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

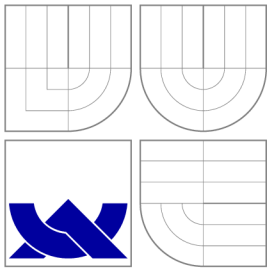
DETEKCE OBLIČEJE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

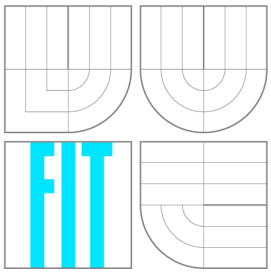
AUTOR PRÁCE
AUTHOR

MIROSLAV ŠTRBA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBLIČEJE

FACE DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MIROSLAV ŠTRBA
AUTHOR

VEDÚCI PRÁCE — Ing. MICHAL HRADIŠ
SUPERVISOR

BRNO 2008

Abstrakt

Táto bakalárska práca obsahuje prehľad súčasných metód detekcie tvárí pomocou klasifikátorov. Obsahuje tiež popis tvorby systému na detekovanie tvárí. V prvej časti sú popísané rôzne metódy na trénovanie klasifikátorov. V druhej sa nachádza analýza, ktorá predchádzala tvorbe systému zameraného na čiernobiele snímky. Implementovaný systém využíva algoritmus WaldBoost a Haarove príznaky. Vo videosekvenciach je možné využiť časticový filter.

Kľúčové slová

detekcia tváre, klasifikátor, AdaBoost, WaldBoost, Haarove príznaky, časticový filter, non-maxima suppression

Abstract

This bachelor thesis contains overview of actual face detection methods using classifier. It also contains description of creating system for face detection. There are described different methods for classifier training in first part. There is analysis, which preceded creation of system focused on black-and-white picture, in second part. Implemented system is using WaldBoost algorithm and Haar features. There is option to use particle filter in video.

Keywords

face detection, classifier, AdaBoost, WaldBoost, Haar features, particle filter, non-maxima suppression

Citácia

Miroslav Štrba: Detekce obličeje, bakalářská práce, Brno, FIT VUT v Brne, 2008

Detekce obličeje

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Hradiša. Uviedol som všetky literárne pramene a publikácie, z ktorých som pri vypracovaní čerpal.

.....
Miroslav Štrba
6. mája 2008

Poďakovanie

Týmto by som chcel poďakovať Ing. Michalovi Hradišovi za odbornú pomoc, konštruktívnu kritiku a cenné rady pri tvorbe bakalárskej práce.

Obsah

Úvod

Táto bakalárska práca nadväzuje na vedomosti získané v rámci predmetu semestrálny projekt (ISP). Detekcia tváre je veľmi aktuálna téma, ktorá svojou podstatou patrí do dvoch oblastí vedy nazývanej *počítačové videnie*. Prvá oblasť, ktorej sa detekcia v obraze týka je analýza obrazu. Tento odbor sa zaoberá vlastnosťami obrazu a v prípade videa napríklad aj detekciou pohybu. Druhá oblasť sa nazýva porozumenie obrazu a zaoberá sa hlavne identifikáciou objektov.

Samotný problém detekcie tváre je nielen v počte rôznych tvarov, ale aj v tom, že každá tvár vyzerá inak. Keď sa osoba usmieva vyzerá inak, ako keď je zamračená alebo nosí okuliare, prípadne má bradu či fúzy. Ako jedným z problémov, ktoré treba riešiť sú aj rôzne svetelné podmienky.

Motivácia

Detekcia tváre sa môže v súčasnosti uplatniť v rôznych oblastiach. Najväčší potenciál by som videl vo zvýšení bezpečnosti. V nasledujúcom odstavci uvádzam niektoré príklady využitia detekcie tváre:

- autentifikácia osôb pri používaný prístroja (notebook, mobil, ...)
- detekcia osôb v oblasti so zakázaným vstupom
- sledovanie pohybujúcich sa účastníkov pri konferencii
- automatické zaostrovanie obrazu v digitálnych fotoaparátoch
- stabilizátor obrazu vo fotoaparátoch
- pri komunikácii robot-osoba

Štruktúra práce

Práca je rozdelená do dvoch hlavných kapitol. V kapitole 1 sú popísané metódy detekcie so zameraním na rôzne spôsoby trénovania klasifikátorov. V ďalšej časti tejto kapitoly sú spomenuté rôzne metódy ako sa vysporiadať s vysokou chybovosťou pri nesprávnych detekciách, alebo akým spôsobom dosiahnuť lepšie výsledky pri detekovaní.

V kapitole 2 je rozpísaný návrh a analýza mnou vybraného spôsobu detekcie tváre. Nachádza sa tu popis tried a sú tu načrtnuté niektoré algoritmy.

Kapitola 1

Metódy detekcie

V tejto kapitole budú rozobraté základné princípy detekcie ľudskej tváre v súčasnosti. Jedným zo spôsobov ako rozhodovať, či oblasť obsahuje tvár alebo nie, je pomocou klasifikátorov. Na túto oblasť sa zameriam.

1.1 Klasifikátory

Klasifikátor je špeciálna funkcia alebo modul, ktorý dokáže zhodnotiť s akou pravdepodobnosťou sa na danom mieste nachádza skúmaný objekt. Klasifikátory rozdeľujeme na:

- binárne – vráti hodnotu „áno“ alebo „nie“ v závislosti na výsledku klasifikácie
- viachodnotové – vráti hodnotu v určitom rozmedzí, ktorá reprezentuje pravdepodobnosť s akou sa detekovaný objekt v obraze nachádza

1.1.1 Klasifikátory trénované pomocou neurónových sietí

Neurónová sieť je paralelný systém, ktorý sa používa na modelovanie vzťahov medzi vstupmi a výstupmi alebo na porovnávanie pomocou vzoru. Funguje ako „čierna skrinka“ pozostávajúca zo vstupov a výstupov. Pričom minimálny počet vstupov a výstupov je 1. Z pomenovania vyplýva, že je určená na simuláciu správania ľudského mozgu. Využíva sa to najmä v prípade, že je potrebné riešiť úlohy, ktorých matematické riešenie je veľmi zložité.

Činnosť neurónovej siete podľa [14] je charakterizovaná funkciou:

$$f\colon X \to Y \tag{1.1}$$

Rozdelenie

Sieť pozostáva z jednotlivých častí nazývaných *neuróny* alebo *uzly*. Neurón prijíma N vstupov a vracia M výstupov. Poznáme tri typy neurónov:

- vstupné – neuróny, ktorých vstupmi sú signály z prostredia
- skryté – neuróny, ktoré sú vstupmi aj výstupmi spojené s inými neurónmi
- výstupné – neuróny, ktorých výstup vedie do prostredia

Podľa [16] sa neurón spracovávajúci informáciu riadi nasledujúcim pravidlom:

$$o_i^{k+1} = f(\sum_{j=1}^{N} w_{ij}^k \times o_j^k - \Theta_i^{k+1}) \tag{1.2}$$

kde:

$0 < i \leq M$	výstup
$0 < j \leq N$	vstup
o_i^{k+1}	vstupná hodnota i-tého neurónu $k+1$ vrstvy
k	index vrstvy
Θ_i^{k+1}	prah excitácie i-tého neurónu $k+1$ vrstvy
w_{ij}^k	váha spojenia medzi j-tým neurónom k-tej vrstvy a i-tým neurónom $k+1$-tej vrstvy
$f()$	ľubovoľná monotónna funkcia

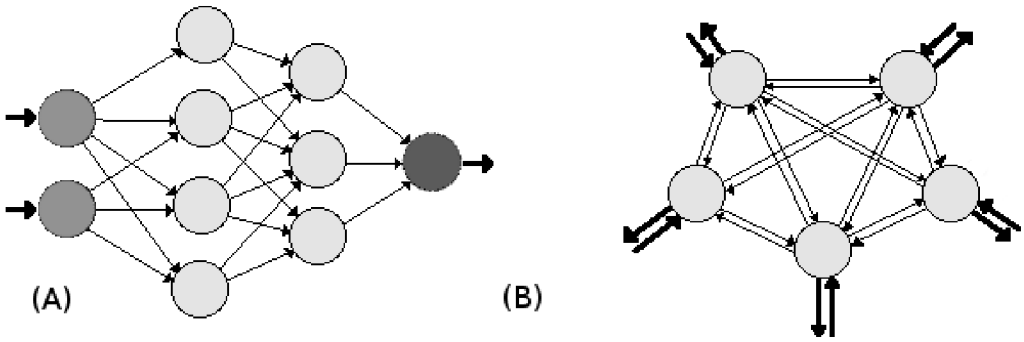
Samotná neurónová sieť môže byť rozdelená do viacerých vrstiev. Táto možnosť sa využíva pri väčšej zložitosti riešeného problému. V závislosti na toku dát je možné podľa [1] rozdeliť neurónové siete do dvoch kategórii:

- **Dopredné** – ang. *feed-forward*

  signál sa šíri iba od vstupných neurónov cez skryté neuróny k výstupným neurónom. Príklad na obr. 1.1(A)

- **Rekurentné** – ang. *recurrent*

  signál sa môže pohybovať aj smerom od výstupov ku skrytým častiam alebo dokonca až k vstupom. Príklad na obr. 1.1(B)



Obrázok 1.1: (A) Dopredná neurónová sieť s dvomi vstupnými, siedmimi skrytými a jedným výstupným uzlom. Obsahuje 4 vrstvy. (B) Rekurentná neurónová sieť obsahujúca neuróny, ktoré sú zároveň vstupné aj výstupné. Neobsahuje skryté neuróny a neuróny nie sú organizované vo vrstvách.

Podstatnou vlastnosťou neurónových sietí je schopnosť učiť sa. Učenie je dôležitá fáza, ktorá sa musí nachádzať pri každej tvorbe siete. V tejto fáze sa upravujú jednotlivé váhy spojení neurónov tak, aby dosiahli požadované výsledky alebo výsledky, ktoré majú k tejto hodnote najbližšie. Správnym naučením sa zaoberá oblasť umelej inteligencie *počítačové učenie* (podrobnosti je možné nájsť v [15]). V oblasti detekcie tváre sa využíva napr. metóda propagácie chyby.

Hodnotenie

Príklad klasifikátoru vytvoreného pomocou neurónovej siete je popísaný v [10]:

> Obsahuje tri typy skrytých uzlov: 4-krát typ, ktorý skúma oblasti 10x10 pixelov, 16-krát typ, ktorý skúma oblasti 5x5 pixelov a 6-krát typ, ktorý skúma prekrývajúce sa 20x5 pixelové horizontálne pruhy. Horizontálne pruhy umožňujú skrytým neurónom detekovať rysy ako sú ústa alebo pár oči, zatiaľ čo skryté neuróny so štvorcovými typmi môžu detekovať rysy tváre ako osamotené oči, nos alebo kútiky úst.

Na záver tejto časti sú uvedené výsledky klasifikátorov natrénovaných podľa [10] bez použitia heuristiky. Testovacia sada pozostávala so 130 obrázkov, na ktorých sa nachádzalo 507 tvári. Výsledky sú v tabuľke 1.1.

klasifikátor	Detekované tváre	Zlé detekcie
1	91.1 %	945
2	92.5 %	862
3	90.9 %	738
4	92.1 %	819

Tabuľka 1.1: Úspešnosť klasifikátorov natrénovaných pomocou neurónových sietí

Následne ešte boli urobené ďalšie testy, kde s akceptovateľnou chybovosťou bola dosiahnutá úspešnosť od 77.9 % – 90.3 %.

1.1.2 Klasifikátory trénované pomocou SVM

Ako sa uvádza v [2] Support Vector Machine (ďalej len SVM) je metóda postavená tak, aby mohla čo najlepšie generalizovať informácie pri trénovaný. Samotné trénovanie je však časovo aj pamäťovo náročné.

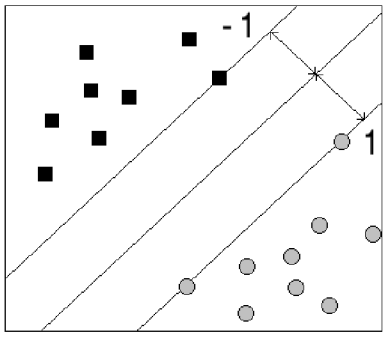
Rozdelenie lineárne oddeliteľných tried

Podstatou SVM je vytvorenie hyperroviny, ktorá zo vstupných dát vytvorí dve sady vektorov v n-rozmernom priestore. Pre každú sadu sa potom vytvorí takáto hyperrovina, ktorá je znova použitá ako vstupná sada.

Vysvetlenie SVM klasifikátora sa nachádza v [9]. V tomto dokumente je popísaný jednoduchý príklad, kde sa nachádzajú dve spojené množiny, ktoré sú lineárne oddeliteľné. Cieľom SVM je vo vstupnej sade $D = \{(X_i, y_i)\}_{i=1}^{n}$, kde $y \in (-1, 1)$, nájsť optimálne lineárne riešenie na základe minimalizácie rizika. Riešením je hyperrovina, ktorá necháva medzi týmito triedami čo najväčší okraj. Okraj roviny je definovaný ako súčet vzdialeností hyperroviny od najbližšieho bodu oboch tried. Ukážka fungovania SVM je na obrázku 1.2. Objekty, ktoré boli ohodnotené -1 patria do jednej skupiny, zatiaľ čo objekty v druhej skupine majú výstupnú hodnotu od SVM klasifikátora 1.

Rozdelenie lineárne neoddeliteľných tried

V prípade, že ide o lineárne neoddeliteľnú množinu je cieľom SVM klasifikátora nájsť riešenie, kde je maximálny okraj a minimálne množstvo zlých klasifikácii. Výmena medzi



Obrázok 1.2: SVM model: ukážka maximalizovanej vzdialenosti pri rozdeľovaný dvoch tried

zlými klasifikáciami a okrajom je ovplyvňovaná konštantou C. Výsledkom je teda rovnica z dokumentu [9]:

$$f(x) = sign\left(\sum_{i=1}^{n} \lambda_i y_i X \cdot X_i + b\right) \tag{1.3}$$

kde koeficient λ je riešením:

$$\begin{aligned} &\text{minimalizuj } \Lambda \qquad \text{pre } W(\Lambda) = -\Lambda \cdot I \ + \frac{1}{2}\Lambda \cdot D\Lambda \\ &\qquad\qquad\quad \text{ak } \Lambda \cdot y = 0,\ \Lambda - CI \leq 0,\ \Lambda \geq 0 \\ &\qquad\qquad\quad \text{kde } (\Lambda)_i = \lambda_i,\ (I)_i = 1,\ D_{ij} = y_i y_j X_i \cdot X_j \end{aligned} \tag{1.4}$$

Riešením vznikne iba malý počet koeficientov, ktoré sú rôzne od 0. Každý koeficient korešponduje iba s častou dát a tak výsledným riešením je súbor bodov asociovaných s nenulovými koeficientmi. Tieto body sa nazývajú *support vectors*.

V prípade detekcie tváre je však problém oddeliteľnosti tried „tvár“ a „netvár“ prostredníctvom hyperroviny extrémne zložitý. Ak ale využijeme transformáciu do priestoru príznakov F, $(\Phi(X_1), y_1), \ldots, (\Phi(X_n), y_n) \in F \times Y$, tak vznikne možnosť na lineárnu klasifikáciu. Tento priestor má viacej dimenzií ako pôvodný a pri transformáciách sa využíva tzv. jadrových funkcií (ang. *kernel tricks*). Tie dokážu podľa [2] efektívne vypočítať skalárny súčin aj bez toho aby poznali mapovanie Φ.

Vlastnosti a hodnotenie

Hlavnou nevýhodou SVM detektorov je, že kernel funkcie a parametre je treba určiť ručne. Kernel funkcie, ktoré navrhol tvorca SVM modelu V. Vapnik v [3] sú Gausian Radial Basis, Polynomická stupňa d a viacvrstvová perceptronová funkcia.

Klasifikátory, ktoré sú popísané v [9], boli natrénované pomocou SVM^{light} na trénovacej sade, ktorá obsahovala 2375 tvári a 4285 vzorov,ktoré tvár neobsahovali. Bol zvolený polynomický kernel stupňa 2 a konštanta $C = 200$. Výsledkom bolo, že na testovacej sade pozostávajúcej zo 104 obrázkov, na ktorých sa nachádzalo celkovo 277 tvári pri rozhodovacej úrovni 0 bolo úspešne nájdených 88,8 % tvári a počet nesprávnych detekcií obrazu bolo 496. Pri rozhodovacej úrovni -0,0168 sa úspešnosť detekcie zvýšila na 90,3 %, ale zvýšil sa aj počet nesprávne detekovaných „netvárových“ oblastí na 507.

1.1.3 Klasifikátory trénované pomocou AdaBoostu

AdaBoost je skrátený anglický názov pre *adaptive boosting*. Tento algoritmus dokáže výrazne znížiť chybu ľubovoľného učiaceho algoritmu, ktorý dokáže dávať výsledky lepšie ako náhodné tipovanie. AdaBoost je adaptabilný z dôvodu využívania viacerých slabých klasifikátorov (ang. *weak classifier*) a pri učení sa využíva množina predtým zle zaradených objektov na natrénovanie nasledujúcich klasifikátorov. Výsledným výtvorom je jeden robustný klasifikátor (ang. *strong classifier*).

Podstatou prispôsobovania sa zle klasifikovaným objektom je algoritmus, v ktorom sa ukladajú váhy jednotlivých klasifikátorov. V prípade, že sa práve zle zaklasifikuje objekt je váha klasifikátora, ktorý tento objekt zdetekoval posilnená, čím sa dosiahne toho, že v nasledujúcom trénovaní algoritmus dokáže svoje rozhodnutia opraviť.

Vlastnosti

Samotný algoritmus je citlivý na všetky objekty. To znamená, že sa snaží dosiahnuť, aby každý objekt bol zaradený do správnej množiny. Táto vlastnosť nie je pre trénovanie klasifikátorov na detekciu tvári moc vhodná, ale vzhľadom na to,že je AdaBoost odolný voči pretrénovaniu (pri vysokom počte prvkov v trénovacej sade nemôže výsledný klasifikátor dávať horšie výsledky ako keby bolo v trénovacej sade prvkov menej) je možnosť túto vlastnosť výrazne eliminovať.

Postup trénovania

Popis fungovania AdaBoostu je vysvetlený v dokumente [5], kde je popísaná základná varianta algoritmu 1.1.1. Dokument [6] obsahuje dve rozšírenia pre varianty, keď je potrebné zatriediť objekty do viacerých skupín.

Algoritmus 1.1.1 popisuje fungovanie AdaBoostu

Vstup: sekvencia vstupov $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
trénovací algoritmus na slabé klasifikátory v príklade nazývaný ako **LearnAlg**
číslo T udávajúce počet iterácií trénovania

Inicializácia: $D_1(i) = 1/m$ pre každé i
pre: $t = 1, 2, \ldots, T$

1. Urči pomocou **LearnAlg** a s využitím D_t hodnotu vrátenú každým klasifikátorom
2. Vráť sa k hypotéze $h_t = X \to \{-1, +1\}$
3. Vypočítaj chybu h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_i(t)$.
   Ak $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
4. Vyber $\alpha_t = \frac{1}{2} \ln \left(\frac{(1-\epsilon_t)}{\epsilon_t} \right)$.

5. Obnov

$$D_t\colon D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{ak } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{ak } h_t(x_i) \neq y_i \end{cases} \tag{1.5}$$

$$= \frac{D_t(i)\ \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{1.6}$$

Kde Z_t je normalizačná konštanta zvolená tak, aby funkcia D_{t+1} zostala distribučným rozložením.

Výstup: hypotéza $h_{fin}(x) = \text{sign}\ \big(\sum_{t-1}^{T} \alpha_t h_t(x)\big)$

Výsledkom je funkcia, ktorá je váhové ohodnotenie slabých klasifikátorov.

WaldBoost

V predchádzajúcej časti bol popísaný algoritmus AdaBoost, ktorého nevýhodou je, že nie je možné rozhodnúť, či daná prehľadávaná oblasť patrí alebo nepatrí do určitej skupiny skôr, ako sa prepočítajú hodnoty zo všetkých slabých klasifikátorov, aj keď je výsledok zrejmý už počas vyhodnotenia časti klasifikátorov. Na riešenie tohto nedostatku vznikla modifikácia AdaBoostu z názvom *WaldBoost* popísaná v [18]. Klasifikátor natrénovaný pomocou WaldBoostu nemusí byť vyhodnotený celý. V prípade, že počas vyhodnocovania slabým klasifikátorom sa prekročí určená spodná alebo vrchná hranica oblasť je zaradená do danej skupiny. V opačnom prípade sa skúmaná oblasť predá nasledujúcemu klasifikátoru.

Algoritmus 1.1.2 popisuje fungovanie WaldBoostu

Vstup: sekvencia vstupov $\langle (x_1, y_1), \ldots, (x_m, y_m)\rangle$; $x_m \in X, y_m \in \{-1, +1\}$

Inicializácia: váha $w_1(x_i, y_i) = 1/m$ pre každé i
horná hranica $A = \frac{1-\beta}{\alpha}$ a dolná hranica $B = \frac{\beta}{1-\alpha}$

pre: $t = 1, 2, \ldots, T$

1. Vyber h_t pomocou rovnice $h^{T+1} = \frac{1}{2} \log \frac{P(y=+1|x, w^{(T)}(x,y))}{P(y=+1|x, w^{(T)}(x,y))}$
2. Odhadni pravdepodobnosť $R_t(x) = \frac{p(h^1, h^2, \ldots, h^t(x)|y=-1)}{p(h^1, h^2, \ldots, h^t(x)|y=+1)}$
3. Nájdi hraničné hodnoty θ_A^t a θ_B^t
4. Odstráň vzorky z trénovacej sady pre ktoré platí $H_t \geq \theta_B^t$ alebo $H_t \leq \theta_A^t$
5. Zaraď vzorky do trénovacej sady

Výstup: silný klasifikátor H_t a hraničné hodnoty θ_A^t a θ_B^t

---

Intuitívne sa dá predpokladať, že rýchlosť WaldBoostu je ovplyvnená počtom slabých klasifikátorov. Z prieskumu v [18] je pri počte 600 klasifikátorov priemerná rýchlosť $\bar{T}_s = 13.92$ a pri počte 300 je hodnota $\bar{T}_s = 9.57$.

1.2 Viola & Jones

V tejto sekcii je popísané využitie AdaBoostu ako metódy pre detekciu tvárí vo videu. S touto myšlienkou prišli páni P. Viola a M. Jones, ktorú prezentovali v dokumente [13]. Základom ich úspechu boli 3 veci, vďaka ktorým bola detekcia veľmi efektívna, čo do rýchlosti a dosahovala nízku chybovosť. Boli to:

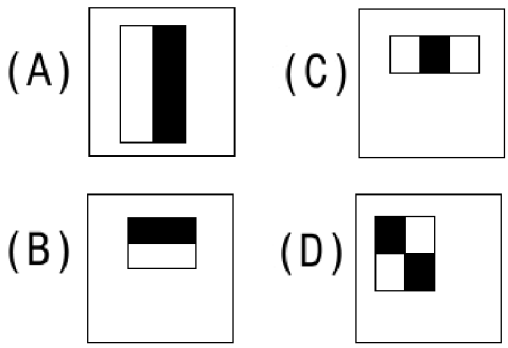
- AdaBoost využívajúci Haarove príznaky
- Integrálny obraz
- Zapojenie klasifikátorov do kaskády

1.2.1 Haarove príznaky

Haarove príznaky sú jednoduché obdĺžnikové oblasti, ktorých hodnota sa vypočítava z intenzity obrazu. V [17] sú ukázané 3 typy Haarovych príznakov, ktoré sú zobrazené na obr. 1.3. Z týchto príznakov sa dajú pomocou rotácii jednoducho vygenerovať ostatné typy príznakov. Výpočet jedného príznaku pozostáva zo sumy bielych oblastí, od ktorých sa odpočíta suma čiernych oblastí:

$$f(x) = \sum_{w \in W} x(w) - \sum_{b \in B} x(b) \tag{1.7}$$

kde: pixel o intenzite x nachádzajúci sa v oblasti skúmaného Haarovho príznaku patrí do skupiny W v prípade, že sa nachádza v bielej oblasti. V opačnom prípade patrí do skupiny B tj. ak sa nachádza v čiernej oblasti Haarovho príznaku.



Obrázok 1.3: 4 typy Haarovych príznakov od Viola & Jones

Haarove príznaky sú založené na sumách obdĺžnikových oblastí. Výpočet obĺžnikových oblastí sa môže vykonávať obyčajným sčítaním, ktoré je citlivé na veľkosť sčítavanej oblasti alebo sa môže na výpočet použiť tzv. integrálny obraz.

1.2.2 Integrálny obraz

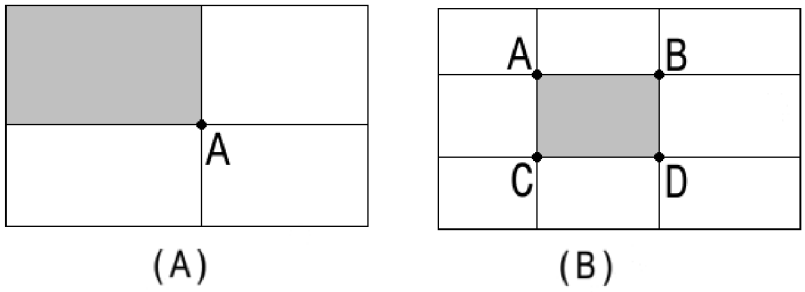
Integrálny obraz dokáže výrazne urýchliť sumáciu obsahu obdĺžnikových oblastí. Jeho hlavnou funkciou je v konštantnom čase vypočítať ľubovolne umiestnený a veľký obdĺžnikový

výrez. Veľkosť integrálneho obrazu je o jeden stĺpec širší a o jeden riadok vyšší. Samotné hodnoty integrálneho obrazu II sú súčtom hodnôt intenzity pixelov obrazu I naľavo a navrch do pozície pixelu v obraze (obrázok 1.4(A)). Z tohto je zrejmé, že pixel umiestnený v pravom dolnom rohu je súčtom intenzít v celom obraze, zatiaľ čo prvý riadok a stĺpec majú hodnotu 0.

$$II(x,y) = \sum_{i,j}^{x,y} I(i,j) \tag{1.8}$$

Výsledkom je, že v prípade keď poznáme pozíciu rohových bodov, označených ako **A, B, C, D**, môžeme pomocou jednoduchej rovnice 1.9 vypočítať intenzitu obdĺžnikovej oblasti. Podstata je zachytená na obrázku 1.4 (B).

$$I_{obd} = A \ - B - C + D \tag{1.9}$$



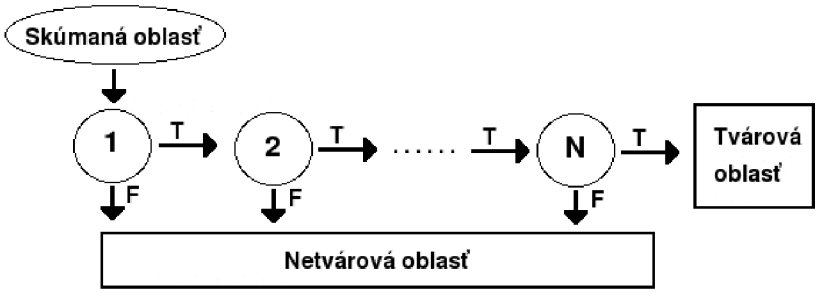
Obrázok 1.4: Integrálny obraz: (A) hodnota bodu A reprezentuje šedú oblasť. (B) oblasť pri využití rovnice 1.9

1.2.3 Kaskádové zapojenie

V skúmaných oblastiach obrazu sa nenachádza vo väčšine prípadov tvár. V snahe zníženia chybovosti a zvýšenia efektivity sa začalo využívať kaskádového zapojenia.

Popis kaskádového zapojenia vychádza z dokumentu [17]. Stupne v kaskáde sú natrénované pomocou AdaBoostu, pričom každý je schopný rýchlo rozhodnúť či je daný výrez v obraze tvár. Tieto klasifikátory opäť nemusia dosahovať vysokej bezchybnosti. Vstupný výrez obrazu je najskôr skúmaný jedným klasifikátorom a v prípade, že ho tento označí za oblasť, kde sa tvár nenachádza, sa táto hodnota vráti. V opačnom prípade sa predá prehľadávaná oblasť ďalšiemu klasifikátoru. Tento znova preskúma oblasť a rozhodne či sa tam tvár nachádza, alebo nie. Výsledok opäť vráti alebo posunie ďalšiemu klasifikátoru. Ak sa skúmaná oblasť dostane až k poslednému klasifikátoru (tzn. všetky predchádzajúce klasifikátory označili oblasť za tvárovú) môže rozhodnúť či sa ozaj v danom výreze nachádza tvár. Postup je schematicky znázornený na obrázku 1.6.

Výhodou takéhoto prístupu je odstránenie oblastí, ktoré neobsahujú tvár už na začiatku, kde môžu byť klasifikátory výrazne jednoduchšie a rýchlejšie, ako klasifikátory na konci kaskády, kde sa môžu nachádzať s podstatne väčšou zložitosťou. Práve na základe tejto myšlienky je ušetrený čas na detekovanie.



Obrázok 1.5: Schematické zobrazenie kaskády

1.3 Štatistické modelovanie

Štatistické modelovanie je určené na eliminovanie zmeny objektu v prípade, že daný objekt sa vzhľadom na pozorovateľa pootočí (ľudská tvár vyzerá úplne inak spredu ako zboku). Túto metódu vymysleli páni Henry Schneiderman a Takeo Kanade a jej detailný popis sa nachádza v [12]. Z tohto dokumentu vychádza aj hlavná časť tejto sekcie a ak nebude uvedené inak, tak pri písaní tejto časti som čerpal z tohto dokumentu.

1.3.1 Podstata štatistického modelovania

Štatistické modelovanie ako algoritmus využíva viacero klasifikátorov, ktoré je možné natrénovať vybraným spôsobom. Každý z množiny klasifikátorov je natrénovaný na určitú polohu objektu voči pozorovateľovi. Keďže rozdiel medzi jednotlivými klasifikátormi spočíva práve v závislosti pohľadu nazývame ich anglickým termínom *view-based detectors*. Podľa empirického zistenia je vhodný počet klasifikátorov pre detekciu tváre 2. Jeden pre vyhľadávanie tváre spredu a druhý pre tvár z profilu. Zvyšné polohy sú práve dopočítavané.

Pre každý detektor sa následne vytvoria dve štatistické rozloženia: $P(obraz|tvar)$ a $P(obraz|netvar)$. Následne sa podľa rovnice testu na pravdepodobnosť výskytu:

$$\frac{P(obraz|tvar)}{P(obraz|netvar)} > \lambda \qquad \left(\lambda = \frac{P(obraz|netvar)}{P(obraz|tvar)}\right) \tag{1.10}$$

rozhodne či sa na danom mieste tvár nachádza, alebo sa tam nachádza „netvar“, čiže je daná oblasť okolie. Problémom je vytvorenie $P(obraz|tvar)$ a $P(obraz|netvar)$, lebo nie sú známe skutočné charakteristiky týchto rozložení. Riešením môže byť použitie flexibilného modelu rozloženia so širokým rozsahom napr.„parzen windows“ alebo „najbližší sused“ (nearest neighbor). Oba tieto modely ale spotrebujú obrovské množstvo času pri výpočte. Ďalším riešením je využitie histogramov, kde sa počas trénovania zbierajú údaje o tom ako často sa objavujú jednotlivé zložky v trénovacej množine.

1.3.2 Hodnotenie

Ukážka úspešnosti tohoto postupu z [12] pri testovacej sade pozostávajúcej z 208 obrázkov, v ktorých sa nachádzalo 441 tvári. Z týchto tvári bolo 347 otočených smerom k snímaciemu zariadeniu. Symbol λ je premenná z rovnice 1.10.

λ	Dobré detekcie (všetky tváre)	Dobré detekcie (profil)	Zlé detekcie
0.0	92.7 %	92.8 %	700
1.5	85.5 %	86.4 %	91
2.5	75.2 %	78.6 %	12

Tabuľka 1.2: Úspešnosť štatistického modelovania

1.4 Detekcia na základe farby pokožky

Táto sekcia vychádza prevažne z dokumentu [2] a bude venovaná detekovaniu tváre na základe farby pokožky. Keďže farba ľudskej kože je výnimočná aj napriek množstvu jednotlivých variácii. Pokožka môže byť rôzne nasvietená alebo snímacie zariadenie môže byť nastavené rôznymi spôsobmi a v neposlednom rade je to samotná pigmentácia pokožky.

1.4.1 Farebný priestor

Existujú rôzne modely reprezentujúce farbu. Najznámejší model je *RGB*, ktorý sa skladá z troch farebných zložiek, kde R reprezentuje červenú farbu (ang. Red), G reprezentuje zelenú farbu (ang. Green) a B reprezentuje modrú farbu (ang. Blue). Tento spôsob udávania farieb je v súčasnej dobe asi najrozšírenejší, keďže sa využíva hlavne na zakódovanie konkrétnej farby do obrazu.

Ďalším známym modelom je model *HSV*, ktorého skratka pochádza z anglických názvov pre farebný tón (ang. Hue), sýtosť (ang. Saturation) a hodnotu jasu (ang. Value). Tento model je oproti RGB modelu vhodnejší pre nastavovanie požadovanej farby užívateľom, keďže užívateľ nemusí rozmýšľať nad tým, ktorou farebnou zložkou dosiahne požadovanú farbu. Je mu bližšie meniť hodnoty podľa farebného tónu a jasu.

1.5 Vylepšenie vlastností klasifikátorov

Táto sekcia je určená na popísanie možností ako vylepšiť niektoré z vlastností klasifikátorov.

1.5.1 Spájanie detekcií

Väčšina klasifikátorov v prípade, že sa na danom mieste nachádza tvár, označí túto oblasť viacnásobne. Buď tým, že je oblasť skúmania posunutá v horizontálnom, prípadne vertikálnom smere a obsahuje prekrývajúce sa oblasti označujúce tvár, alebo tým, že daný klasifikátor je zväčšený a preskúmava tvárovú oblasť aj s okolím. V prípade [10] zistili, že ak vzniknú práve takéto zhluky detekcií je istota správnej detekcie oveľa vyššia. Naopak, ak sa tieto zhluky nevytvoria a je na danej oblasti len jedna detekcia tváre, tak je veľká pravdepodobnosť, že daný výsledok je chybný.

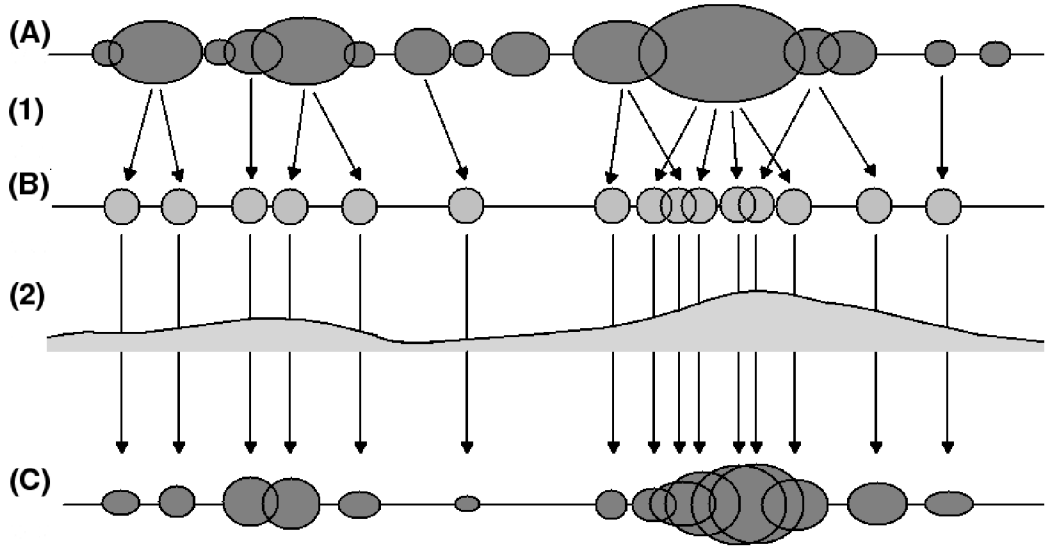
Týmto spôsobom sa dajú výrazne ovplyvniť zlé detekcie. Nevýhodou takejto anulácie daných jednorázových oblastí môže byť, že sa niektoré z dobre nájdených tvárových oblastí môžu odstrániť a tým pádom klesne aj úspešnosť detekovania tvárových oblastí.

1.5.2 Časticový filter

V tejto časti sa budem zaoberať algoritmom na sledovanie pohybu a bude vychádzať z dokumentu [7] a z [8]. Časticový filter (ang. *Particle Fiter*) je založený na metóde Monte Carlo. Cieľom je vytvorenie vektoru X_t, ktorý pomôže systému v diskrétnom čase na základe predchádzajúcej hodnoty určiť nasledujúcu pozíciu detekcie. Tento popis je vyjadrený vzťahom:

$$X_t = F_t(X_{t-1}, V_t) \tag{1.11}$$

V tejto rovnici V_t je náhodný rozptyl. Pričom funkcia F_t dokáže vypočítať novú hodnotu častice.



Obrázok 1.6: Schematické zobrazenie „života“ častíc: (A) častice v kroku t, (B) častice s normalizovanými váhami, (C) výsledné častice, (1) náhodný posun vybraných častíc, (2) meranie miest reprezentujúcich časticami

Postup pri určovaní detekcie pomocou časticového filtra sa skladá z niekoľkých krokov znázornených na obrázku 1.6. Pri kroku (1) sa využívajú častice (A), ktoré majú svoju váhu. Váha je v prípade detekcie tváre závislá na pravdepodobnosti s akou sa nachádza tvár na mieste, ktoré častica reprezentuje. Inicializácia môže byť prevedená nastavením rovnakých váh pre všetky častice. Samotný krok potom reprezentuje náhodný pohyb vybraných častíc, pričom sú náhodne vyberané častice v závislosti na ich váhe. Vybrané častice potom majú rovnakú váhu (B). Pri ďalšom kroku (2) sa odmeria hodnota miesta, ktorú daná častica reprezentuje. Podľa tejto hodnoty sa znova nastavia váhy samotným časticiam (C).

1.5.3 Non-maxima suppression

Táto metóda je určená na elimináciu viacerých detekcií jednej oblasti. V súčasnosti sa pri spracovaní obrazu využíva najmä v algoritmoch, kde nie je exaktne jasné, ktorý bod danú podmienku splňuje a ktorý nie. Pri riešení takýchto úloh vznikajú oblasti s miestami, kde pri očakávaní jedného objektu sa objaví objektov niekoľko napr. pri detekcii hrán, rohov. Tieto body je treba odstrániť a vybrať jeden, optimálne taký, ktorý dosahuje najlepšie výsledky.

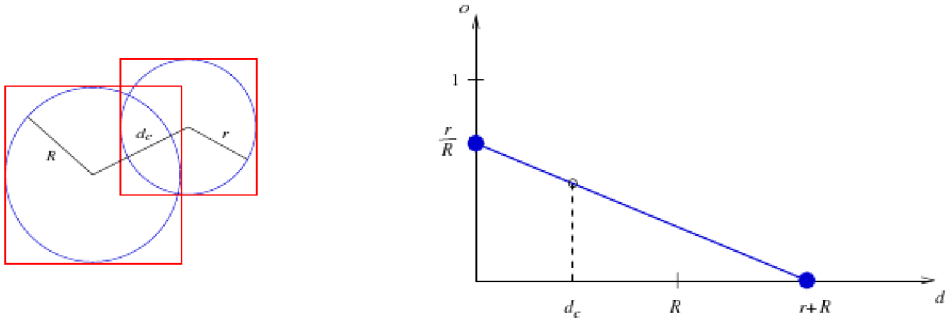
Detekcia rohov

Non-maxima suppression dosahuje najvýznamnejšie výsledky v prípade detekcie rohov, kde môže byť jeden roh detekovaný aj 10-krát. Keďže v obraze samotnom sa nachádza veľké množstvo takýchto oblastí je žiadúce túto metódu využívať, čím sa výrazne zníži počet detekcií a niekoľkonásobne sa potom urýchli ich spracovanie.

Non-maxima suppression pri detekcii rohov využíva zmeny intenzity jednotlivých susediacich pixelov. Čím je rozdiel medzi susednými pixelami väčší, tým je pravdepodobnosť rohu vyššia. V prípade detekovania rohu sa preto vezme okolie bodu a porovná sa pravdepodobnosť jednotlivých bodov okolia s aktuálnym bodom. Pokiaľ nie je bod lokálne maximum, tak to nie je ani rohový bod. Pričom okolie je dopredu určené a býva v rozsahu niekoľkých pixelov.

Detekcia tvári

V prípade detekcii tváre je postup non-maxima suppression o niečo zložitejší, keďže sa jedná o vyhľadanie detekcie, ktorá najlepšie reprezentuje danú tvárovú oblasť. Vychádzajúc z dokumentu [18], na detekciu tváre nemôže byť využité hľadanie lokálneho maxima, pretože sa nedá jednoznačne určiť, aké veľké má byť okolie zdetekovaných objektov. V takomto prípade sa využíva spôsob, kde sa berú do úvahy len už zdetekované oblasti, pričom dve detekcie sú zlúčené do jednej v prípade, ak ich prekrývajúca sa oblasť je väčšia ako určená hranica. Výsledkom je jedna detekcia pre každú oblasť s najvyšším ohodnotením.



Obrázok 1.7: (A) Dve prekrývajúce sa detekcie (B) Interpolácia dvoch prekrývajúcich sa detekcií. Obrázok pochádza z dokumentu [18].

Výpočet prekrývajúcich sa oblastí je znázornený na obrázku 1.7. Kde r je polomer malého kruhu a R je polomer veľkého kruhu pričom d_c reprezentuje vzdialenosť medzi týmito dvomi kruhmi. Jednoduchým spôsobom dokážeme určiť dve hodnoty prekrývania sa zo zadaných polôh. Prvú hodnotu ľahko určíme v prípade, že sú kružnice sústredné. Ich prekrytie sa dá vyjadriť ako $\frac{r}{R}$. Druhou hodnotou je 0, ktorá je výsledkom v prípade, že sú kružnice vzdialené minimálne súčtu ich polomerov. Pri vynesení týchto bodov na graf a použitím lineárnej interpolácie nám vznikne 1.7 (B) pričom prekryv detekcii je vyjadrený rovnicou:

$$o = \frac{r}{R}\left(1 - \frac{d_c}{r+R}\right) \tag{1.12}$$

Kapitola 2

Tvorba systému

Zadaním bolo preštudovať základy spracovania obrazu a vytvoriť si prehľad o súčasných metódach detekcie tváre v obraze. Na základe týchto znalostí bolo požadované vytvoriť systém na detekciu tváre v reálnom čase. Táto kapitola je určená na popis vývoja modulu, ktorý má aplikovať klasifikátory uložené v XML súbore na detekciu tváre. Vývoj systému pozostával z 3 fáz:

- Analýza — obsahuje rozobratie problému
- Návrh — obsahuje výber vhodného postupu
- Implementácia – obsahuje samotnú tvorbu systému

2.1 Analýza

Ľudská tvár má mnoho podôb a je náročné nájsť optimálny spôsob ako určiť polohu tváre. Na druhú stranu každá obsahuje črty, ktoré sú pre všetkých ľudí rovnaké. Takýmito črtami sú napr. oči, nos alebo ústa. V neposlednom rade je unikátna farba kože v rámci prírody.

2.1.1 Spracovanie podľa farby

Jednou z najintuitívnejších metód ako určiť či daná farba patrí do určitej kategórie je zaradiť ju tam podľa rozsahu jej farebných zložiek. Znamená to, že napríklad v RGB modeli sú kategórie rozdelené pomocou pravidiel tvaru: *ak* $R > G$ *a* $R > B$, prípadne inými podobnými. Zložitejšie pravidlá sa empiricky určujú zložito a preto sa na ich tvorbu využívajú rôzne metódy strojového učenia.

V prípade klasifikácie farby je rozumnejšie používať HSV model, lebo je možné ignorovať hodnotu jasu, ktorá je jedným z faktorov, ktorý mení v obraze všetky farby rovnako.

Konverzia RGB na HSV

Konverzia z RGB modelu do modelu HSV je popísaná v algoritme [4]. Existujú aj konverzie, pri ktorých sa využívajú goniometrické funkcie. Požitie týchto funkcií je vzhľadom na výpočty vykonávanými počítačom nevhodné. Nasledujúce rovnice popisujú konverziu farby z RGB modelu na model HSV, kde rovnica 2.2 popisuje farebný tón, 2.3 sýtosť a 2.3 hodnotu jasu.

$$h = \begin{cases} 0 & \text{ak } \max = \min \\ 60° \times \frac{g-b}{\max - \min} + 0°, & \text{ak } \max = r \text{ a } g \geq b \\ 60° \times \frac{g-b}{\max - \min} + 360°, & \text{ak } \max = r \text{ a } g < b \\ 60° \times \frac{b-r}{\max - \min} + 120°, & \text{ak } \max = g \\ 60° \times \frac{r-g}{\max - \min} + 240°, & \text{ak } \max = b \end{cases} \tag{2.1}$$

$$s = \begin{cases} 0, & \text{ak } \max = 0 \\ 255 * \frac{\max - \min}{\max}, & \text{inak} \end{cases} \tag{2.2}$$

$$v = \max \tag{2.3}$$

Keďže mojim cieľom je, aby fungovala detekcia aj na čiernobielych obrazoch, možnosť detekcie práve na základe farby kože neprichádza v úvahu. Tento spôsob by ale mohol byť vhodný ako predspracovanie obrazu, v ktorom by sa mohli časti obrazu, ktoré by neobsahovali farbu podobnú koži, zmeniť na jednotnú farbu. Takto by sa urýchlilo spracovanie vybranej časti klasifikátormi, keďže by bola daná oblasť monotónna a bolo by jasné, že sa tam tvár nenachádza, prípadne by sa takáto oblasť mohla preskočiť a nemusela by sa vykonávať detekcia vôbec.

Spôsob ako detekovať tvár v čiernobielych obrazoch je využiť detekcie jej jednotlivých rysov. V takomto prípade ide hlavne o nájdenie svetlých a tmavých miest vo vhodnom rozpoložení. Riešením by mohlo byť využitie Haarovych príznakov popísaných v časti 1.2.1 alebo použitie alternatívnych príznakov, ako sú Gáborove vlnky alebo „Local Rank Distance“.

2.1.2 Rýchlosť detekcie

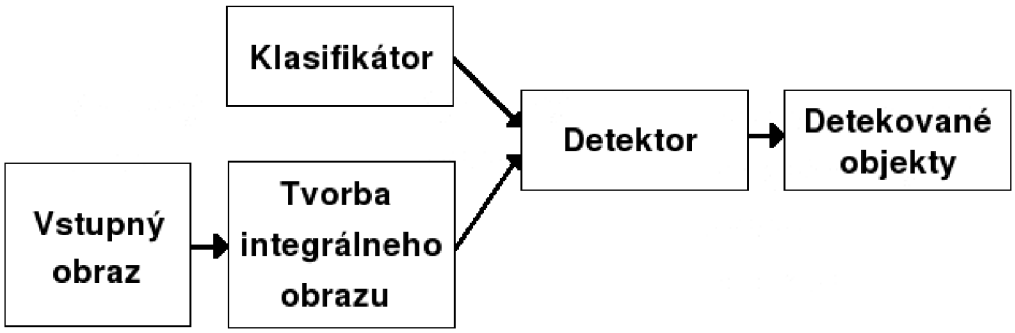
Jedným z hlavných kritérií na úspešnosť systému je rýchlosť. Pre urýchlenie detekcie je možné využiť kaskádové zapojenie viacerých klasifikátorov. V prípade využitia jedného klasifikátora by mal byť vybraný spôsob trénovania taký, ktorý umožnuje veľmi efektívnu klasifikáciu, napríklad neurónovými sieťami alebo alternatívou k AdaBoostu WaldBoostom. Pričom výhodou WaldBoostu oproti neurónovým sieťam je v jednoduchosti trénovania a v tom, že klasifikátory odvodené od algoritmu AdaBoostu nie je možné pretrénovať.

Jednou z možností ako urýchliť detekciu tvári na celom obraze v prípade videa je neprehľadávať zakaždým celý snímok, ale využiť informácie o predchádzajúcom snímku. Spôsob ako realizovať túto možnosť je využitím algoritmu na sledovanie pohybu.

V neposlednom rade je treba spomenúť aj paralelizáciu systému, ktorá je v súčasnosti veľmi výhodná, hlavne keď sa na trhu s výpočtovou technikou objavuje čoraz viacej počítačov s viacjadrovými procesormi, prípadne s počítačmi s viacerými procesormi.

2.2 Návrh riešenia

V tejto časti je popísaný návrh riešenia, ktorý bude implementovaný. Jej časť vychádza z prevedenej analýzy, ktorá je popísaná v predchádzajúcej kapitole. Predstava ako by mal byť výsledný systém realizovaný je zobrazený na 2.1 a bude sa skladať z troch hlavných častí: klasifikátor, obraz a samotný detektor.



Obrázok 2.1: Systém na detekciu tváre

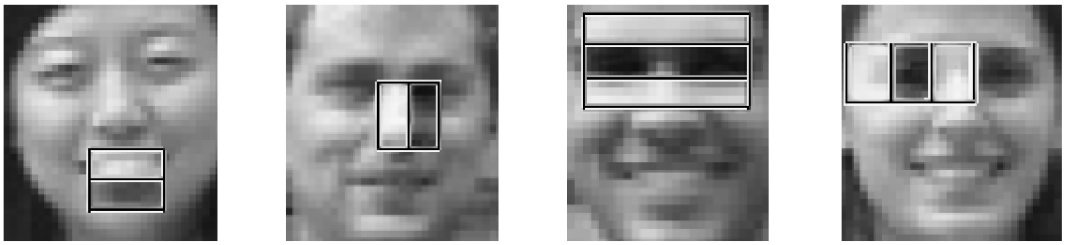
2.2.1 Klasifikátor

Vzhľadom na to, že požadovaný systém by mal pracovať v reálnom čase je vhodné využívať klasifikátory, ktoré boli natrénované pomocou metódy WaldBoost. Táto metóda dokáže ukončiť vyhodnocovanie oblasti predčasne a to i v prípade, že sa nachádza hodnota výsledku pod minimálnou hranicou. Na samotné trénovanie klasifikátorov sa využijú Haarove príznaky. Pri detekcii tváre sa využijú ich 4 modifikácie, ktorých ukážka využitia sa nachádza na obrázku 2.2:

- Horizontálny dvojdielny
- Vertikálny dvojdielny
- Horizontálny trojdielny
- Vertikálny trojdielny

Samotný natrénovaný klasifikátor by mohol byť uložený v XML súbore pre lepšiu možnosť spracovania. Následne na spracovanie XML súboru bude možné využiť niektorého voľne dostupného modulu pre zvolený jazyk.

Na vylepšenie klasifikátora a odstránenie násobných detekcií som sa rozhodol využiť metódy *non-maxima suppresion*.



Obrázok 2.2: Haarove príznaky: (A) Horizontálny dvojdielny (B) Vertikálny dvojdielny (C) Horizontálny trojdielny (D) Vertikálny trojdielny

2.2.2 Detektor

Detektor bude hlavná časť systému. Jeho funkciou bude pomocou poskytnutého klasifikátora vyhľadať na danom obraze oblasti obsahujúce tvár. Intuitívne, asi najrozumnejšia možnosť ako takéto oblasti nájsť je postupne prechádzať klasifikátorom z ľavého horného rohu do pravého dolného rohu.

Druhou funkciou detektoru bude alternatíva k prechádzaniu celého obrazu klasifikátorom, spôsob, pri ktorom sa budú brať do úvahy výsledky predchádzajúcej detekcie. V takomto prípade je najvhodnejšie využiť jednu z metód na sledovanie pohybu. V mojom prípade som sa rozhodol pre možnosť využitia *časticového filtra* s ohľadom na to, že tento spôsob som počas študovania sledovania pohybu objavil a nebol ešte použitý na detekciu tváre s využitím klasifikátorov.

Keďže je výhodné pri detekovaní klasifikátorom, ktorý využíva Haarove príznaky, využiť integrálny obraz, rozhodol som sa implementovať aj tento spôsob urýchlenia.

2.2.3 Obraz

Pre výpočet Haarovych príznakov je dôležité vedieť rýchlo spočítať hodnoty intenzít. Preto je vhodné vytvoriť objekt, ktorý bude mať schopnosť tieto hodnoty poskytnúť. Aby bola výpovedná hodnota príznakov relevantná je potrebné tieto príznaky normalizovať. V mojom prípade bude normalizácia vykonávaná pomocou štandardnej odchýlky. Na samotný výpočet odchýlky sú potrebné hodnoty aj druhých mocnín intenzity. Na základe tejto potreby bude vhodné až nutné implementovať štruktúru nazývanú *integrálny obraz na druhú*.

Integrálny obraz na druhú budem počítať podobne ako sa vypočítava integrálny obraz, ktorý je popísaný v časti zaoberajúcej sa spôsobom detekcie podľa Viola & Jones 1.2.2. Jediná zmena, ktorá bude na výpočet integrálneho obrazu na druhú vykonaná je sumácia intenzít pixelov na druhú, namiesto sumácie intenzít. Výsledná zmena sa premietne vo vzorci 1.8, ktorý sa zmení na:

$$II(x,y) = \sum_{i,j}^{x,y} I^2(i,j) \tag{2.4}$$

2.3 Implementácia

Táto podkapitola je zameraná na samotnú implementáciu systému. Vychádza z analýzy a návrhu riešenia popísaných vyššie.

2.3.1 Implementačný jazyk

Základným rozhodnutím pri implementácii je výber jazyka, v ktorom bude daný program, systém, prípadne modul vytvorený. Vzhľadom na požadovanú rýchlosť systému je lepšie využiť kompilované jazyky na rozdiel od skriptovacích jazykov, ktoré sú náročnejšie na dobu vyhodnocovania. V tomto prípade zostávajú možnosti použiť jazyk *Java* alebo jazyk *C/C++*. Ja osobne mám väčšie skúsenosti s jazykom C/C++ a preto som sa rozhodol práve pre tento jazyk.

2.3.2 XML parser

V návrhu riešenia systému vznikla myšlienka využiť klasifikátory, ktoré boli natrénované algoritmom WaldBoost a výsledok je uložený v XML súbore. Je potrebné vytvoriť modul, ktorý bude obsluhovať načítanie klasifikátora zo súboru alebo použiť voľne dostupné knižnice. V prípade detekcie je možné, aby daná knižnica nedosahovala maximálnu možnú rýchlosť, keďže načítavanie jedného klasifikátora sa bude vykonávať iba raz a aj to sa vykoná ešte pred samotnou detekciou.

Rozhodol som sa využiť možnosť zakomponovania voľne dostupnej knižnice implementujúcej spracovanie XML súborov do svojho programu. Táto knižnica by mala byť jednoduchá a nemala by byť vysoko náročná na pamäť a čas, pretože bude využívaná najmä na čítanie. Vhodnou voľbou sa zdala byť knižnica s názvom XML parser[1].

2.3.3 OpenCV

Na spracovávanie obrazu mi bola vedúcim bakalárskej práce odporučená voľne šíriteľná knižnica *OpenCV*, kde skratka CV pochádza z anglického názvu pre počítačové videnie (*Computer Vision*) a Open znamená voľne šíriteľná. Bola pôvodne vyvíjaná firmou Intel. Knižnica je multiplatformová a dokáže pracovať pod systémom Windows, Linux a aj MacOS. Obsahuje množstvo funkcií zameraných hlavne na spracovanie obrazu. Keďže mojím cieľom bolo vytvoriť systém zameraný na detekciu tváre tak som sa rozhodol využívať danú knižnicu minimálne a používať ju len na načítavanie obrázku alebo snímku z videa a na zobrazovanie výsledkov.

Pri obrázkoch sú podporované rôzne formáty, kde medzi základné patrí: BMP, JPG, JPEG, PNG, TIF, TIFF .... V prípade videa je pracovanie s jednotlivými snímkami rovnaké, pričom sa pri načítaní konkrétneho snímku použije iná funkcia. Video formáty podporované OpenCV sú: AVI, MPG, MPEG, FLV ....

2.3.4 OpenMP

Jednou z možností ako urýchliť činnosť programov je využiť viacej vlákien. V mojom prípade, keď som chcel využívať čo najmenší počet externých knižníc, ktoré by boli potrebné pri kompilácii programu rozhodol som sa využiť knižnicu OpenMP (z ang. Multi-Proccesing). Spomínaná knižnica bola vytvorená spoločnosťou The OpenMP Architecture Review Board pôvodne pre jazyk Fortran. V roku 2002 sa publikoval aj štandard pre jazyk C/C++. Trend prechodu na jazyk C/C++ sa prejavil aj v tom, že verzia prekladača *gcc 4.2* má už v sebe zakomponovanú plnú podporu tejto knižnice a preto nie je potrebné pridávanie externých súborov. Práve tento dôvod bol rozhodnujúci pri výbere spomínanej knižnice.

Používanie knižnice je pomocou direktívy prekladača *pragma*, podľa ktorých prekladač vytvorí viacvláknový program. Formát takejto direktívy má tvar:

```
#pragma omp <príkaz>
```

2.3.5 Popis systému

Táto časť rozoberá jednotlivé podproblémy a samotné vlastnosti, ktoré je nutné dodržiavať pri využívaní systému.

[1] bližšie informácie sú na www stránke `http://iridia.ulb.ac.be/~fvandenb/tools/xmlParser.html`

Obraz

Ako bolo spomenuté v analýze a v návrhu riešenia je vhodné vytvoriť vhodnú reprezentáciu obrazu, v ktorej budú zakomponované štruktúry integrálneho obrazu a integrálneho obrazu na druhú. V prípade, že obraz je vo farebnom prevedení je potrebné vypočítať intenzitu jednotlivých pixelov. Keďže OpenCV využíva RGB model je vhodné využiť rovnice 2.5.

$$I \ = 0.299 * R + 0.587 * G + 0.114 * B \tag{2.5}$$

Klasifikátor

Klasifikátor je načítavaný z XML súboru pomocou modulu popísaného v časti 2.3.2. Klasifikátor musí byť natrénovaný pomocou metódy Waldboost, ktorej výhody sú pomenované v návrhu riešenia. V súčasnosti sú implementované ako slabé klasifikátory Haarove príznaky. XML súbor musí obsahovať uzol pomenovaný *WaldBoostClassifier*. Tento uzol obsahuje jednotlivé slabé klasifikátory reprezentované uzlom *stage*. Slabý klasifikátor k svojej funkcii potrebuje dve hodnoty: hornú a dolnú medzu pre rozhodovanie, ktoré sú atribútmi tohto uzla. Uzol *DecisionTreeWeakHypothesis* reprezentuje rozhodovací strom, ktorý vráti výslednú hodnotu. Uzol je určený na diskretizáciu nameraných hodnôt. Atribút *binMap* reprezentuje hodnoty určujúce poradie predpočítanej hodnoty uložené v atribúte *predictionValues*. Uzol *TCont2DiscFeature* obsahuje len pomocné veličiny pre klasifikáciu ako sú minimálna a maximálna veľkosť predikčnej hodnoty a počet hodnôt v atribúte binMap uzlu DecisionTreeWeakHypothesis. Samotný typ Haarovho príznaku je poduzlom uzlu TCont2DiscFeature. Názov tohto uzlu závisí na type príznaku kde:

- Horizontálny dvojdielny → HaarHorizontalDoubleFeature
- Vertikálny dvojdielny → HaarVerticalDoubleFeature
- Horizontálny trojdielny → HaarHorizontalTernalFeature
- Vertikálny trojdielny → HaarVerticalTernalFeature

Hodnoty atribútov udávajú horizontálnu a vertikálnu pozíciu príznaku v rámci klasifikátora, šírku a výšku jednej časti Haarovho príznaku (bielej alebo čiernej).

Ukážka XML súboru, kde hodnoty *int*, *uint*, *double* reprezentujú typy premenných v jazyku C/C++:

```
<WaldBoostClassifier>
  <stage posT="double" negT="double">
    <DecisionTreeWeakHypothesis binMap="uint uint ...  uint"
predictionValues="double double ...  double">
      <TCont2DiscFeature minValue="double" maxValue="double"
numberOfBins="int">
        <HaarHorizontalTernalFeature positionX="int" positionY="int"
blockWidth="int" blockHeight="int"/>
      </TCont2DiscFeature>
    </DecisionTreeWeakHypothesis>
  </stage>
  ...
</WaldBoostClassifier>
```

Aplikátor

Keďže nie všetky hodnoty v rámci prechádzania klasifikátorom sú potrebné pri danom snímku vypočítavať znova, vznikla trieda, ktorá slúži na prikladanie jednotlivých dielov klasifikátora na obraz. Jeden aplikátor je možné využiť na obrázky s rôznym obsahom, ale podmienkou je, aby mali rovnakú šírku. V opačnom prípade sa automaticky vytvorí iný aplikátor, vhodný na daný obrázok. Zmena veľkosti klasifikačného okna sa mení práve na základe zmeny veľkosti aplikátora, ktorý je implementovaný ako vektor jednotlivých stupňov aplikátora v samotnom klasifikátore.

Záchyt

Záchytom je označované miesto, ktoré bolo detekované ako tvárová oblasť. Úloha záchytov pozostáva v uchovávaní potrebných informácií na určenie presnej polohy a veľkosti tohto miesta. Ďalej obsahuje vlastnosť udávajúca hodnotu istoty (ang. *likelihood*), s akou je daný klasifikátor, ktorý oblasť prehlásil za tvárovú presvedčený o tom, či daná oblasť vyhovuje požiadavkám. Pričom táto hodnota je vypočítaná pomocou rovnice:

$$likelihood = \exp(measure * 2) \tag{2.6}$$

kde *measure* označuje ohodnotenie klasifikátorom. Rovnica vychádza z dokumentu [11], ktorý obsahuje rovnicu[2] 2.7 hľadajúcu váhy slabých klasifikátorov matematicky.

$$\alpha = \frac{1}{2} \ln \left(\frac{W_+}{W_-} \right) \tag{2.7}$$

Súčet jednotlivých váh je potom polovica nášho merania. Inverznou funkciou k tejto rovnici je práve rovnica 2.6.

Častica

Častica je veľmi podobná záchytom. Jej odlišnosť spočíva v tom, že obsahuje premenné indikujúce smer jej pohybu a premennú udávajúcu zmenu veľkosti okna, ktoré častica reprezentuje. Jej využitie spočíva pri spracovávaní podobných snímkov pri videu. Pri detekcii tvárových oblastí sa vytvorí v ich okolí určitý počet častíc. Tie sú rozmiestnené náhodným spôsobom a zmeria sa im váha (využitie vlastnosti likelihood).

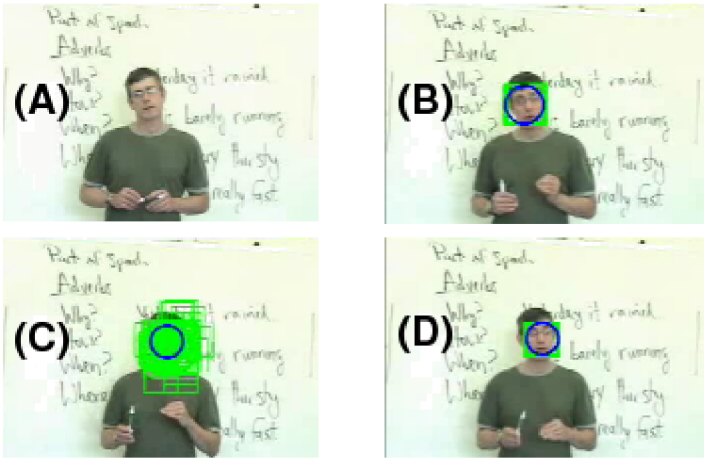
Detektor

Detektor je hlavná časť projektu a jeho hlavnou činnosťou je postupná aplikácia klasifikátora na samotný obrázok. Klasifikátor a aj snímok, ktorý ma byť klasifikátorom vyhodnotený je predávaný pomocou parametrov. Jeho výstupom je zoznam záchytov. Tento zoznam obsahuje iba tie časti obrázku, ktoré sú nad úrovňou hranice zadanej užívateľom.

V prípade detektora pracujúceho s časticami je využitá metóda *Partice filter*. V mojom prípade je využívané dynamického modelu, kde každá častica obsahuje svoju rýchlosť. Princíp je možné popísať počas dvoch spracovávajúcich sa snímkov. Pri prvom snímku sa pomocou celoplošného preskúmavania pohľadajú tvárové oblasti. V tomto kroku môže prebehnúť aj inicializácia častíc, kde sa každej častici náhodne nastaví rýchlosť a zmena veľkosti. Pravdepodobnosť (likelihood) sa nastaví na rovnakú hodnotu. V nasledujúcom snímku sa potom vezmú tieto častice a náhodne sa vyberajú častice. Náhodnosť

[2] v dokumente pod číslom 22

je ale ovplyvnená veľkosťou likelihoodu, pričom sa využíva celková suma likelihoodu častíc a náhodne sa vyberá číslo od 0 po $\sum likelihood$. Dané číslo potom zodpovedá konkrétnej častici, pričom pravdepodobnosť s akou sa častica vyberie zodpovedá veľkosti likelihoodu. Toto je intuitívne správne pretože sa vyberajú častice, ktoré reprezentujú viacej tvárové oblasti. Následne sa vybrané častice náhodným spôsobom pohnú, pričom vychádzajú z predchádzajúceho náhodného posunu, čo opäť dokazuje správnosť výberu na základe veľkosti likelihoodu. Reprezentujú tým smer pohybu objektu a vytvárajú tak model chovania, ktorého tuhosť môže byť ovplynená nastavením parametrov určujúcich ako veľmi sa má brať do úvahy predchádzajúci smer častice.



Obrázok 2.3: Využitie časticového filtra. Modrý krúžok reprezentuje výslednú detekciu, Zelené štvorce reprezentujú častice. Počet častíc je konštantný. (A) Inicializačný snímok, (B) vykreslená inicializácia, (C) vyhľadávanie časticami, (D) výber pri nasledujúcom snímku

Obmedzenia

Počas implementácie vzniklo viacero obmedzení. Jedným sú podporované vstupné formáty. Obmedzenie vychádza z použitej knižnice OpenCV na načítavanie a spracovanie vstupných obrázkov. Napriek tomu, že je počet vstupných formátov dostatočne veľký, vždy sa môžu objaviť formáty súborov, ktoré nie sú podporované a preto je vhodné pri neúspešnom načítaní vstupného snímku vziať do úvahy, že daný typ nemusí byť knižnicou OpenCV podporovaný.

Druhým obmedzením je maximálna veľkosť klasifikátora. Dôvod prečo toto obmedzenie vzniklo je spôsobené výberom metódy zväčšovania klasifikátora namiesto zmenšovania vstupného snímku. Závisí najmä na počítači, na ktorom bola prevedená kompilácia, keďže sa táto hodnota odvíja od spôsobu implementácie *integeru*, ktorý sa využíva pri výpočtoch integrálneho obrazu a integrálneho obrazu na druhú. V prípade, že je integer 32bitový je maximálna veľkosť klasifiátora 255.

$$m_{int} = m_s * m_v * 255^2 \tag{2.8}$$

kde: m_{int} – maximálna hodnota int, m_s – maximálna šírka klasifikátora, m_v – maximálna výška klasifikátora, 255 – maximálna intenzita pixelu

Kapitola 3

Testovanie systému

Cieľom mojej práce bolo vytvorenie systému, ktorý by dokázal v reálnom čase detekovať tváre. V tejto kapitole by som sa chcel zamerať na ohodnotenie toho, či sa mi tento cieľ podarilo splniť. Najväčším problémom bolo aby systém fungoval rýchlo, preto sa zameriam hlavne na testovanie rýchlosti. Ďalším mojim kritériom bude úspešnosť klasifikátorov v hľadaní. Rýchly klasifikátor, ktorý má vysokú chybovosť je nepoužiteľný. Pri hodnotení správnosti sa používajú dva pojmy:

- false positive — skrátene fp, označuje zle označené pozitívne príklady (chybne označené oblasti za tvárové)
- false negative — skrátene fn, označuje zle označené negatívne príklady (tváre, ktoré neboli nájdené)

3.1 Výsledky a hodnotenie

Pri testovaní som použil 4 klasifikátory, ktoré mi boli poskytnuté vedúcim bakalárskej práce. Každý z týchto klasifikátorov pozostával z 1000 slabých klasifikátorov vytvorených na základe Haarovych príznakov. Testovanie prebiehalo na PC, ktorého procesor bol Intel Celeron 1.6GHz. Základný klasifikátor použitý na testovanie bol *r-WB-Haar-a02.xml*. Na porovnanie s ostatnými klasifikátormi bol použitý referenčný obrázok 3.1, ktorý obsahoval 13 tvári. Jeho veľkosť bola 800x600, 700x525, 600x450 a 500x375.

Obrázok 3.1: Referenčný obrázok na porovnávanie klasifikátorov

Základná veľkosť okna klasifikátora bola 24px. Pri prehľadávaní bola potom táto veľkosť zväčšovaná v pomere 1:1,2 . Výsledky všetkých klasifikátorov sú zobrazené v prílohe A. Nameraný čas udáva počet sekúnd, ktoré boli potrebné na prehľadanie obrázku detektorom. Nezahrnuje v sebe čas potrebný na načítanie klasifikátora a ani čas na vykreslenie detekcií.

V tejto časti je popísané testovanie klasifikátorov na zistenie počtu prehľadávaných okien v závislosti na nastaveniach detektora, kde posun vo vodorovnom a zvislom smere sa vypočíta:

$$dx = m_x + sizeX/s_x \qquad dy = m_y + sizeY/s_y$$

kde: dx/dy — je posun vo vodorovnom/zvislom smere
m_x/m_y — minimálny posun v horizontálnom/vertikálnom smere
$sizeX/sizeY$ — aktuálna šírka/výška okna klasifikátora
s_x/s_y — hodnota určujúca zmenu v závislosti na šírke/výške okna ak sa rovná 0 je d_x/d_y rovné min_x/min_y

Detekovanie prebiehalo na 10 obrázkoch o veľkosti 600x450, na ktorých sa nachádzalo spolu 33 tvári. Pričom každá fotografia obsahovala od 1 do 9 osôb. Výsledky sú zobrazené v tabuľke 3.1, kde *počet okien* udáva hodnotu koľkokrát bol klasifikátor priložený na daný obrázok, $\bar{p}_{st}$ udáva priemerný počet slabých klasifikátorov, po ktorých dokázal natrénovaný silný klasifikátor rozhodnúť o danej oblasti a st_{last} udáva koľko bolo takých oblastí, ktoré boli ohodnotené ako tvárové, tj. dosiahli toho, že prešli všetkými stupňami klasifikátora.

parametre				počet	a02.xml		a05.xml		a1.xml		a2.xml	
m_x	m_y	s_x	s_y	okien	$\bar{p}_{st}$	st_{last}	$\bar{p}_{st}$	st_{last}	$\bar{p}_{st}$	st_{last}	$\bar{p}_{st}$	st_{last}
1	1	10	10	1 107 980	4,69	445	2,73	426	1,74	209	1,36	169
1	1	20	20	3 318 070	4,81	1 444	2,81	1 345	1,78	705	1,36	513
1	1	0	0	26 969 540	5,47	14 908	3,25	13 412	2,02	7 367	1,47	5 099
2	2	0	0	6 753 480	5,46	3 693	3,25	3 358	2,02	1 830	1,48	1 311
5	5	0	0	1 090 920	5,45	577	3,25	534	2,03	305	1,47	208

Tabuľka 3.1: Výsledky klasifikátorov r-WB-Haar-a02.xml, t-t-WB-Haar-a05.xml, t-t-WB-Haar-a1.xml a t-t-WB-Haar-a2.xml

Testovanie na video sekvenciach pri využívaní Particle filter nebolo uskutočnené, keďže už od pohľadu bolo zrejmé, že daný výsledok nedosahuje požadovanú kvalitu. Pričom existovali dve varianty. Prvou bolo využiť malý počet (približne 100) častíc. V tomto prípade ale nebolo možné nastaviť parametre tak, aby dokázali danú tvár sledovať a už po niekoľkých snímkoch sa častice od tvári vzďaľovali. Druhou variantou, ktorá pripadala do úvahy bolo použiť väčšieho počtu (približne 1000) častíc. Nevýhodou ale bolo, že pri vyššom počte častíc narastal aj čas, ktorý potreboval detektor na výpočet novej detekcie, keďže musel vyhodnotiť väčší počet častíc. Narozdiel od preskúmavania celého obrazu, kde je priemerná doba vyhodnocovania do 6 slabých klasifikátorov, sa častice zdržiavajú v tvárovej oblasti a doba na vyhodnotenie je oveľa väčšia. V prípade, že sa všetky vyberané častice umiestnia v okolí tváre môže každá častica prejsť cez všetkých 1 000 stupňov. Ak sa tisíc častíc nachádza v oblasti tváre a prejde cez 1 000 slabých klasifikátorov dostávame 1 000 000 meraní na nájdenie jednej tváre, čo je z časového hľadiska neprijateľné.

Záver

Práca sa zaoberala metódami vyhľadávania tváre v obraze so zameraním hlavne na metódy využívajúce klasifikátory. V prvej časti boli rozobrané súčasné spôsoby trénovania klasifikátorov.

Ďalším cieľom bolo vytvorenie systému na detekovanie tvárí, ktorý by bol schopný pracovať v reálnom čase. Z testovania vyplynulo, že túto úlohu sa mi nepodarilo celkom splniť, keďže som si dal za cieľ aby systém bol schopný pracovať s videom. Jediný použiteľný spôsob ako pracovať s videom je prechádzanie zakaždým celých snímkov, lebo implementovaný postup, ktorý mal využívať informácie z predchádzajúcich obrazov sa ukázal ako nevhodný.

Do budúcnosti je možné vylepšiť systém na monitorovanie pohybu tváre. Napríklad použitím adaptabilného algoritmu časticového filtru, ktorý by na základe istoty nájdenia tváre určoval počet častíc. Ďalej by bolo vhodné vytvoriť aj grafické užívateľské rozhranie.

Literatúra

[1] BABJAK, J.: Neurónové siete sú čiernou skrinkou. [online; cit. 2008-04-18; rev. 2003-05-07].
URL `<http://www.root.cz/clanky/neuronove-siete-su-ciernou-skrinkou/>`

[2] BESZÉDEŠ, M.: *Detekcia tvárí a tvárových čŕt v obraze*. Dizertační práce, Slovenská technická univerzita v Bratislave, Fakulta elektrotechnity a informatiky, Katedra telekomunikácií, 2007, ved. Miloš Oravec.

[3] BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N.: A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, New York, NY, USA: ACM, 1992, ISBN 0-89791-497-X, s. 144–152, doi:http://doi.acm.org/10.1145/130385.130401.

[4] CARDANI, D.: Adventures in HSV Space. júl 2001, [Online; cit. 2008-04-24].
URL `<robotica.itam.mx/espanol/archivos/hsvspace.pdf>`

[5] FREUND, Y.; SCHAPIRE, R. E.: Experiments with a new boosting algorithm. In *international conference on machine learning*, 1996, s. 148–156.

[6] FREUND, Y.; SCHAPIRE, R. E.: A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, ročník 14, č. 5, september 1999: s. 771–780.

[7] HUE, C.; CADRE, J.-P. L.; PEREZ, P.: Tracking multiple objects with particle filtering. *Aerospace and Electronic Systems, IEEE Transactions on*, ročník 38, č. 3, júl 2002: s. 791–812, ISSN 0018-9251, doi:10.1109/TAES.2002.1039400.

[8] ISARD, M. A.: *Visual Motion Analysis by Probabilistic Propagation of Conditional Density*. Dizertační práce, Oxford University, 1998.

[9] ROOHI, M.; MIRJALILY, G.; SADEGHI, M. T.: Face Detection Using a Modified SVM-Based Classifier. In *ICCIMA '07: Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, Washington, DC, USA: IEEE Computer Society, 2007, ISBN 0-7695-3050-8, s. 354–360, doi:http://dx.doi.org/10.1109/ICCIMA.2007.191.

[10] ROWLEY, H.; BALUJA, S.; KANADE, T.: Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 20, č. 1, Január 1998: s. 23–38.

[11] SCHAPIRE, R. E.; SINGER, Y.: Improved Boosting Using Confidence-rated Predictions. *Machine Learning*, ročník 37, č. 3, 1999: s. 297–336.

[12] SCHNEIDERMAN, H.; KANADE, T.: A Statistical Model for 3D Object Detection Applied to Faces and Cars. In *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jún 2000.
URL <http://www.ri.cmu.edu/pub_files/pub2/schneiderman_henry_2000_1/schneiderman_henry_2000_1.pdf>

[13] VIOLA, P.; JONES, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. *cvpr*, ročník 1, 2001: str. 511, ISSN 1063-6919, doi:http://doi.ieeecomputersociety.org/10.1109/CVPR.2001.990517.

[14] Wikipedia: Artificial neural network — Wikipedia, The Free Encyclopedia. 2008, [Online; cit. 2008-04-21].
URL <http://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=206085996>

[15] Wikipedia: Machine learning — Wikipedia, The Free Encyclopedia. 2008, [Online; cit. 2008-04-21].
URL <http://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=205836800>

[16] Wikipédia: Neurónová sieť — Wikipédia, Slobodná encyklopédia. 2007, [Online; cit. 2008-04-21].
URL <http://sk.wikipedia.org/w/index.php?title=Neur%C3%B3nov%C3%A1_sie%C5%A5&oldid=860783>

[17] YAN, Y.; GUO, Z.; YANG, J.: Multi-view Face Detection Based on the Enhanced AdaBoost Using Walsh Features. IEEE, júl 2007, ISBN 978-0-7695-2909-7, s. 200–205.

[18] ŠOCHMAN, J.; MATAS, J.: WaldBoost - Learning for Time Constrained Sequential Detection. IEEE, jún 2005, ISBN 0-7695-2372-2, s. 150–156.

Dodatok A

Test klasifikátorov

dx – posun klasifikátora v x-ovom smere, dy – posun klasifikátora v y-ovom smere, sX – aktuálna šírka klasifikátora, sY – aktuálna výška klasifikátora, t – čas, fp – false positve, fn – false negative

r-WB-Haar-a02.xml															threshold = 1
parametre	dx=1+sX/10 dy=1+sY/10			dx=1+sX/20 dy=1+sY/20			dx=1 dy=1			dx=2 dy=2			dx=5 dy=5		
veľkosť	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn
800x600	0.41	0	5	1.17	2	3	9.8	8	2	2.56	2	2	0.44	0	4
700x525	0.24	0	2	0.69	1	3	5.99	5	1	1.55	1	3	0.27	1	3
600x450	0.13	0	3	0.38	0	1	3.19	1	1	0.82	0	1	0.15	0	3
500x375	0.09	0	5	0.24	0	5	1.93	1	3	0.49	0	5	0.09	0	7
t-t-WB-Haar-a05.xml															threshold = 1
parametre	dx=1+sX/10 dy=1+sY/10			dx=1+sX/20 dy=1+sY/20			dx=1 dy=1			dx=2 dy=2			dx=5 dy=5		
veľkosť	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn
800x600	0.3	0	2	0.87	3	2	7.04	11	2	1.84	4	2	0.33	2	4
700x525	0.16	0	1	0.47	1	1	3.96	4	1	1.04	2	1	0.18	0	3
600x450	0.08	0	3	0.22	0	0	1.89	1	1	0.49	0	0	0.09	0	3
500x375	0.05	0	6	0.14	0	4	1.16	1	4	0.3	0	5	0.06	0	10
t-t-WB-Haar-a1.xml															threshold = 1
parametre	dx=1+sX/10 dy=1+sY/10			dx=1+sX/20 dy=1+sY/20			dx=1 dy=1			dx=2 dy=2			dx=5 dy=5		
veľkosť	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn
800x600	0.16	0	2	0.46	1	2	3.88	4	2	1.01	2	2	0.19	0	3
700x525	0.1	0	0	0.28	1	0	2.31	1	1	0.61	1	0	0.12	0	3
600x450	0.05	0	2	0.15	1	2	1.16	1	2	0.3	1	2	0.06	0	5
500x375	0.04	0	7	0.09	0	6	0.7	0	5	0.18	0	6	0.03	0	12

t-t-WB-Haar-a2.xml															threshold = 1
parametre	dx=1+sX/10 dy=1+sY/10			dx=1+sX/20 dy=1+sY/20			dx=1 dy=1			dx=2 dy=2			dx=5 dy=5		
veľkosť	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn	t	fp	fn
800x600	0.11	0	3	0.32	0	3	2.58	4	2	0.68	1	3	0.13	0	4
700x525	0.07	0	0	0.19	0	1	1.59	0	0	0.42	0	1	0.08	0	4
600x450	0.04	0	4	0.11	1	1	0.85	1	1	0.22	1	1	0.03	0	9
500x375	0.03	0	8	0.07	0	7	0.51	0	4	0.13	0	7	0.03	0	12