. Ďalšie zdetekované torzá už nie sú skúmané.

Po vyhodnotení tváre a torza ako ľudskej postavy dochádza k zaznamenaniu stredného bodu zdetekovanej tváre a taktiež k zaznamenaniu stvoruholníka obopínajúceho zdetekvoanú takto zdetekovanú postavu. Následne dôjde k nastaveniu regiónu záujmu podľa následujúcich podmienok

- šírka je 5-násobkom šírky tváre,
- výška je 6-násobkom výšky tváre,
- región je vertiklne vycentrovaný k vertikálnej osi tváre,
- vrchný okraj je 4-násobok výšky tváre nižšie ako vrchný okraj tváre.

Po nastavení tohoto regiónu záujmu je na neho aplikovaný klasifikátor nôh. Opäť, ako v predchádzajúcom prípade, som predpokladal, že v regióne záujmu budú zdetekované žiadne, alebo práve jedny nohy, no z dôvodu nesprávne označenej detekcie som zvolil `for` cyklus na prechod všetkými zdetekovanými nohami. Pokiaľ veľkosť a poloha zdetekovaných nôh spĺňa následujúce podmienky

- šírka je 3-násobkom šírky tváre (toleranica pol šírky tváre),
- výška je 5-násobkom výšky tváre (tolerancia pol šírky tváre),
- zdetekované nohy sú vertikálne vycentrované k vertikálnej osi tváre (tolerancia 25% šírky tváre),
- vrchný okraj je 5-násobok výšky tváre nižšie ako vrchný okraj tváre (tolerancia pol výšky tváre),

dochádza k priradeniu zdetekovaných nôh k tvári a torzu. V podstate to znamená, že sú upravené rozmery štvoruholníka, ktorý obopína zdetekovanú osobu.

Ako posledné sa nastavujú regióny záujmu pre hľadanie pravej a ľavej ruky, pre ktoré platí následovné

- šírka je 4-násobkom šírky tváre (platí pre obe ruky),
- výška je 6-násobkom výšky tváre (platí pre obe ruky),
- vrchný okraj je o výšku tváre vyššie ako vrchný okraj tváre (platí pre obe ruky),

- ľavý okraj je posunutý o šírku tváre doprava od pravého okraja tváre (platí pre ľavú ruku),
- pravý okraj je posunutý o šírku tváre doľava od ľavého okraja tváre (platí pre pravú ruku).

Na tieto regióny je postupne aplikovaný klasifikátor rúk. Pre všetky zdetekované ruky sa pre každý región skúmajú nasledujúce podmienky

- šírka je približne rovnaká ako šírka tváre (platí pre obe ruky s toleranciou 30% šírky tváre),
- výška je približne rovnaká ako výška tváre (platí pr eobe ruky s toleranciou 30% výšky tváre).

Prvá zdetekovaná ruka pre každú oblasť, ktorá spĺňa vyššie uvedené podmienky, je vyhodnotená ako zdetekovaná ruka patriaca k rovnakej ľudskej postave, ako zdetekovaná tvár. Ďalšie detekcie rúk v regióne záujmu sa nevyhodnocujú. Po pozitívnej detekcii rúk dôjde k zaznamenaniu stredného bodu zdetekovanej ruky a úprave rozmerov štvoruholníka tak, aby zdetekované ruky boli vo vnútri tohoto štvoruholníka. Súradnice stredných bodov oboch zdetekovaných rúk spolu so súradnicami stredného bodu tváre a rozmermi štvoruholníka (súradnice ľavého horného rohu, šírka a výška), ktorý opisuje zdetekovanú postavu, sú pridané do zoznamu, ktorý je návratovou hodnotou. Tento zoznam môže obsahovať výsledky viacerých detekcií. Nastavovanie regiónov záujmu pre torzo, nohy a ruky sa realizuje pre všetky zdetekované tváre v snímku. Po nastavení každého regiónu záujmu sa kontrolujú rozmery a v prípade, že presahujú hranice snímku, sú upravené tak, aby tieto hranice nepresahovali.

4.6 Trieda `HumanBodyRect`

Trieda reprezentujúca jednu zdetekovanú ľudskú postavu. Táto trieda obsahuje sedem privátnych premenných, šesť z nich je typu `int`, jedna je typu `CvRect`. Premenná typu `CvRect` nesie informáciu o štvoruholníku, ktorý označuje celú zdetekovanú ľudskú postavu. U zvyšných šiestich premenných ide o tri dvojice premenných. Jedna dvojica reprezentuje súradnice stredného bodu pre zdetekovanú tvár, zvyšné dve dvojice sú súradnice stredného bodu ľavej a pravej ruky. Inštanciu tejto triedy je možné vytvoriť pomocou konštruktora bez parametrov

```
new HumanBodyRect();,
```

alebo konštruktora s parametrami

```
new HumanBodyRect(humanBodyStartX, humanBodyStartY, HumanBodyWidth,
HumanBodyHeigh, faceX, faceY, leftArmX, leftArmY, rightArmX,
rightArmY);.
```

V prvom prípade sa jedná o konštruktor bez parametrov, ktorý sa využíva v prípade snímkov, u ktorých nie je realizovaná detekcia. Po vytvorení inštancie tejto triedy týmto spôsobom dôjde k inicializácii premennej typu `CvRect`, no nedochádza k nastaveniu rozmerov štvoruholníka a všetky tri dvojice reprezentujúce súradnice stredných bodov tváre a rúk sú nastavené na hodnotu -1. Pri vytvorení inštancie tejto triedy druhým spôsobom dôjde k vytvoreniu štvoruholníka, pričom prvé dva parametre v konštruktore reprezentujú súradnice ľavého horného bodu štvoruholníka, tretí parameter je šírka štvoruholníka, štvrtý parameter je výška štvoruholníka a zvyšných šesť parametrov sú dvojice stredných bodov postupne pre tvár, ľavú ruku a nakoniec pre pravú ruku. Zoznam inštancií tejto triedy je návratovou hodnotou funkcie `doWork()` triedy `DetectorAll`. Tento zoznam sa spolu so snímkom z pôvodnej videosekvencie ukladá do medzipamäte. Snímok môže byť upravený na základe rozpoznaného gesta, no k označeniu zdetekovaných osôb dochádza až pred samotným zobrazením snímku po vyčítaní z medzipamäte.

Na prístup k jednotlivým hodnotám obsiahnutým v tejto triede slúžia nasledujúce funkcie

- `getRect()` - funkcia vracia štvoruholník opisujúci zdetekovanú ľudskú postavu,
- `getLeftArm()` - funkcia vracia stredný bod ľavej zdetekovanej ruky,
- `getRightArm()` - funkcia vracia stredný bod pravej zdetekovanej ruky,
- `getFace()` - funkcia vracia stredný bod zdetekovanej tváre.

4.7 Trieda `ModifiedIplImage`

Táto trieda reprezentuje jednu bunku v medzipamäti upravených snímkov z pôvodnej videosekvencie. Obsahuje štyri privátne premenné. Prvá je typu `long` a jedná sa o časovú známku, ktorá slúži na zabezpečenie správneho poradia zobrazovaných snímkov. Druhá premenná je typu `IplImage` a jedná sa o snímok vyčítaný z videosekvencie, ktorý môže byť v prípade rozpoznania gesta upravený podľa tabuľky 3.2. Tretia premenná je typu `List<HumanBodyRect>` a obsahuje výsledky detekcie,

konkrétne štvoruholník obopínajúci zdetekovanú ľudskú postavu a taktiež stredné body zdetekovanej tváre a stredné body zdetekovaných rúk. Posledná premenná je príznak s informáciou, či na snímok bol aplikovaný detekčný algoritmus, alebo nie.

Túto triedu je možné vytvoriť len jedným spôsobom. Použitím konštruktora s parametrami, ktorého hlavička je nasledujúca

```
public ModifiedIplImage(long stamp, IplImage image,
List<HumanBodyRect> rectangularList, boolean fromDetect);.
```

Poradie parametrov konštruktora je totožné s vyššie popísanými premennými.

Táto trieda ďalej obsahuje nasledujúce funkcie, ktoré slúžia na prístup k jednotlivým dátam

- `getStamp()` - funkcia vracia časovú známku snímku. Využíva sa pri vyčítaní snímku z medzipamäte pred zobrazením. Návratová hodnota sa porovnáva so zoznamom časových známok, ktoré sa ukladajú v triede `Process` pri vyčítaní snímku z videosekvencie. Takto je zabezpečené vyčítanie snímkov z medzipamäte v správnom poradí.
- `getData()` - funkcia vracia samotný snímok,
- `getRectangulars()` - funkcia vracia zoznam informácií (rozmery štvoruholníka, súradnice stredných bodov tváre a rúk) o zdetekovaných osobách v snímku,
- `hasRect()` - funkcia vracia veľkosť zoznamu označených zdetekovaných osôb, a teda aj počet osôb zdetekovaných v snímku.
- `isFromDetect()` - funkcia vracia príznak, ktorý v podstate určuje, či na snímok bol aplikovaný detekčný algoritmus. Využíva sa po vyčítaní snímku z medzipamäte na určenie, či má byť snímok označený výsledkami z detekcie tohoto snímku, alebo s výsledkami detekcie posledného snímku, na ktorom bola detekcia realizovaná.

4.8 Enum `ArmMove`

Enum `ArmMove` slúži na reprezentáciu pohybu rúk a obsahuje v sebe hodnoty pre pohyb hore, dole, vpravo, vľavo a taktiež hodnotu reprezentujúcu žiadny pohyb a hodnotu reprezentujúcu nezdetekovanú ruku. Tieto hodnoty sa vkladajú do zoznamu pohybu rúk po vyhodnotení aktuálnej polohy oproti predchádzajúcej polohe ruky.

4.9 Trieda `ArmPosition`

Trieda reprezentujúca jednu zdetekovanú osobu vo videosekvencii s vyznačením súradníc dôležitých bodov pre sledovanie pohybu. Táto trieda obsahuje osem privátnych premenných. Sú to tri dvojice reprezentujúce súradnice stredného bodu tváre a súradnice stredných bodov pravej a ľavej ruky. Posledné dve privátne premenné sú zoznamy pohybov pravej a ľavej ruky. Súradnice stredného bodu tváre slúžia na priradenie pohybu rúk k správnej osobe, nakoľko som vychádzal z predpokladu, že pri zadávaní gesta zdetekovaná postava nebude príliš hýbať hlavou. Súradnice stredných bodov rúk slúžia na uloženie poslednej vkladanej polohy pravej a ľavej ruky, pretože do zoznamu pohybov a pre ľavú a pravú ruku sa nevkladajú súradnice, ale pohyb, ktorý ruka spravila oproti poslednej zaznamenanej polohe. Vytvorenie inštancie tejto triedy je možné len jedným konštruktorom, ktorý obsahuje tri parametre, konkrétne sú to súradnice stredného bodu tváre a súradnice stredného bodu ľavej a pravej ruky. Hlavička konštruktoru vyzerá následovne

`public ArmPosition(int[] faceDim, int[] leftArmDim, int[] rightArmDim)`.

Táto trieda obsahuje nasledujúce funkcie

- `addLeftArmPos(ArmMove move)` - funkcia, ktorá na koniec zoznamu pohybu ľavej ruky vloží pohyb predaný ako parameter funkcie. Využíva sa vo funkcii `addArms` po vyhodnotení, aký pohyb bol spravený od poslednej polohy ľavej ruky.
- `addRightArmPos(ArmMove move)` - funkcia, ktorá na koniec zoznamu pohybu pravej ruky vloží pohyb predaný ako parameter funkcie. Využíva sa vo funkcii `addArms` po vyhodnotení, aký pohyb bol spravený od poslednej polohy pravej ruky.
- `getFacePos()` - funkcia, ktorá vracia súradnice stredného bodu tváre. Táto funkcia sa používa v prípade viacerých zdetekovaných osôb na identifikáciu, ku ktorej osobe má byť pohyb zdetekovaných rúk vložený.
- `addArms(int[] dimFace, int[] leftArmDim, int[] rightArmDim)` - funkcia, ktorej vstupné parametre sú súradnice stredného bodu tváre a súradnice stredného bodu ľavej a pravej ruky. Funkcia je zavolaná potom, ako je správne

identifikovaná osoba, ktorej sa má rozšíriť postupnosť pohybov. Po vyhodnotení pohybu z poslednej známej polohy pre ľavú a pravú roku dôjde k vloženiu pohybu do postupností. Zároveň dochádza k prepisu poslednej polohy stredného bodu tváre a prepisu poslednej polohy stredného bodu ľavej a pravej ruky.

- `getMove(int difX, int difY)` - funkcia, ktorá sa používa vo vyššie uvedenej funkcii. Úlohou tejto funkcie je z dvoch vstupných parametrov určiť pohyb, ktorý bol vykonaný. Vstupné parametre sú rozdielom aktuálnej a poslednej známej polohy stredného bodu pre ľavú, alebo pravú ruku.
- `getLeftArmMoves()` - funkcia vracia redukovanú postupnosť pohybov ľavej ruky, ktorá je tvorená podľa popisu uvedeného v kapitole 3.6. Funkcia má návratovú hodnotu typu `String`.
- `getRightArmMoves()` - funkcia vracia redukovanú postupnosť pohybov pravej ruky, ktorá je tvorená podľa popisu uvedeného v kapitole 3.6. Funkcia má návratovú hodnotu typu `String`.
- `flushLeftArmMoves()` - funkcia, ktorá slúži na vyprázdnenie postupnosti pohybov ľavej ruky po úspešnej detekcii gesta.
- `flushRightArmMoves()` - funkcia, ktorá slúži na vyprázdnenie postupnosti pohybov pravej ruky po úspešnej detekcii gesta.

4.10 Trieda `ArmBuffer`

Inštancia tejto triedy sa nachádza v triede `Main` a používa sa v triede `Detect` po realizácii detekcie. Reprezentuje postupnosti pohybov rúk pre všetky zdetekované osoby. Trieda obsahuje dve privátne premenné. Prvým je zoznam objektov typu `ArmPosition` a druhým je príznak typu `boolean`, ktorý sa využíva pri vkladaní polohy rúk k rozhodovaniu, či ide o priradenie pohybu k už vytvorenej postupnosti, alebo sa má v zozname vytvoriť záznam pre pohyby rúk ďalšej osoby. Inštanciu tejto triedy je možné vytvoriť pomocou konštruktora bez parametrov, ktorého hlavička vyzerá nasledovne

```
public ArmBuffer().
```

V konštruktore sa vytvorí prázdny zoznam typu `List<Armposition>` a príznak, že ide o vytvorenie novej osoby je nastavený na hodnotu `true`. Ďalej táto trieda

obsahuje nasledujúce funkcie

- `add(int[] dimFace, int[] dimLeftArm, int[] dimRightArm)` - funkcia, ktorá sa volá priamo z triedy `Detect`, a slúži na priradenie pohybu rúk do správnej postupnosti. Na základe pozície stredného bodu tváre dôjde k rozhodnutiu, či sa polohy rúk vložia k nejakej z existujúcich postupností, alebo sa vytvorí nová postupnosť.
- `size()` - táto funkcia vracia počet osôb, pre ktoré existujú postupnosti pohybov rúk.
- `getList()` - táto funkcia vracia zoznam všetkých postupností. Jedná sa o zoznam objektov typu `Armposition`.

5 LADENIE APLIKÁCIE

Na vývoj a testovanie aplikácie bol použitý notebook s procesorom Intel Core i5-3230M s frekvenciou $2,6GHz$ a $8GB$ operačnej pamäte.

Behom vývoja aplikácie som sa stretával postupne s niekoľkými problémami, ktoré ma viedli k finálnemu riešeniu. Upustil som od spôsobu detekcie, kedy boli všetky dielčie klasifikátory aplikované súčasne na celý snímok a následne dochádzalo k spájaniu do výslednej zdetekovanej postavy. Tento prístup sa ukázal ako časovo náročný a nakoniec som zvolil variantu, kedy sa na celý snímok aplikuje len klasifikátor tváre a následne sa nastavujú regióny záujmu postupne pre každý ďalší dielčí klasifikátor.

Ďalšou nutnou úpravou bola zmena rozlíšenia videosekvencie, aby aplikácia spracovanie snímkov nebolo časovo závislé na rozlíšení videosekvencie. Výsledná videosekvencia sa zobrazuje v pôvodnom rozlíšení, no v prípade, že je šírka obrazu väčšia ako $768pixelov (1.2 * 640pixelov)$, dochádza k zmenšeniu videosekvencie na šírku $640pixelov$ pri zachovaní pomeru strán. Toto rozlíšenie považujem za ideálny kompromis medzi časom a kvalitou detekcie.

Asi najzávažnejšou zmenou, ktorá bola behom ladenia aplikácie spravená, bolo pridanie možnosti obmienať klasifikátory a umožniť tak jednoduchú zmenu bez nutnosti úprav v kóde. Dôvodom tejto zmeny boli nedostatočné detekčné schopnosti niektorých dielčích klasifikátorov. Tieto klasifikátory nebolo možné využívať pri ladení metódy postupnej detekcie, ktorá je v práci implementovaná. Zo štvorice klasifikátorov, ktoré som naténoval, fungoval dobre len klasifikátor tváre. V ostatných prípadoch došlo k náhrade klasifikátorov. Ako klasifikátor torza a bôh boli použité klasifikátory, ktoré sú distribuované spolu s OpenCV. Klasifikátor ruky bol nahradený klasifikátorom päste, dostupný na webovej adrese `https://github.com/yandol/GstHanddetect`. Výsledkom teda je odladená metóda, ktorá k správnemu fungovaniu potrebuje dostatočne robusné klasifikátory.

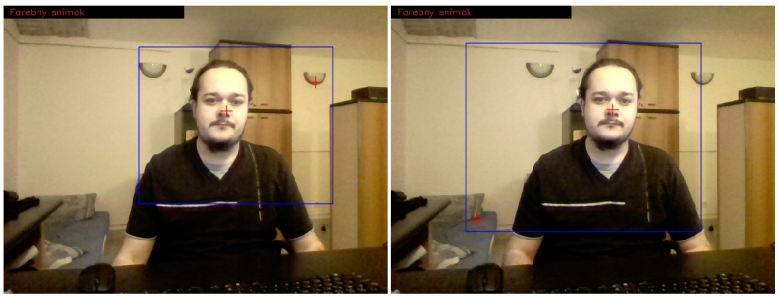
Aj napriek pôvodnému zámeru, dosiahnúť oneskorenie približne $300ms$ je celkové oneskorenie vyššie a je závislé najmä na spracovaní prvého snímku, ktorý trvá zhruba $600 - 700ms$.

6 VÝSLEDKY

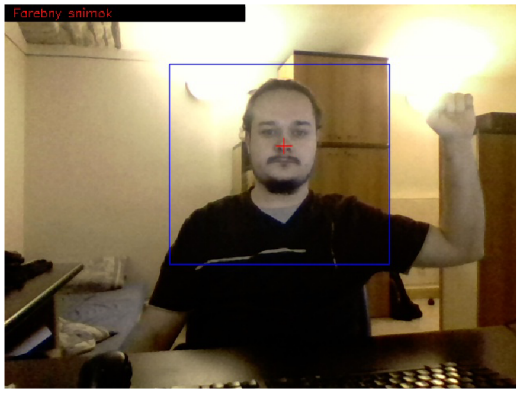
Výsledky prezentované v tejto kapitole sú získané z videosekvencie získanej pomocou web kamery a z videosekvencie, ktorá je na DVD priloženom k práci. V oboch prípadoch ide o videosekvenciu s rozlíšením $640x480px$. V príapde videosekvencie z web kamery bola snímková frekvencia $7sn./s$, v prípade druhej videosekvencie bola snímková frekvencia $24sn./s$. Priemerná doba spracovania jedného snímku, na ktorom došlo k detekcii ľudskej postavy, je zhruba $400ms$. Maximálne hodnoty doby spracovania snímku sa pohybujú okolo $650ms$. Nakoľko sa pri zobrazovaní upravenej videosekvencie kontroluje správne poradie, je maximálna hodnota, čo i len u jedného snímku, automaticky rovná oneskoreniu oproti pôvodnej videosekvencii. Na obrázku 6.1 sú znázornené regióny zaujmu pre jednotlivé časti, na ktorých je vidieť aj ohraničenie regiónu záujmu, ktorý zasahoval mimo hranice snímku. Jedná sa konkrétne o región záujmu pre nohy a ľavú ruku.

Obr. 6.1: Regióny záujmov behom detekcie. Vľavo hore región záujmu pre pravú ruku, hore uprostred pre torzo, pravo hore pre ľavú ruku. V spodnej časti región záujmu pre nohy.

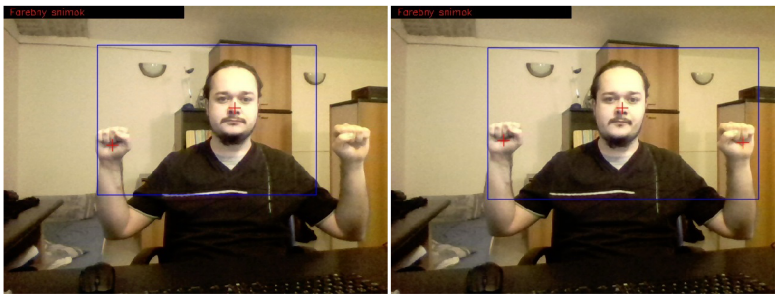
Na obrázku 6.2 sú znázornené chybné detekcie spôsobené pozadím scény. Na obrázku 6.3 je znázornená neúspešná detekcia ľavej ruky vplyvom nadmerného osvetlenia v pozadí. Na obrázku 6.4 sú dva snímky z tesnej bližkosti, pričim na jednom je neúspešne zdetekovaná ľavá ruka. Na obrázkoch 6.5 a 6.6 sú znázornené série snímkov behom vykonávania gesta. Posledný snímok je vždy snímok, na ktorom sa už aplikuje zmena definovaná gestom.



Obr. 6.2: Nesprávne označené detekcie rúk vplyvom pozadia scény.



Obr. 6.3: Nezdetekovaná ľavá ruka vplyvom osvetleniai v pozadí.



Obr. 6.4: Snímky v tesnej blízkosti. Na snímku vľavo nezdetekovaná ľavá ruka, na snímku vpravo správne zdetekované obe ruky.

Obr. 6.5: Séria snímkov, na ktorej sa realizuje gesto „hore, vľavo“.

Obr. 6.6: Séria snímkov na ktorej sa realizuje gesto „dole, vľavo“.

7 ZÁVER

V teoretickej časti bol popísaný obecný postup detekcie ľudskej postavy vo videosekvencii. Následne sú rozobrané jednotlivé kroky a sú popísané najčastejšie používané metódy detekcie, ako aj ich postupný vývoj.

Po preštudovaní rôznych algoritmov bol vybraný algoritmus Viola&Jones s využitím reálne AdaBoostu. Hlavným cieľom bolo implementovať metódu detekcie ľudskej postavy a sledovanie a rozpoznávanie vopred určených gest v programovacom jazyku JAVA. Výsledkom je spustitelná aplikácia, ktorá obsahuje implementovanú metódu. Dosiahnuté výsledky sú popísané v predchádzajúcej kapitole.

Behom realizácie bolo natrénovaných aj niekoľko klasifikátorov, no žiaľ nie všetky tieto klasifikátory dosahovali dostatočne dobré výsledky, a preto boli pri ladení aplikácie, ako aj pri získavaní dát vo výsledkoch použité náhradné klasifikátory, ktoré sú opomenuté v kapitole 5.

V navrhnutej metóde sa neriešia prípady, kedy dielčí detektor označí nesprávne danú časť, ktorá však zodpovedá podmienkam na priradenie ku zdetekovanej osobe (najčastejšie sa jednalo o ruku, ktorá bola nesprávne zdetekovaná vplyvom pozadia scény).

Aplikácia je schopná pracovať s video súbormy, rovnako s videosekvenciou prijímanou z web kamery. Na rozpoznávaciu schopnosť má mierny vplyv pozadie scény, rovnako tak svetelné podmienky.

Aplikácia je schopná pracovať v reálnom čase s konštantným oneskorením, no toto oneskorenie dosahuje hodnoty rádovo v stovkách milisekúnd, čo je pri prijímaní videosekvencie z webkamery citeľné.

LITERATÚRA

[1] CHEN, Y.; CHEN, Ch.: *Fast Human Detection Using a Novel Boosted Cascading Structure With Meta Stages.* [online] [cit. 7. 12. 2012] URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4563465>.

[2] CUTLER, R.; DAVIS, S., L.: *Robust real-time periodic motion detection, analysis, and applications* In Proc. IEEE Transaction on Pattern Analysis and Machune Intelligence, 22(8): 781–796, 2000.

[3] DALAL, N.; TRIGGS, B.: *Histogram of oriented gradients for human detection.* In proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 886–893, 2005.

[4] *Distance transformation.* Wikipedia: The free encyclopedia [online] Vytvorené: 7. 4. 2004. Aktualizácia: 10. 2 2012 [cit. 7. 12. 2012] URL:<http://en.wikipedia.org/wiki/Distance_transform>.

[5] FELZENSZWALB, P.F.; HUTTENLOCHER, D.P.: *Efficient matching of pictorial structures.* In Computer Vision and Pattern Recognition [online]. 2000 [cit. 23. 5. 2014]. URL: <http://people.cs.uchicago.edu/~pff/papers/blobrec2.pdf>.

[6] FREUND,Y.; SCHAPIRE, E. R.: *Experiments with a New Boosting Algorithm.* In Machine Learning: Proceedings of the Thirteenth International Conference [online]. 1996 [cit. 7. 12. 2012] URL:<http://cseweb.ucsd.edu/~yfreund/papers/boostingexperiments.pdf>.

[7] GAVRILA, M., D.; GIEBEL, J.: *Shape-based pedestrian detection and tracking.* In proc. IEEE Intelligent Vehicle Symposium, 1:8–14, 2002.

[8] GAVRILA, M., D.; PHILOMIN, V.: *Real-time Object Detection For „Smart“ vehicles* [online] [cit. 7. 12. 2012] URL:<http://www.cs.cmu.edu/~efros/courses/AP06/Papers/gavrila-iccv-99.pdf>.

[9] MATAS, J.; ŠOCHMAN, J.:*AdaBoost.* [online] [cit. 23. 5. 2014] URL:<http://www.robots.ox.ac.uk/~az/lectures/cv/adaboost_matas.pdf>.

[10] MICILOTTA, S. A.; ONG, J. E.; BOWDEN, R.: *Detection and Tracking of Humans by Probabilistic Body Part Assembly.* [online] [cit. 15. 5. 2014] In proc. of British Machine Vision Conference. URL:<http://cms.brookes.ac.uk/staff/PhilipTorr/BMVC2005/papers/40/micilotta_et_al_BMVC2005.pdf.

[11] MOHAN, A.; PAPAGEORGIOU, C.; POGGIO, T.: *Example-based Object Detection in Images by Components* [online] [cit. 7. 12. 2012] In proc. IEEE Transaction on Pattern Analysis and Machine Intelligence. Vol. 23, 2001, s. 349–361 URL:<http://cbcl.mit.edu/cbcl/publications/ps/mohan-ieee.pdf>.

[12] MU, Y.; YAN, S.; HUANG, T.; ZHOU, B.: *Discriminative Local Binary Patterns for Human Detection in Personal Album.* [online] [cit. 7. 12. 2012] URL:<http://140.114.71.170/upload/activities/25/32.pdf>.

[13] OGALE, N.: *A survey of techniques for human detection.* [online]. [cit. 7. 12. 2012] URL:<http://www-lb.cs.umd.edu/Grad/scholarlypapers/papers/neetiPaper.pdf>.

[14] PAPAGEORGIOU, C.; EVGENIOU, T.; POGGIO, T.: *A Trainable Pedestrian Detection System.* [online] In Proceedings of Intelligent Vehicles. 1998 [cit. 7. 12. 2012] URL:<http://reference.kfupm.edu.sa/content/t/r/a_trainable_pedestrian_detection_system__138592.pdf>.

[15] PAPAGEORGIOU, C.; POGGIO, T.: *A Trainable System for Object Detection.* [online] 2000 [cit. 23. 5. 2014] URL:<http://info.hit-karlsruhe.de/info-ws11/WS11_Human_Detection_HOG/data/a%20trainable%20system%20for%20object%20detection.pdf>

[16] SCHAPIRE, R; SINGER, Y.: *Improved Boosting Algorithms Using Confidencerated Predictions.* [online] 1999 [cit. 7. 12. 2012] URL:<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.9277&rep=rep1&type=pdf>.

[17] SIDENBLADH, H.: *Detecting human motion with support vector machines.* In proc. 17th International Conference on Pattern Recognition, 2:188–191, 2004.

[18] *Support vector machines.* Wikipedia: the free encyclopedia. [online].[] [cit. 7. 12. 2012] URL:<http://cs.wikipedia.org/wiki/Support_vector_machines>.

[19] ŠPANĚL, M.; BERAN, V.: *Obrazové segmentační techniky* [skriptum] Brno: Vysoké učeni technické v Brně, FIT, UPGM, 2005.

[20] UTSUMI, A.; TETSUTANI, N.: *Human detection using geometrical pixel value structures* In Proc. Fifth IEEE Iternational Conference on Automatic Face and Gesture Recognition, page 39, 2002.

[21] VIOLA, P.; JONES, M.: *Rapid object detection using boosted cascade of simple features* In Computer Vision and Pattern Recognition [online] 2001 [cit. 7. 12. 2012] URL:<http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_CVPR2001.pdf>.

[22] VIOLA, P.; JONES, J., M.; SNOW, D.: *Detecting pedestrians using patterns of motion and appearance.* In proc. IEEE International Conference on Computer Vision, 2:734–741

[23] WU, B.; NEVATIA, R.: *Detection of Multiple, Partially, Occluded Humans in a Single Image by Bayesian Combination od Edgelet Part Detectors* [online] In Proc. IEEE International Conference on Computer Vision. 2005 [[cit. 7. 12. 2012]] s. 90–97. URL:<http://iris.usc.edu/Outlines/papers/2007/wu-nevatia-ijcv07.pdf>.

[24] ZHAO, Q.; KANG, J.; TAO, H.; HUA, W.: *Part Based Human Tracking in A Multiple Cues Fusion Framework.* [online] In proc. Pattern Recognition, ICPR 2006. 2006 [cit. 7. 12. 2012], s. 450–455 URL:<http://www.klab.caltech.edu/~qzhao/resources/icpr06.pdf>.

[25] ZHAO, T.; NEVARTIA, R.: *Bayesian Human Segmentation in Crowded Situation* [online] In *Computer Vision and Pattern Recognition* Vol. 3. 2003 [cit. 7. 12. 2012], s. 459-466 URL:<http://iris.usc.edu/outlines/papers/2003/cvpr-zhao.pdf>.

[26] ZHU, Q.; AVIDAN, S; YEH, M.; CHENG, K.: *Fast Human Detection Using a Cascade of Histograms of Oriented Gradients.* [online] [cit. 7. 12. 2012] In proc.

IEEE Computer Society Conference on Computer Vision and Pattern Recognition. URL:<http://www.merl.com/papers/docs/TR2006-068.pdf>.