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

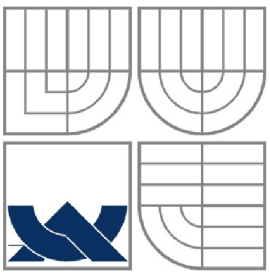
DETEKCE OBLIČEJE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

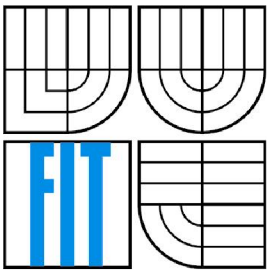
AUTOR PRÁCE
AUTHOR

Bc. ONDŘEJ ŠAŠINKA

BRNO 2009



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

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ONDŘEJ ŠAŠINKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN JURÁNEK

BRNO 2009

Abstrakt

Tato diplomová práce se zabývá detekcí obličeje v obraze. Bude zde uveden přístup, který nejprve detekuje jednotlivé obličejové rysy (oči, nos, koutky úst), které pak podle určitých pravidel spojuje do výsledné detekce obličeje. Pro detekci obličejových rysů jsou použity klasifikátory natrénované algoritmem AdaBoost. Jako příznaky pro klasifikaci jsou použity Haarovy vlnky.

Klíčová slova

Detekce obličeje, detekce obličejových rysů, AdaBoost, Haarovy vlnky, integrální obraz.

Abstract

This diploma thesis deals with face detection in image. In this approach, facial features (eyes, nose, mouth corners) are detected first and then joined to the whole face. For the facial features detection, classifiers trained with AdaBoost algorithm are used. Haar wavelets are used as features for classification.

Keywords

Face detection, facial features detection, AdaBoost, Haar wavelets, integral image.

Citace

Šašinka Ondřej: Detekce obličeje. Brno, 2009, diplomová práce, FIT VUT v Brně.

Detekce obličeje

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Romana Juránka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Na tomto místě bych rád poděkoval Ing. Romanu Juránkovi za odborné vedení, užitečné rady a konzultace, které mi byly velkým přínosem při tvorbě diplomové práce.

Obsah

1 Úvod

Digitální zpracování obrazu v posledních letech, hlavně díky obrovskému vzrůstu výkonnosti počítačů, zažívá velký rozvoj. Přirozeně se proto také velmi rozšířilo jeho využití v nejrůznějších oblastech jako je doprava, průmysl, medicína atd. Zpracování obrazu se zabývá mnoha různými úlohami a jednou z nich je i detekce objektů v obraze. Obličej je v tomto smyslu také objekt, který může být pomocí postupů zpracování obrazu detekován či lokalizován.

Detekce obličejových rysů, potažmo obličeje je základem každého systému, který jakýmkoliv způsobem pracuje s lidským obličejem. Správná lokalizace obličeje je tedy pro tyto systémy naprosto esenciální úlohou, bez které by systémy nemohly fungovat.

Tato diplomová práce se zabývá detekcí obličeje v obraze. Částečně navazuje na moji bakalářskou práci, která se věnovala detekci obličejových rysů.

Bude zde uveden přístup, který nejprve detekuje jednotlivé obličejové rysy (oči, nos, koutky úst), které pak podle určitých pravidel spojuje do výsledné detekce obličeje. Pro detekci obličejových rysů jsou použity klasifikátory natrénované algoritmem AdaBoost. Jako příznaky pro klasifikaci jsou použity Haarovy vlnky.

Práce je členěna celkem do devíti kapitol. Úvodní druhá kapitola stručně představuje problematiku zpracování obrazu. Jsou zde vysvětleny pojmy jako obraz či filtrace. Třetí kapitola se zabývá rozpoznáváním vzorů. Vedle obecného úvodu do problematiky zde budou uvedeny některé druhy příznaků, které se při detekci objektů v obraze používají. Také popíše vybrané metody klasifikace, jako jsou umělé neuronové sítě a support vector machines. Detailněji bude popsána metoda AdaBoost, která je v této práci použita, proto je jí věnována čtvrtá kapitola. V páté kapitole je představena problematika detekce obličeje spolu s přehledem různých přístupů. Návrhu metody, která je předmětem diplomové práce, je věnována kapitola šestá. Jsou zde podrobně vysvětleny jednotlivé kroky samotné detekce. Sedmá kapitola probírá detaily implementace navržené metody a osmá prezentuje výsledky, které detekce touto metodou dosahuje. Závěrečná devátá kapitola shrne a zhodnotí celou práci.

2 Základy zpracování obrazu

Zpracování obrazu je obor, který se zabývá analýzou obsahu obrazu. Je to jakýsi protipól *počítačové grafiky*, jejíž úkolem je naopak syntéza obrazu. Typickými úlohami ve zpracování obrazu jsou např. geometrické transformace, úpravy barev, filtrace, detekce hran, atd.

Na zpracování obrazu navazuje tzv. *počítačové vidění*. Jak již sám název napovídá, jedná se v určitém smyslu o napodobení lidského vidění počítačem. Typické úlohy řešené v rámci počítačového vidění jsou rekonstrukce objektů, rozpoznávání sémantiky v obraze apod.

Zpracování obrazu lze chápat jako zpracování dvourozměrného signálu, kdy vstupem je obraz (fotografie, snímek z videosekvence) a výstupem může být opět obraz, nebo množina parametrů (příznaků), vztahujících se k obsahu obrazu.

Zpracování obrazu a následující rozpoznání jeho obsahu je možné rozdělit do několika po sobě následujících kroků

- *Snímání, digitalizace a uložení obrazu v počítači*
- *Předzpracování obrazu*
- *Segmentace a popis objektů*
- *Porozumění obsahu obrazu, případně klasifikace objektů*

Prvním krokem rozumíme pořízení obrazu a jeho uložení do počítače. Snímání převádí vstupní optické veličiny na elektrický signál spojitý v čase i úrovni. Vstupní informací rozumíme jas, nebo několik spektrálních složek (červená., zelená, modrá) při barevném snímání. Digitalizace představuje diskretizaci vstupního signálu. Na vstupní analogový signál nahlížíme jako na funkci *f(x,y)* dvou proměnných, ty odpovídají souřadnicím v obraze. Funkční hodnota je pak např. jas, případně trojsložkový vektor RGB. Úkolem digitalizace je pak tento signál vzorkovat a kvantovat. Výstupem je tedy matice čísel popisujících obraz. Prvku takovéto matice se pak říká obrazový element – *pixel* (picture element).

Předzpracování obrazu jako druhý krok znamená potlačení šumu a zkreslení, které vzniká při digitalizaci a přenosu obrazu. Znamená to např. ekvalizace histogramu a další. V jiných případech má tento krok za úkol zvýraznit určité požadované rysy obrazu, které jsou pak důležité pro další zpracování. Toto může představovat např. detekce hran.

Segmentace, třetí krok zpracování obrazu umožňuje najít v obraze hledané objekty. Hledanými objekty rozumíme ty části obrazu, které nás při dalším zpracování zajímají.

Poslední fáze – porozumění obsahu obrazu – má spíše blíže k oboru počítačové vidění. Klasifikaci objektů bude věnována jedna z následujících kapitol, proto zde jen uvedu, že pod pojmem klasifikace je myšleno přiřazení objektu jedné z definovaných tříd.

Obraz

Ve zpracování obrazu je obraz chápán v intuitivním smyslu, tzn. jako obraz na sítnici lidského oka nebo obraz sejmutý TV kamerou. Obraz může být modelován matematicky pomocí spojité skalární funkce f dvou nebo tří proměnných, která se nazývá *obrazová funkce*. Statický obraz je popsán obrazovou funkcí dvou souřadnic $f(x, y)$ v rovině. Obrazová funkce tří proměnných se použije, buď když se plošné obrazy mění v čase t, tj. $f(x, y, t)$ nebo v případě objemových obrazů $f(x, y, z)$, např. u tomografu.

Hodnoty obrazové funkce odpovídají některé měřené fyzikální veličině, např. jasu u obrazu z černobílé TV kamery, teplotě u termovizní kamery nebo schopnosti pohlcovat záření v daném místě objemu u rentgenového tomografu atd.

Obrazové funkce dělíme na *spojité* a *diskrétní*. Spojité funkce mají definiční obor i obor hodnot spojitý. Analogicky pro funkce diskrétní platí, že obory jsou množiny diskrétních bodů.

Nejčastěji ve zpracování obrazu pracujeme s tzv. *monochromatickým obrazem*, který je reprezentován pouze jednou obrazovou funkcí. Jinak řečeno funkční hodnoty obrazové funkce jsou skalární veličiny. V jiném případě hovoříme o obrazu *multispektrálním*. Při této variantě každé dvojici souřadnic připadne vektor hodnot. Složky vektoru odpovídají hodnotám jasů jednotlivých barevných složek obrazu. Nejčastěji jsou těmito barevnými složkami červená, zelená a modrá – RGB složky. Barevný obraz se při zpracování obrazu používá pouze tehdy, pokud má pro nás tato informace význam, tzn. je pro náš postup potřebná. Jinak se pracuje pouze s obrazem monochromatickým. Z barevného obrazu se monochromatický získá jednoduchým vztahem:

$$I = 0{,}3\,R + 0{,}59\,G + 0{,}11\,B. \tag{1}$$

Je vidět, že jednotlivé složky jsou do výsledné intenzity zahrnuty s jinými koeficienty. Toto odpovídá faktu, že lidské oko vnímá intenzitu barevných složek různě. Nejvíce je citlivé na zelenou barvu, jak plyne z uvedeného vztahu. Je nutné dodat, že tento vztah byl vytvořen empiricky.

Jelikož počítač není schopen pracovat se spojitou obrazovou informací, je nutné provést diskretizaci. Diskretizace se dosáhne pomocí *vzorkování* a *kvantování*. Vzorkování znamená diskretizaci obrazu v prostoru do matice bodů. Kvantování zase použití určitého omezeného počtu úrovní jasu.

Filtrace

Pod pojmem filtrace se obecne rozumí proces, při němž se určitá část daného systému propouští a jiná zadržuje či zeslabuje.

Filtrace ve zpracování obrazu se provádí dvojím způsobem: *filtrace v prostorové oblasti* a *filtrace ve frekvenční oblasti.*

Filtrace v prostorové oblasti

Při filtraci v prostorové oblasti se používá tzv. *konvoluce*. Konvoluce je jednou z nejzákladnějších operací ve zpracování obrazu. Nás zajímá hlavně konvoluce dvojrozměrných funkcí, která je definována následujícím vztahem:

$$f(x,y)*h(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a,y-b)h(a,b)\,dadb. \qquad (2)$$

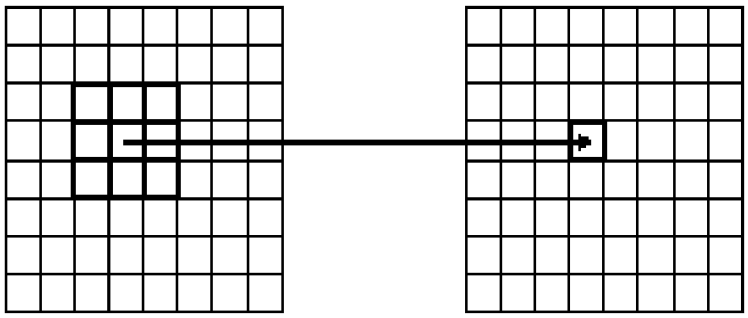
Funkce $h(x,y)$ se nazývá *konvolučním jádrem*. Protože digitální obraz má disktrétní charakter, používá se následující varianta konvoluce:

$$I(x,y)*h(x,y)=\sum_{i=-k}^{k}\sum_{j=-k}^{k} I(x-i,y-j)h(i,j). \qquad (3)$$

I(x,y) je diskrétní obraz a h(x,y) jádro konvoluce.

Zpravidla se využívá pravoúhlé okolí. Aby toto okolí bylo symetrické vůči středovému elementu, je velikost volena z množiny lichých přirozených čísel. Nejčastěji to bývá okolí 3×3, či 5×5. Filtry jsou typicky 2D FIR, tedy filtry s konečnou impulzní odezvou.

Ilustrace výpočtu konvolučního filtru je uvedena na následujícím obrázku:



Obrázek 1 - Grafické znázornění výpočtu konvolučního filtru

Filtrace ve frekvenční oblasti

Filtrace ve frekvenční oblasti je založena na *Fouriérově teorému*, který říká, že každá funkce *f(x)*, která splňuje tzv. *Dirichletovy podmínky*, může být rozložena na součet harmonických funkcí. Podle druhu signálu rozlišujeme Fourierovu transformaci spojité funkce (*FT*) a diskrétní Fouriérovu transformaci (DFT). V případě obrazu se tedy jedná o DFT.

Filtrace ve frekvenční oblasti se skládá ze tří částí:

- Fouriérova transformace – převedení do frekvenční oblasti
- Násobení filtrem – při filtraci ve frekvenční oblasti se nepoužívá konvoluce, ale násobení jednotlivých koeficientů
- Inverzní Fouriérova transformace – převedení zpět do prostorové oblasti

Toto lze formálně zapsat jako:

$$f(x) = IFFT(FFT(x) \cdot F), \tag{4}$$

kde F je frekvenční charakteristika filtru f a "$\cdot$" je násobení jednotlivých koeficientů.

Filtraci je možné možné rozdělit na *lineární* a *nelineární*. Linearita takovéto operace znamená splnění principu *superpozice* nad vstupy a výstupy filtrů, což v jednoduchém zápisu lze vyjádřit jako:

$$f(x_1 + x_2) = f(x_1) + f(x_2). \tag{5}$$

Lineární konvoluční filtry jsou jedním z nejzákladnějších prostředků pro zpracování obrazu. Výpočetní nároky jsou většinou nevelké a lze jimi provést frekvenční filtraci v širokém rozsahu. Jsou také používány v nelineárních filtrech, které jejich výstupy kombinují ve výsledný nelineární filtr.

Mezi operace využívající filtraci patří v*yhlazování obrazu*, jehož cílem je potlačení vyšších frekvencí obrazové funkce. Toto znamená odstranění náhodného šumu, současně však dochází k potlačení ostatních náhlých změn obrazové funkce, což se projeví jako rozmazání hran.

Nejjednodušším způsobem vyhlazování je obyčejné *průměrování*. Výstupem filtrace je hodnota, která je průměrem vstupních hodnot z určitého zvoleného okolí. Často se také zvyšuje váha středového bodu (či jeho sousedů) pro lepší aproximaci šumu s Gaussovským rozložením. Takovéto filtry pak vypadají například následovně:

$$h_{16} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, h_{10} = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Na následujícím obrázku je vidět výsledek použití filtru h_{16}. Z obrázku v pravo (po filtraci) jsou patrné výše popsané vlastnosti vyhlazování obrazu, tedy částečné potlačení šumu nutně spojené s výrazným rozmazáním hran.

Obrázek 2 - Výsledek vyhlazení obrazu

Dalším příkladem konvolučního filtrů je filtr pro „zostření" obrazu, který oproti předchozímu „vyhlazovacímu" filtru zvýrazní vyšší frekvence. Jedná se tedy naopak o horní propust. Konvoluční maska spolu s výsledkem filtrace může vypadat následovně:

$$h_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

Obrázek 3 - Výsledek zaostření obrazu

Detekce hran patří k dalším základním operacím zpracování obrazu. Hrana představuje body v obraze, ve kterých dochází ke změnám obrazové funkce. Čím větší je změna, tím je hrana výraznější. Tomuto odpovídají tzv. o*brazové diference*, které jsou diskrétní variantou derivací. Představují tedy výpočet rozdílů mezi pixely. Toto lze vyjádřit následujícím vztahem:

$$\begin{aligned} I_x(i,j) &= \sum_k w(k)(I(i-k,j) - I(i+k,j)), \\ I_y(i,j) &= \sum_k w(k)(I(i,j-k) - I(i,j+k)), \end{aligned} \tag{6}$$

kde $k \in 1,\ldots,n$ a w je případná váhová funkce.

Lineární filtry však nejsou schopné detekovat hranu vedoucí „libovolným směrem“, ale pouze hrany určitého druhu, např. jen horizontální, či vertikální. Proto je nutné aplikovat nelinearitu na výsledky lineárních konvolučních filtrů, která umožní provést detekci ve více směrech zároveň. Následující konvoluční masky představují detekci horizontálních (h_h) a vertikálních hran (h_v):

$$h_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, h_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

Kombinace lineárních filtrů je možné provést mnoha způsoby. Nejběžnější jsou následující:

$$\begin{aligned} f &= max(f_1, f_2, \ldots, f_n), \\ f' &= |f_1| + |f_2| + \ldots + |f_n|, \\ f'' &= \sqrt{f_1^2 + f_2^2 + \ldots + f_n^2}, \end{aligned} \tag{7}$$

kde $f_1, f_2, \ldots, f_n$ jsou dílčí lineární filtry. Příkladem této třídy filtrů je *Sobelův filtr* pro zvýraznění hran. Je definován jako kombinace dvou lineárních filtrů. Tyto filtry zvýrazňují hrany ve dvou na sebe kolmých směrech, a to ve směru horizontálním a vertikálním.

Následující čtyři obrázky představují jednotlivé kroky při použití Sobelova filtru. Obrázek vpravo nahoře je obrázkem výchozím. Pravý horní představuje aplikaci výše uvedeného filtru h_h, levý dolní pak filtru h_v. A konečně levý dolní obrázek znázorňuje výslednou kombinaci těchto dvou lineárních filtrů pomocí rovnice $S = \sqrt{f_1^2 + f_2^2}$

Obrázek 4 - Ukázka detekce hran

Výše zmíněné operace tedy slouží zejména k předzpracování obrazu tak, aby se odstranily některé nedostatky, jako je šum. Co se týče detekce obličeje, je např. možné použít tzv. Cannyho hranový detektor pro určení oblastí, které obsahují příliš málo, či naopak příliš mnoho hran a tyto oblasti neprohledávat, neboť obličej neobsahují. Toto může urychlit proces detekce.

3 Rozpoznávání vzorů

Tato kapitola si klade za cíl stručně popsat problematiku rozpoznávání vzorů. V úvodu bude popsán princip klasifikace, dále pak stručně uvede některé klasifikační algoritmy. Také se zmíním o příznacích, se kterými je možné se setkat při rozpoznávání vzorů v obraze.

Rozpoznávání vzorů (pattern recognition) je součástí oboru *strojové učení*, jehož hlavním cílem je návrh a vývoj algoritmů a technik, které umožňují počítači „učit se“, tzn. dělat správná rozhodnutí u předem neznámých situací. Rozpoznávání vzorů je jednoduše řečeno **zařazení** objektů nebo událostí do **předem známých kategorií**.

Je důležité vymezit základní pojmy, jako jsou:

- **vzor** – objekt, proces, či událost
- **třída (kategorie) –** množina vzorů se stejnými vlastnostmi
- **příznak (příznakový vektor) –** výsledek měření a pozorování, podle nějž je zařazení (klasifikace) prováděna

Aplikačních oblastí rozpoznávání vzorů je velmi mnoho. Některé příklady uvádí následující seznam:

- Rozpoznávání řeči
- Identifikace mluvčího
- Rozpoznávání jazyka
- Kategorizace textů – zařazení textů do jedné z kategorií (např. jedá-li se o spam, či ne)
- Rozpoznávání písma – OCR
- Rozpoznávání lidské tváře
- Otisky prstů
- a další

Při rozpoznávání vzorů je možno použít různé přístupy, které se dají rozdělit do následujících kategorií:

- ***Strukturní rozpoznávání*** – třídy vzorů jsou popsány formálně, a to např. Gramatikou, či automatem
- ***Statistické rozpoznávání*** – založeno na statistických modelech vzorů a tříd, objekt je popsán n-ticí čísel, která reprezentuje jeho vlastnosti

V následujících odstavcích se trochu blížeji zaměřím na statistické rozpoznávání.

Jak již bylo zmíněno, důležitým pojmem je příznakový vektor, na jehož základě samotné přiřazení do třídy probíhá.

Označíme-li příznakový vektor $\boldsymbol{x} \in X$, kde X je příznakový prostor (feature space) a Y označíme jako množinu tříd, pak *klasifikaci* do tříd můžeme vyjářit jako zobrazení q:

$$q : X \to Y. \tag{8}$$

Jedná se tedy vlastně o nalezení separační linie (hyperplochy v případě vícerozměrného prostoru příznaků), která bude schopna oddělit vzory do tříd.

Nalezení takovéto funkce je úkolem tzv. *trénování*, kdy na základě trénovacích příznaků odhadujeme funkci *q*. Existují základní dva druhy trénování, a to:

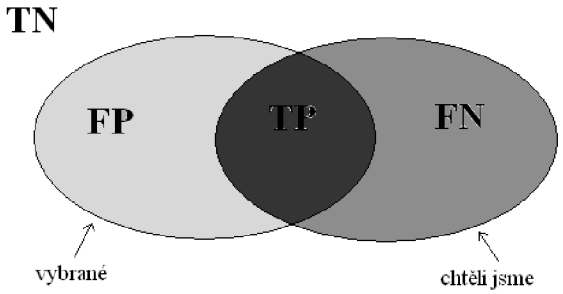
- ***učení s učitelem*** – známe příslušnost trénovacích vektorů k jednotlivým třídám
- ***učení bez učitele*** – máme k dispozici jen trénovací data, nikolik však jejich zařazení do tříd

V případě postupu probíraném v této práci se jedná o učení s učitelem. Máme tedy informaci, která trénovací data patří k jaké třídě.

Důležitým pojmem je tzv. *přetrénování (overfitting)*, kdy klasifikátor ztrácí schopnost generalizovat vlivem přílišného zaměření klasifikátoru na rozeznávání pouze konkrétních trénovacích dat. Výsledkem tak bývá špatná klasifikace. Řešením je tzv. *cross-validace*, kdy se část dat použije pro trénování a část na testování.

Pro klasifikaci je také nutné vyjádřit její chybu. Tato chyba se určuje na testovací sadě dat, u kterých je známa jejich příslušnost ke třídě. Zavadějí se čtyři hodnoty: True Positive (TP, správné přijetí), True negative (TN, správné odmítnutí), False Positive (FP, chybné přjetí) a False Negative (FN, chybné odmítnutí). Výynam hodnot ilustruje obrázek 5. Výpočet těchto hodnot vyjadruje vztah:

$$\begin{aligned} TP &= \sum_i H(x_i) = y_i, \quad y_i = +1 \\ TN &= \sum_i H(x_i) = y_i, \quad y_i = -1 \\ FP &= \sum_i H(x_i) \neq y_i, \quad y_i = +1 \\ FN &= \sum_i H(x_i) \neq y_i, \quad y_i = -1 \end{aligned} \tag{9}$$



Obrázek 5 - Veličiny při hodnocení klasifikátorů

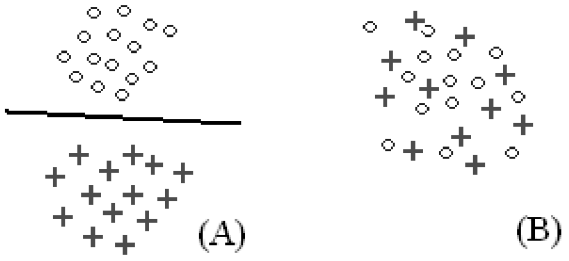
H(x) je klasifikační funkce, x_i vzor a y_i jeho ohodnocení. Chyba se potom vypočítá jako poměr chybně klasifikovaných a celkového počtu vzorů, což vyjadřuje následující vzorec:

$$chyba = \frac{FP + FN}{TP + TN + FP + FN} \tag{10}$$

Klasifikátory se hodnotí mnoha způsoby, standardem se však stala tzv. *ROC křivka* (Recieiver Operating Characteristic). Tato křivka popisuje závislost FP na FN při různých nastaveních klasifikátoru.

3.1 Příznaky

Velmi podstatnou součástí všech metod statistického rozpoznávání vzorů je právě výběr příznaků. Přiznaky musí být použitelné pro klasifikaci, tzn. musejí od sebe co nejlépe oddělit vzory z odlišných tříd. Tento základní a nejdůležitější požadavek ilustruje následující obrázek:



Obrázek 6 - Použitelnost příznaků – (A) dobře separovatelné příznaky, (B) hůře separovatelné

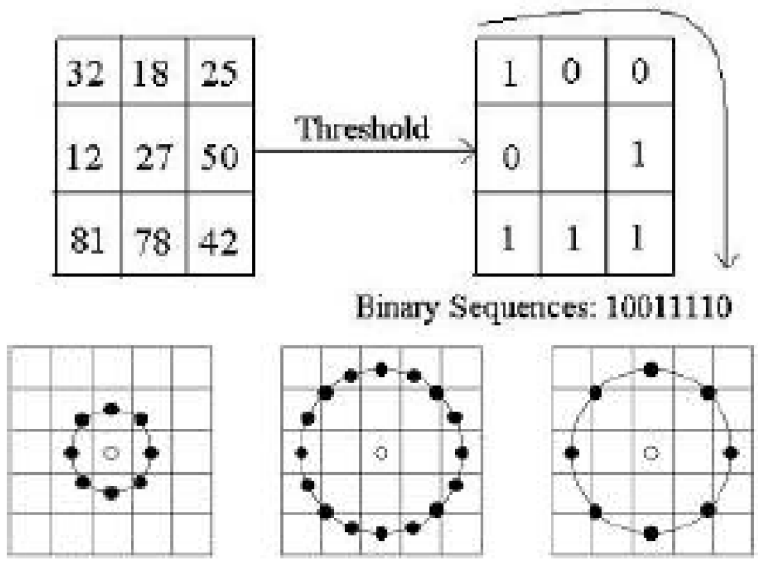
Na obrázku jsou dva typy příznaků, příznaky v levé části obrázku jsou dobře separovatelné do dvou tříd, zatímco příznaky napravo bychom velice těžce od sebe oddělovali.

U příznaků je také důležité, zda jsou invariantní vůči změnám v obraze. Mezi takovéto změny patří např. translace (poloha v obrázku), rotace, velikost, překrytí jiným objektem, úhel pohledu, změny osvětlení, deformace.

Ve výběru příznaků máme velmi velké možnosti. Jak bylo však již dříve zmíněno, je zde požadavek, aby bylo možné na základě zvolených příznaků dobře separovat vzory do jednotlivých tříd. Protože se tato diplomová práce zabývá detekcí obličeje, výběr příznaků je logicky podřízen tomuto účelu, což představuje další požadavky. V následující části uvedu několik příznaků, které se při detekci obličeje používají.

Local binary patterns - LBP

LBP je operátor poskytující lokální texturní informaci invariantní vůči změnám osvětlení. LBP je tvořeno binárním kódem, který vznikne prahováním určitého kruhového okolí. Jako práh je brán střed tohoto kruhu. Podrobnější informace naleznete v [4]. Postup tvorby LBP je patrný z obrázku 7 (převzato z [15]). Jednoduše je možné dosáhnout rotační invariance pomocí prosté bitové rotace.



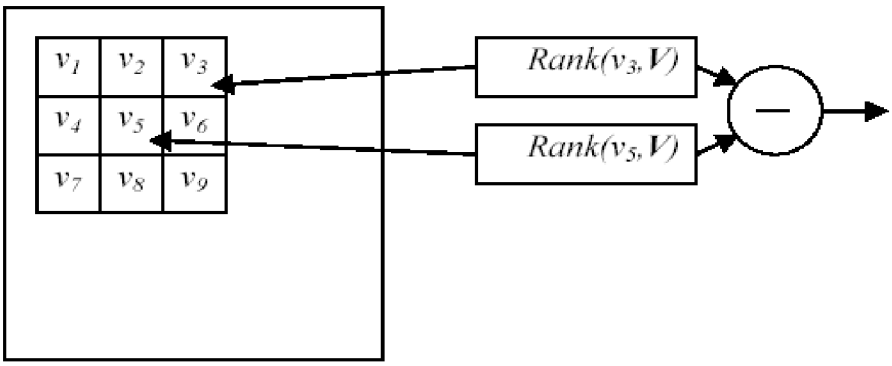
Obrázek 7 - Local binary patterns

Local Rank Differences - LRD

LRD (představeny v [15]) používají k výpočtu množinu hodnot $V=\{v_1, v_2, \ldots v_N\}$ odvozenou z určitého pravoúhlého okolí (jak naznačuje obrázek 8). Výsledná hodnota je definována následujícím vztahem:

$$LRD(v_a, v_b, V) = Rank(v_a, V) - Rank(v_b, V), \tag{11}$$

kde $V=\{v_1, v_2, \ldots, v_n\}$ a $Rank(v, V)$ vrací kolik čísel z V je menších než v.



Obrázek 8 - Local Rank Differences

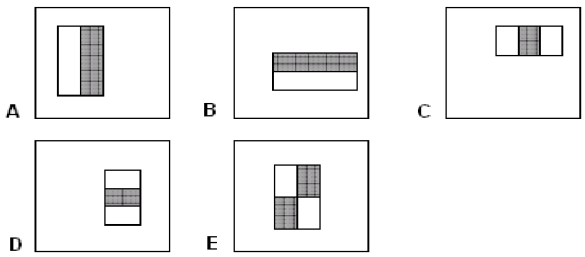
Haarovy vlnky

Haarovy vlnky, vycházející z Haarových funkcí, byly pro účel detekce obličeje poprvé použity v [5]. Jednorozměrná Haarova funkce je definována následujícím vztahem:

$$\psi(x) \equiv \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} < x \le 1 \\ 0 & otherwise \end{cases} \tag{12}$$

Haarovy vlnky používané v detekcí objektů jsou dvojrozměrnou variantou této funkce. Taková vlnka je reprezentována několika parametry. Jsou to:

- *druh*
- *x-ová a y-ová souřadnice – odpovídá levému hornímu rohu vlnky*
- *šířka a výška*



Obrázek 9 - Haarovy vlnky

Varianty A a B jsou rozděleny na dvě části. Tyto části mají stejnou velikost i tvar. Pro nás jsou důležité součty hodnot pixelů v těchto částech. Pokud máme vlnku typu A s těmito parametry: x-ová souřadnice x, y-ová souřadnice y, šířka w, výška h, pak se součet W pro bílou část vypočte podle jednoduchého vzorce:

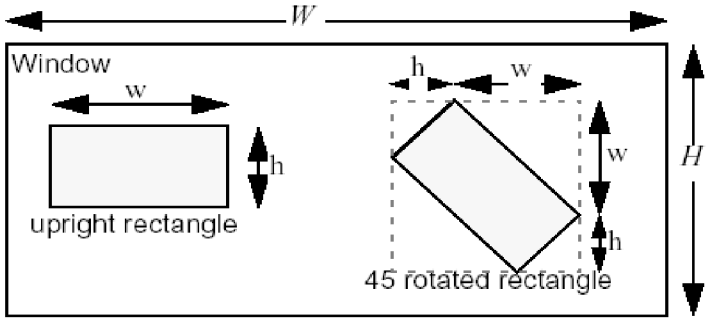
$$W = \sum_{i=x}^{x+\frac{w}{2}} \sum_{j=y}^{y+h} S(i, j), \tag{13}$$

kde $S(i,j)$ je hodnota pixelu ležícího na souřadnicích i, j. Součet pro šedou část (označme ji G) by se pak vypočítal analogicky. Výsledná hodnota vlnky je dána rozdílem G - W.

U variant C a D je vlnka rozdělena na tři stejné části a její výsledná hodnota je určena tak, že jsou od součtu hodnot pixelů v šedé oblasti odečteny součty bílých oblastí.

V případě vlnky typu E, která obsahuje čtyři shodné části, se odečte součet bílých oblastí od součtu oblastí šedých.

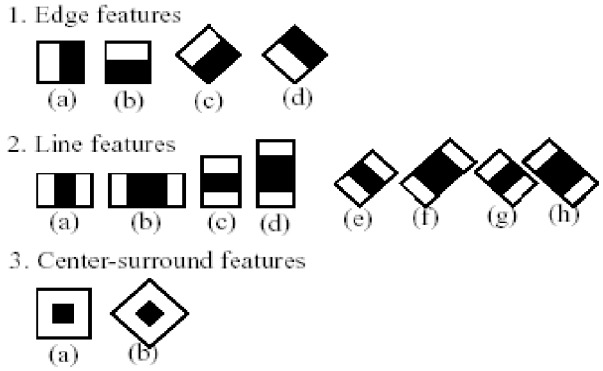
R. Lienhart a J.Maydt v článku [6] rozšířili původně používanou množinu Haarových vlnek o vlnky rotované o 45°, jak naznačuje následující obrázek (převzato z [6]).



Obrázek 10 - Rozšířené Haarovy vlnky

Tyto rotované vlnky přinášejí zlepšení při trénování a následné klasifikaci. Jsou totiž schopné postihnout některé vlastnosti obrazu, které jsou základními Haarovými vlnkami hůře popsatelné.

V této práci je použito i těchto rozšířených Haarových vlnek. Seznam použitých druhů vlnek je uveden na následujícím obrázku (převzato z [6]).



Obrázek 11 - Použité Haarovy vlnky

První druh Haarových vlnek, tzn. vlnky složené ze dvou částí jsou schopné popsat hrany v obraze a to ve čtyřech směrech podle orientace vlnky. Druhý typ dokáže vystihnout linii a třetí druh bod.

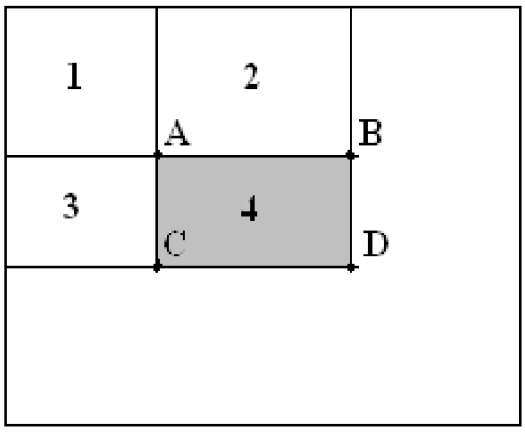
Tyto příznaky v podobě Haarových vlnek jsou velmi jednoduché v porovnání s jinými způsoby reprezentace obrazu, jako jsou např. Gáborovy vlnky, které poskytují lepší možnosti pro analýzu obrazu. Zde použité příznaky ale v kombinaci s integrálním obrazem, kterému bude věnována následující kapitola, nabízí velmi efektivní nástroj pro detekci obličejových rysů.

Integrální obraz

Integrální obraz (prezentován v [1]) představuje odlišnou reprezentaci obrazu, než je klasická matice hodnot jasu. Hodnota integrálního obrazu v bodě o souřadnicích *x, y* je rovna součtu pixelů v obdélníkové oblasti nad a nalevo od tohoto bodu(včetně jeho samého):

$$ii(x, y) = \sum_{x' \le x,\, y' \le y} i(x', y'), \tag{14}$$

kde $ii(x, y)$ je integrální obraz a $i(x, y)$ je obraz originální.



Obrázek 12 - Výpočet sumy pixelů pomocí integrálního obrazu

Díky této reprezentaci může být suma pixelů v obdélníkové oblasti vypočtena pouze na základě čtyř přístupů do paměti, jak naznačuje obrázek 12.

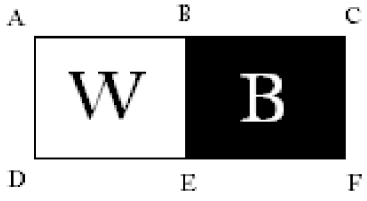
Hodnota integrálního obrazu v bodě *A* se rovná součtu pixelů uvnitř obdélníku 1. Hodnota v bodě *B* pak je 1+2, v bodě *C* 1+3 a konečně v bodě D 1+2+3+4. Z toho plyne, že součet pixelů uvnitř šedé oblasti 4 lze spočítat jako A+D-B-C.

Pro výpočet integrálního obrazu stačí díky následujícím rekurentním vztahům projít zdrojový obraz pouze jednou:

$$\begin{aligned} s(x, y) &= s(x, y-1) + i(x, y) \\ ii(x, y) &= ii(x-1, y) + s(x, y), \end{aligned} \tag{15}$$

kde $s(x, y)$ je suma všech pixelů ve sloupci nad bodem (x, y), $s(x, -1) = 0$ a $ii(-1, y) = 0$.

Pro vyhodnocení součtu pixelů v jakékoliv obdélníkové oblasti tedy stačí pouze čtyři přístupy do paměti. Výpočet hodnot Haarových vlnek je tedy velmi zefektivněn. V případě vlnky složené ze dvou částí si vystačíme pouze se šesti přístupy do paměti, u vlnky se třemi částmi stačí osm přístupů a vlnka se čtyřmi částmi potřebuje pro své vyhodnocení přístupů devět. Příklad výpočtu Haarovy vlnky složené ze dvou částí:



Obrázek 13 - Výpočet Haarovy vlnky

$$
\begin{aligned}
F &= W - B \\
F &= (A - B - D + E) - (B - C - E + F) \\
F &= (A + 2 \cdot E - 2 \cdot B - D + C - F)
\end{aligned}
\tag{16}
$$

Výpočet rotovaných Haarových vlnek je obdobný, je popsán v [6]. Použitím reprezentace pomocí integrálního obrazu se tedy značně zrychlí výpočet hodnot příznaků, které se jak při trénování detektoru, tak hlavně při samotné detekci budou vyhodnocovat velice často.

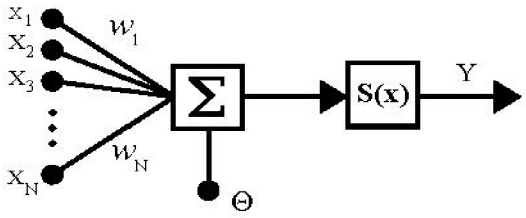
3.2 Metody klasifikace

Jak již bylo výše uvedeno, klasifikace je zařazení objektu do jedné z předem známých tříd. Toto se provádí na základě n-rozměrného vektorů příznaků. Úkolem je tedy najít separační hyperplochu, která oddělí data jednotlivých tříd od sebe. Tento problém se dá řešit mnoha způsoby, v následujícím textu budou uvedeny a stručně popsány metody, které se běžně pro tento účel používají.

Umělé neuronové sítě – NN

Vzorem pro chování neuronových sítí je lidský mozek. Umělá neuronová síť se skládá z umělých *neuronů*, což je jakási abstrakce biologického neuronu. Neurony jsou propojeny a předávají si signály, které různými transformacemi putují až do výstupu. Jako zdroj k NN jsem čerpal zejména z [9].

Umělý neuron se nejčastěji modeluje následujícím způsobem:



Obrázek 14 - Model umělého neuronu

Předchozí obrázek lze zapsat následovně:

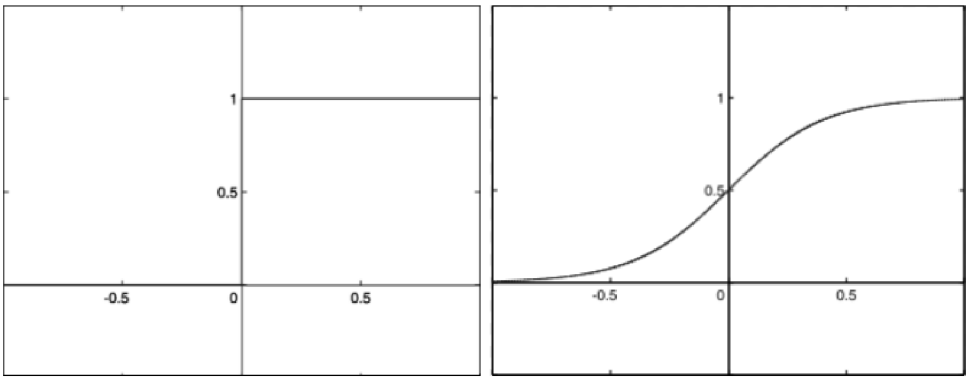
$$
Y = S\left(\sum_{i=1}^{N} (w_i x_i) - \theta\right), \tag{17}
$$

kde x_i jsou vstupy neuronu, w_i synaptické váhy, θ práh, $S(x)$ přenosová funkce neuronu (někdy aktivační funkce), a Y je výstup neuronu.

Přenosové funkce mohou být různé, jako například:

Skoková přenosová funkce – vrací nulu pro vstup menší než je stanovený práh, jedničku pro hodnoty vyšší.

Sigmoidální přenosová funkce – má tvar $f(x)=\frac{1}{1+e^{-kx}}$. Výhodou této přenosové funkce je, že oproti skokové má rozumné derivace v každém bodě.



Obrázek 15 - Skoková přenosová funkce (vlevo) a sigmoidální přenosová funkce

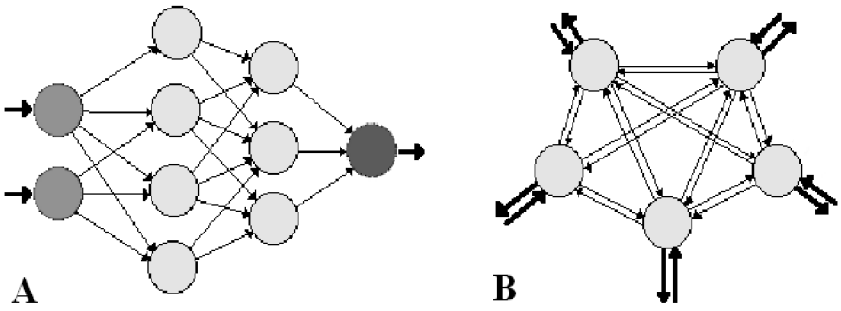
Dalšími přenosovými funkcemi jsou například funkce hyperbolické tangenty, či radiální báze.

Samotná neuronová síť je pak spojením určitého počtu takovýchto neuronů, které se (pokud mluvíme o vrstvových sítích) dělí do tří vrstev:

- **vstupní** – neurony, jejichž vstupy jsou vstupy celé neuronové sítě
- **skryté** – jsou spojeny s dalšími neurony, skrytých vrstev může být i více
- **výstupní** – jejich výstup je výstupem celé neuronové sítě

Důležitou vlastností každé sítě je způsob vzájemného propojení neuronů, toto se nazývá *architektura sítě*. Podle toku signálu neuronovou sítí pak sítě dělíme na:

- **rekurentní síť** – neboli zpětnovazební, mezi neurony existují zpětné vazby
- **dopředná síť** – vazby jsou vždy jen jedním směrem



Obrázek 16 - A - Dopředná NN, obsahuje 4 vrstvy. B - Rekurentní NN - neurony nejsou organizovány ve vrstvách

Aby mohla neuronová síť fungovat a tedy například správně klasifikovat, je třeba provést tzv. *učení*. Pro učení je nutné mít k dispozici množinu vstupních vektorů (učení bez učitele), nebo množinu dvojic vstupní vektor – požadovaný výstupní vektor (učení s učitelem). Existují různé způsoby učení, které se dají rozdělit do třech kategorií – *korelační učení*, *soutěživé učení* (upravují se váhy pouze „vítězného neuronu“) a *adaptační učení* (upravují se váhy všech neuronů sítě). Hojně používanou metodou učení neuronových sítí je tzv. metoda *backropagation*, která umožňuje učení dopředné NN pomocí upravování vah na základě zpětného šíření chyby.

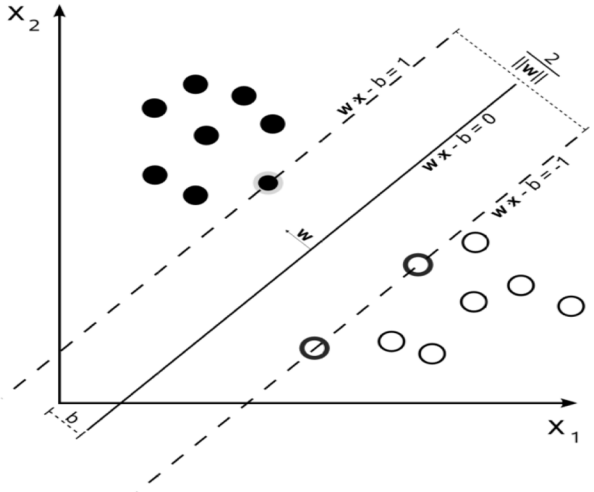
Umělé neuronové sítě kromě klasifikace můžou sloužit i k jiným účelům. Některé z nich přčedstavuje následující seznam:

- **Shlukování** – vytváření skupin podobných vstupních vektorů
- **Vektorová kvantifikace** – přiřazení vstupního vektoru nejbližšímu vzorovému vektoru
- **Funkční aproximace**
- **Predikce**
- **Identifikace systémů**
- **Optimalizace**

Support vector machines - SVM

Support vector machines (SVM) [8] je metoda strojového učení. Je určena pro klasifikaci lineárních dat. Pro nelineární data existuje možnost převést je do vícedimenzionálního prostoru (pomocí tzv. *jádrové transformace*), ve kterém je již možné data separovat lineárně.

Pomocí trénování se metoda snaží na základě trénovacích dat nalézt hyperpolochu nějlépe rozdělující data v prostoru. Optimalita rozdělení je definována jako maximum minima vzdálenosti mezi body různých tříd. Toto ilustruje následující obrázek:

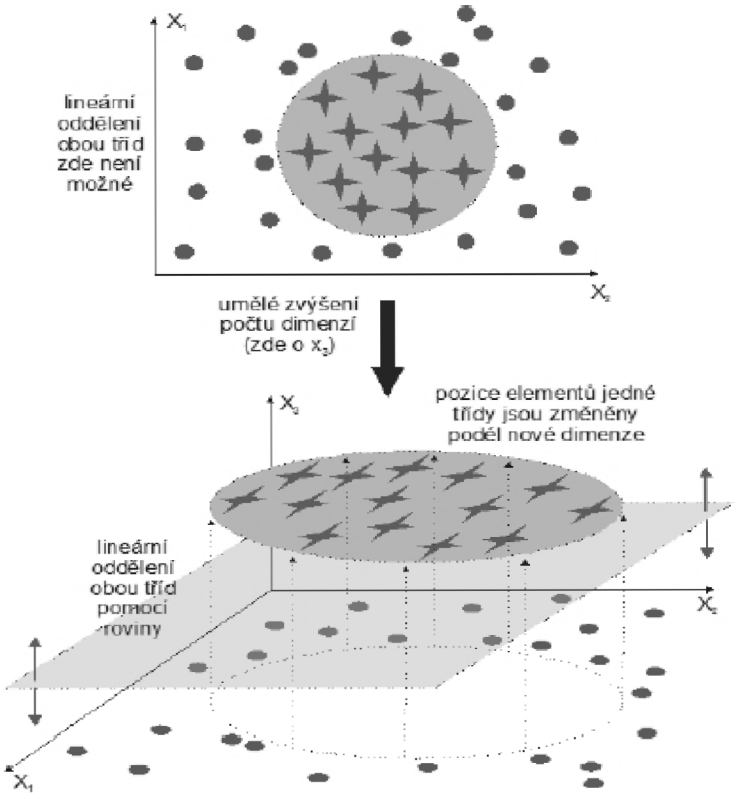


Obrázek 17 - SVN - Optimální dělící linie

Obrázek znázorňuje separační linii (plná čára) , pro níž platí, že vzdálenost dat od této linie je vyšší než pro kteroukoliv jinou dělící linii (hyperplochu pro vícedimenzionální prostor). Body, které se

nacházejí nejblíže separační linii, se nazývají **support vectors** (podpůrné vektory), vzhledem k tomu, že jejich funkcí je „podpora“ oddělovací linie, která na nich spočívá.

Již zmíněné převedení lineárně neseparovatelných dat do prostoru o více dimenzích, které umožní lineární oddělitelnost, popisuje následujícíc obrázek:



Obrázek 18 - SVM - projekce dat do vícedimenzionálního prostoru a jejich následná lineární separovatelnost

4 AdaBoost

AdaBoost (zkratka pro Adaptive Boosting) je algoritmus strojového učení spadající do tzv. Boosting metod. Tyto metody jsou založené na myšlence vytvořit množinu tzv. *slabých klasifikátorů*, jejichž klasifikační přesnost může být jen o trochu lepší než náhodný klasifikátor. Složením takovýchto slabých klasifikátorů do tzv. *silného klasifikátoru* je však vytvořen klasifikátor s přesností velmi vysokou. Výhodou takovýchto klasifikátorů je také jejich velmi rychlé vyhodnocení (díky jednoduchosti slabých klasifikátorů), což je vhodné pro úlohy, kde je rychlost klasifikace důležitá, jako např. detekce objektů ve videosekvenci v reálném čase.

AdaBoost byl poprvé představen v [2], pro detekci objektů jej bylo porvé použito v [1]. Je jednou z nejpoužívanějších variant boosting metod. Úkolem algoritmu je vybrat z velkého množství slabých klasifikátorů podmnožinu takovou, aby tyto klasifikátory byly schopny co nejlépe rozdělit vzory do příslušných tříd. V již zmiňovaném článku [2] autoři uvedli vedle základní varianty pro klasifikaci do dvou tříd i rozšíření na více tříd.

AdaBoost umožňuje při návrhu přidávat slabé klasifikátory (slabý klasifikátor – může klasifikovat pouze o něco málo lépe, než kdyby hádal náhodně – přesnost jen o málo lepší než 50%) tak dlouho, dokud není dosažena určitá požadovaná nízká hodnota klasifikační chyby skupiny klasifikátorů. Výsledný klasifikátor má tvar:

$$f(x)=\sum_{t=1}^{T}\alpha_t h_t(x), \tag{18}$$

kde $h_t(x)$ je slabý klasifikátor (hypotéza), α_t je váha příslušející danému slabému klasifikátoru, $H(x)=sign(f(x))$ je silný klasifikátor a T je počet slabých klasifikátorů ve výsledném silném klasifikátoru.

Vstupem je množina dvojic $(x_1,y_1),\ldots,(x_m,y_m); x_i \in X, y_i \in \{-1,1\}$, kde X je trénovací množina, y_i je předem známá klasifikace trénovacího prvku. Na začátku se váhy trénovacích vzorků inicializují. Celý algoritmus by se dal popsat následujícím pseudokódem:

Inicializace $D_1(i)=\frac{1}{m}$ (pro všechny vzorky)

Pro $t=1,\ldots,T$:

1. Vyber slabý klasifikátor $h_t : X \to \{-1,1\}$ s nejmenší chybou ε_t pro distribuci D_t
2. Přiřaď mu váhu $\alpha_t=\frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

3. Aktualizuj váhy trénovacích vzorků podle vzorce

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t},$$

kde Z_t je normalizační konstanta, taková, aby D_{t+1} odpovídalo rozdělení pravděpodobnosti.

Každý trénovací vzorek dostane přidělenu váhu α_t. Je-li vzorek klasifikován do správné třídy, jeho šance na opětovný výběr pro následující klasifikátor klesá; v opačném případě roste. AdaBoost se tak soustředí na obtížné vzorky.

Na počátku dostanou vzorky stejné hodnoty vah. V každém iteračním kroku t je výběrem tvořena trénovací množina vzhledem k těmto vahám(které ovlivňují pravděpodobnost výběru). Na této množině je natrénován klasifikátor h_t - je vybrán příznak s nejmenší chybou klasifikace trénovacích dat:

$$h_t = \min_{h_j \in H} err_j = \sum_{t=1}^{m} D_t(i)[y_i \neq h_j(x_i)],$$

kde err_j je chyba klasifikace.

Alfa se spočítá pomocí následujících vzorců:

$$\alpha_t = \frac{1}{2} \log\left(\frac{1+\varepsilon_t}{1-\varepsilon_t}\right),$$

kde $\varepsilon_t = \sum_{i=1}^{m} D_t(i) h_t(x_i)\, y_i$.

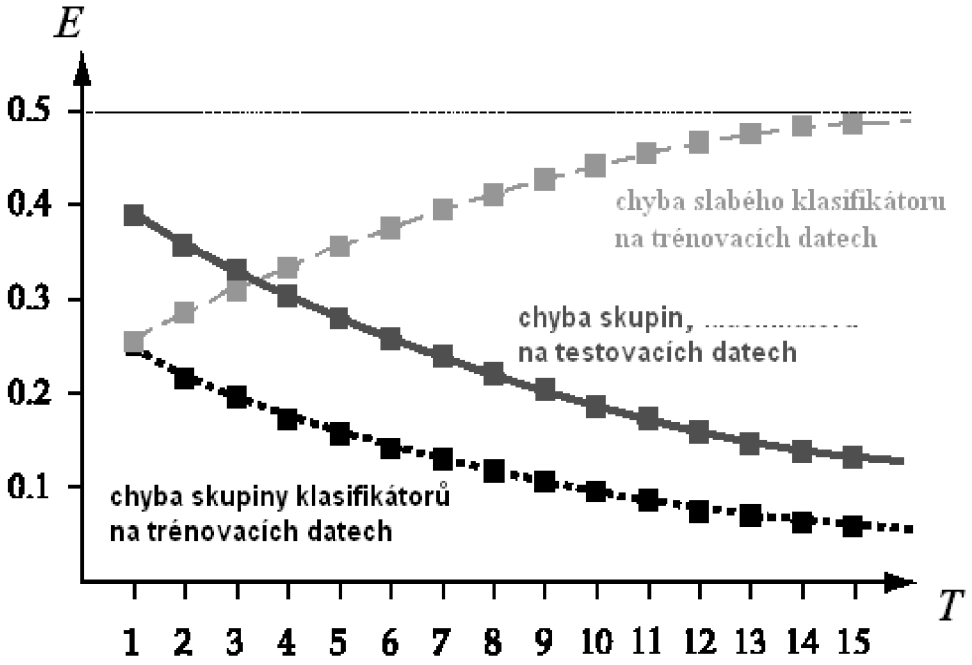
V následujícím kroku jsou zvýšeny váhy vzorků klasifikovaných chybně a sníženy váhy vzorků klasifikovaných správně pomocí h_t. Vzorky jsou pak na základě nového rozdělení pravděpodobnosti výběru použity při vytváření dalšího klasifikátoru a celý proces iterativně pokračuje.

I když mohou existovat případy dat, pro něž uvedený postup nemusí poskytnout očekávané výsledky, tak skupina slabých trénovaných klasifikátorů je schopna docílit libovolně nízké klasifikační chyby (pro trénovací data) pro dostatečně velký počet klasifikátorů T. Často je také výše uvedený cyklus trénování ukončován při dosažení určité (námi požadované) přesnosti.

Celková chyba silného klasifikátoru, tedy skupiny slabých klasifikátorů je pak dána vzorcem:

$$E = \prod_{t=1}^{T} \left[2\sqrt{E_t(1-E_t)}\right],$$

což ilustruje následující obrázek:



Obrázek 19 - Průběh chyb

AdaBoost redukuje trénovací chybu exponenciálně v závislosti na rostoucím počtu klasifikátorů T. Algoritmus *AdaBoost* je zaměřen na ty příklady, které způsobují klasifikační potíže, takže každý další přidaný klasifikátor má obecně větší chybu na trénovacích datech než kterýkoliv z předchozích (viz křivku *chyba slabého na trénovacích datech*). Dokud každý z klasifikátorů ve skupině dává lepší než náhodný výsledek (tj. např. pro dvě kategorie je chyba menší než 0.5), váhované rozhodnutí celé skupiny zaručuje pokles chyby na trénovacích datech. Velmi často dochází k podobnému poklesu i na testovacích datech.

Zvyšování počtu klasifikátorů ve skupině ale může vést k tzv. *přetrénování* (ztrátě schopnosti generalizovat vlivem přílišného zaměření klasifikátorů na rozeznávání pouze konkrétních trénovacích dat). Experimenty však ukázaly, že k tomu dochází relativně zřídka i pro extrémně vysoké hodnoty T . Pro praktickou aplikaci AdaBoostu platí základní pravidlo:

Boosting zlepšuje klasifikaci pouze tehdy, pokud klasifikátory ve skupině poskytují lepší než jenom náhodné výsledky – to však nelze zaručit předem. Uvedená metoda poskytuje v mnoha reálných problémech velmi dobré výsledky.

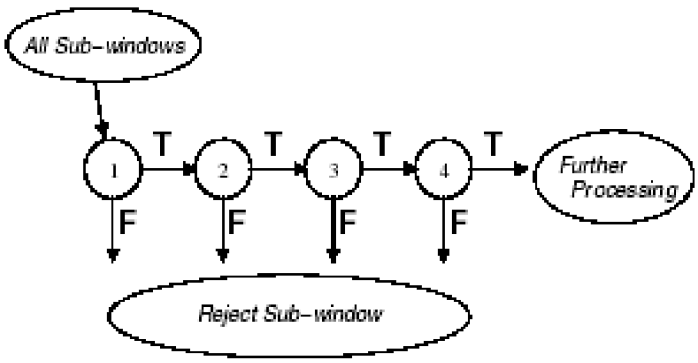
Kaskáda

V již zmiňované práci [1] bylo kromě algoritmu AdaBoost a integrálního obrazu použito i dalšího postupu, který výrazně vylepšil rychlost klasifikace. Jedná se o zapojení několika klasifikátorů vytvořených algoritmem AdaBoost za sebe do tzv. *kaskády*. Tento princip vychází z myšlenky, že v obraze je mnohem menší množství oblastí kde se hledaný objekt vyskytuje, než těch oblastí, kde se

nachází něco jiného (pozadí). Proto je žádoucí ušetřit čas rychlým klasifikováním pozadí. Z tohoto vyplývá, že by bylo dobré vytvořit jednodušší detektor, který bude schopen klasifikovat na základě několika málo vlnek pozadí, čímž výrazně sníží dobu detekce oproti variantě, kdy i oblasti pozadí musí projít celým složitým detektorem.

Principem kaskády je tedy zapojení několika postupně složitějších detektorů za sebe. Pozitivní odezva prvního klasifikátoru spouští vyhodnocení druhého a tak dále. Negativní odezva v jakémkoliv místě vede k „zahození" oblasti, tedy k jejímu označení za pozadí.

Jednotlivé detektory se nazývají *stage*. Jejich trénování probíhá algoritmem AdaBoost s následným upravením prahu tak, aby byla minimalizována hodnota false negative, tedy chybné zamítnutí. Toto je pochopitelné, neboť nám jde o to, abychom chybně nezahodili oblast, kde se objekt ve skutečnosti nachází. Článek [1] uvádí, že první stage může být složena např. pouze ze dvou příznaků.



Obrázek 20 - Kaskádní zapojení

WaldBoost

Vlastností algoritmu AdaBoost je, že je nutné vyhodnotit všechny slabé klasifikátory, než je možné zařadit vzorek do jedné ze tříd. Slabých klasifikátorů může být však velké množství, což se jeví jako nevýhoda. Algoritmus WaldBoost (představen v [10]) tuto nevýhodu odstraňuje. Klasifikátor nemusí vyhodnotit všechny slabé klasifikátory, ale pokud je dosaženo určitého prahu, lze vyhodnocení ukončit. To znamená výrazné urychlení detekce objektů.

FloatBoost

FloatBoost [12] je variantou AdaBoostu, která po každém přidaném slabém klasifikátoru projde zpětně všechny slabé klasifikátory a smaže ty, které nesnižují celkovou chybu klasifikace.

Totally Corrective Steps

Toto je opět variantou algoritmu AdaBoost. Rozdíl spočívá v tom, že po každém kroku, tzn. výběru slabého klasifikátoru jsou upraveny váhy všech slabých klasifikátorů, což snižuje celkovou chybu. Algoritmus byl představen v [11].

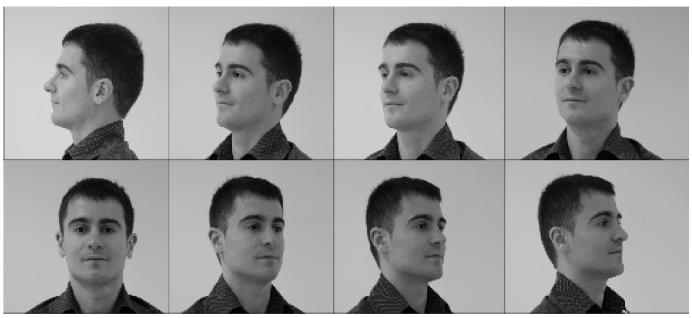
5 Detekce obličeje

Detekce obličeje, potažmo jeho lokalizace je úkol, který musí řešit každý systém určený pro práci s lidským obličejem. Tyto systémy se uplatňují zejména v oblastech jako jsou:

- ***Komunikace s počítačem*** – v současné době se výzkum v této oblasti soustředí na přirozenou komunikaci člověka s počítačem, tzn. komunikaci pomocí řeči, gest apod. Tento způsob komunikace je pro člověka pochopitelně pohodlnější a přirozenější než pomocí myši a klávesnice, či jiných existujících ovládacích prvků. Pro tvůrce takovýchto systémů však představuje nesrovnatelně náročnější problém a tím pádem použití daleko složitějších a sofistikovanějších postupů.
- ***Fotoaparáty a kamery*** – v dnešní době jsou již poměrně rozšířené např. fotoaparáty umožňující automatické zaostření na obličej nebo samospoušť, která vyfotí snímek až poté, co se fotograf přesune před objektiv.
- ***Biometrické systémy*** – tyto systémy umožňují automatické rozpoznávání lidí na základě jejich charakteristických anatomických rysů (např. otisk prstu, sítnice oka, obličej) a charakteristického chování (např. podpis, chůze) Člověk pak ke své identifikaci nepotřebuje heslo či průkaz, ale jeho totožnost je určena právě na základě nějakého jeho rysu. Tyto systémy v poslední době zažívají velký rozvoj a používají se na čím dál více místech. Tento trend je očekáván i do budoucna a dokonce snad i ve vyšší míře než doposud. Právě obličej je nejčastěji používanou biometrickou vlastností. Využití těchto systémů: letiště, přístupové systémy, docházkové systémy, policejní využití, kontrola návštěvníků akcí s velkým počtem lidí.

Detekce obličeje je prvním krokem ve všech těchto systémech, které dále již pracují jen s konkrétní oblastí obsahující obličej a zbytek ignorují. Proto je nutné, aby lokalizace byla co nejpřesnější, probíhala rychle a byla robustní. Při bližším pohledu se objeví problémy, se kterými je nutné při návrhu detektoru obličeje počítat. Jsou to hlavně:

- *Natočení hlavy*



Obrázek 21 - Natočení hlavy

- *Rotace hlavy*

Obrázek 22 - Rotace hlavy

- *Velikost obličeje v obraze*
- *Přítomnost některých komponent* - obličej obsahuje některé objekty, které ztěžují detekci, jako jsou například bradka, knír, či brýle.
- *Zakrytí* - Obličej může být částečně zakryt jiným objektem
- *Výrazy obličeje*

Obrázek 23 - Výrazy obličeje

- *Různé osvětlení* – intenzita, směr osvětlení
- *Nehomogenní pozadí* – tzn. pozadí není jednobarevné, ale obsahuje určité objekty, které komplikují detekci

Náročnost detekce obličeje se mění s podmínkami, které jsou na systém kladeny. Jednodušší samozřejmě je lokalizovat obličej ve snímku, který má jednotnou barvu pozadí. Zde se mohou uplatnit některé z jednodušších metod. Zato analyzovat snímek obsahující na pozadí změť různých objektů je problémem složitějším.

5.1 Existující metody

Detekce obličeje spadá do obecnější kategorie - *detekce objektů*. K této problematice existuje několik přístupů, které uvádí následující seznam. Při klasifikaci metod jsem vycházel z [3].

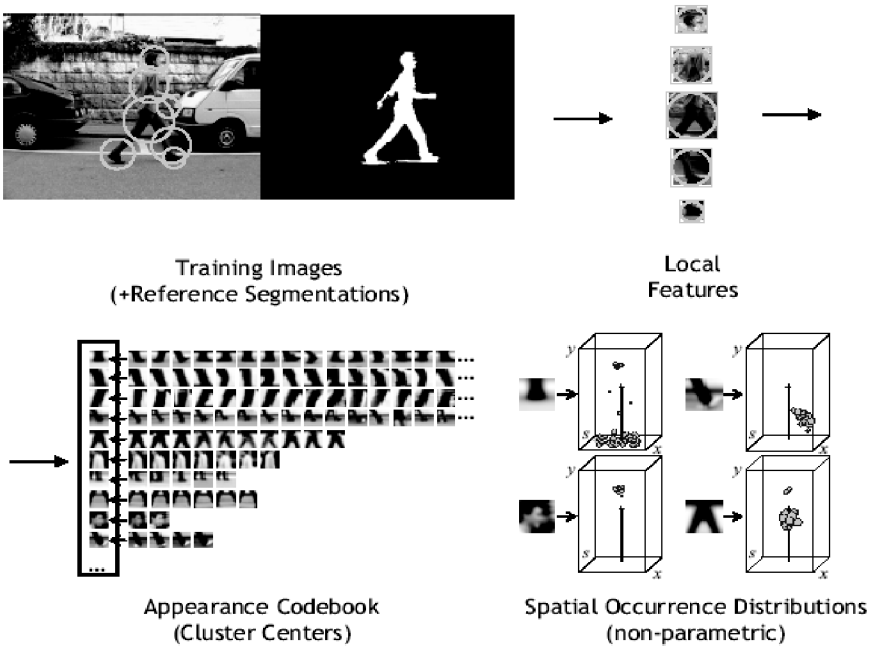
- ***Přístup shora dolů (Top-down)***
- ***Přístup zdola nahoru (Bottom-up)***
- ***Metody založené na šablonách (Template matching)***
- ***Appearance-based metody***

Přístup shora dolů (Top-down approach)

Zde se postupuje od vysoké údovně k detailům. Kandidáti jsou hledáni pomocí rychlých testů na nízké úrovni detailů. Z nich jsou pak podle detailnějších testů určovány skutečné detekce požadovaného objektu.

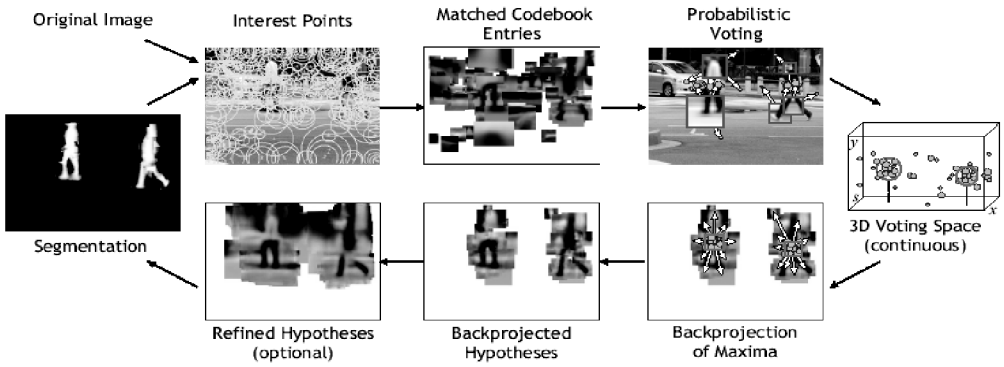
Přístup zdola nahoru (Bottom-up approach)

Tento přístup v současné době představuje velmi silný nástroj pro detekci objektů a výzkum v této oblasti se velmi rozvíjí. Přístup je založen na principu nalezení částí objektů a jejich následnému spojení do vyšších celků. V článku [13] je použit postup, kdy v trénovacích datech jsou extrahovány oblasti kolem význačných bodů. Tyto oblasti jsou uloženy společně s vyjádřenám své polohy v detekovaném objektu, jak naznačuje obrázek 24 (převzato z [13]):



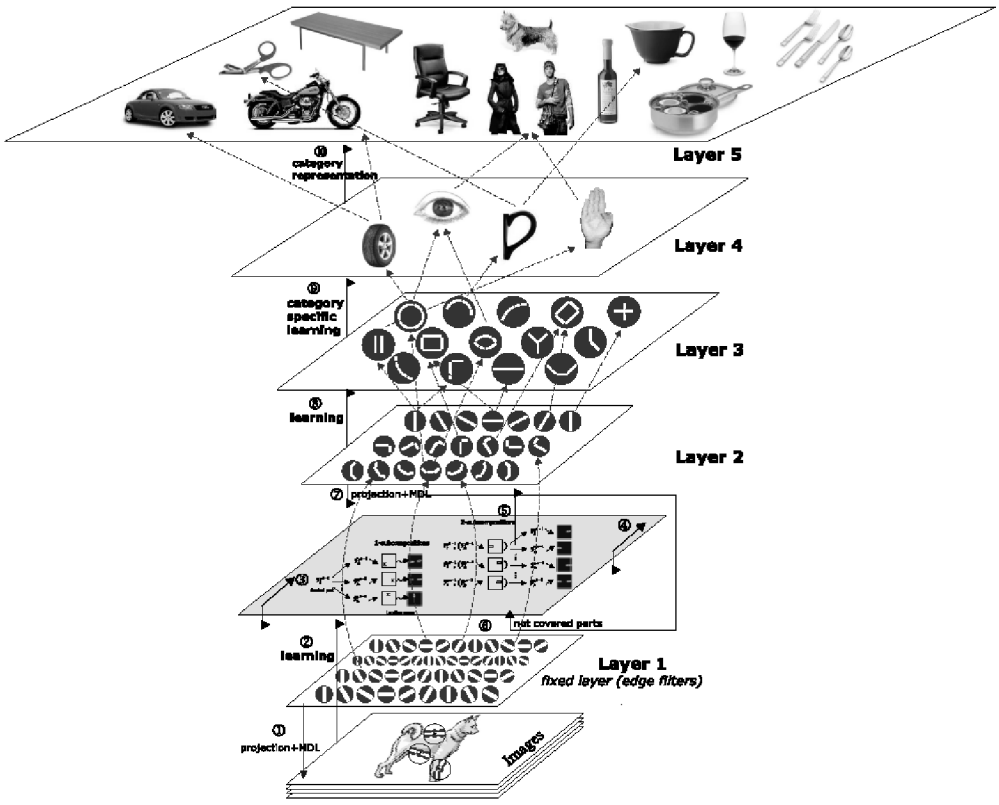
Obrázek 24 - Bottom-up přístup

Samotná detekce pak probíhá tak, že v obrázku jsou nadetekovány význačné body. Kolem každého z nich se opět extrahuje určité okolí, které je porovnáno s údaji získanými při trénování. Pokud si oblasti odpovídají, jsou údaje o jejich poloze v objektu započteny do 3D prostoru. Potom se již hledají lokálním maxima v tomto prostoru, které odpovídají hledanému objektu. Postup detekce ilustruje obrázek 25 (převzato z [13]):



Obrázek 25 - Postup detekce

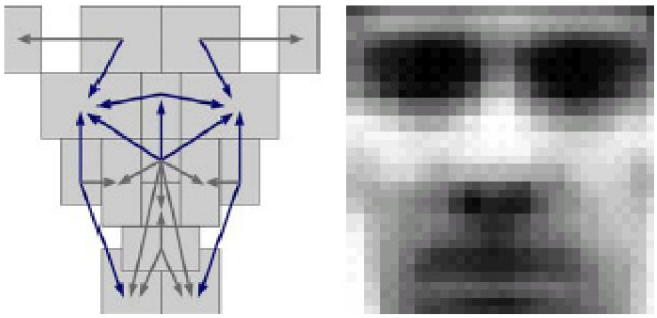
Článek [14] zase vytváří hierarchii objektů, které se postupně skládají do složitějších a složitějších celků, až je možné takto popsat celé komplexní tvary. Postupuje od nejjednodušších hranových filtrů přes jednoduché tvary (oblouk, roh, atd.) až k celým objektům, jak nažnačuje obrázek 26 (převzato z [14]):



Obrázek 26 - Bottom-up přístup - hierarchie

Metody založené na šablonách (Template matching, model fitting)

V tomto přístupu je vytvořen model objektu popisující tvar, barvu, či texturu – jakási šablona. Samotná detekce pak spočívá ve zjišťování, jak dané místo v obraze tomuto modelu odpovídá. Taková šablona může vypadat např. jako na obrázku 27. Vidíme zde určitý počet oblastí, mezi kterými jsou definovány poměry hodnot jasu.



Obrázek 27 - Příklad šablony

Appearance-based metody

Tento přístup představuje prohledávání obrazu pomocí klasifikátoru, který je natrénován na předložených datech. Vstupem pro klasifikátor mohou být hodnoty jednotlivých pixelů, toto se však moc nepoužívá. Častěji to jsou odezvy určitých (zpravidla jednoduchých) filtrů. Pro klasifikaci jsou pak použity metody, které byly popsány v kapitole 3.2. Jsou to např. neuronové sítě, support vector machines, či AdaBoost.

6 Návrh detektoru obličeje

Metoda detekce obličeje, která je předmětem této diplomové práce, je založena na detekci jednotlivých obličejových rysů (oči, nos, levý a pravý koutek úst) a následné kombinace těchto výsledků pro detekci celého obličeje.

Jednotlivé detektory jsou natrénovány algoritmem AdaBoost za použití Haarových vlnek jako příznaků. Dále jsou vyjádřeny polohy jednotlivých obličejových rysů vůči středu obličeje. Na základě těchto pravidel jsou detekce spojovány a provedena celková detekce obličeje.

Vzhledem k rozdělení metod detekce objektů uvedenému v kapitole 5.1 je tato metoda kombinací dvou přístupů. Natrénování detektorů jednotlivých obličejových rysů spadá do kategorie appearance-based metod, zatímco celkový přístup kombinace jednodušších částí do vyššího celku je příkladem přístupu zdola nahoru.

6.1 Tvorba detektorů obličejových rysů

Nejprve je nutné vytvořit detektory pro všechny požadované obličejové rysy. Byly zvoleny tyto:

- ***oko***
- ***nos***
- ***levý koutek úst***[1]
- ***pravý koutek úst***

Jako trénovací algoritmus byl použit AdaBoost ve variantě, která používá kaskádního zapojení pro urychlení detekce. Tento algoritmus byl popsán v kapitole 4. Jako příznaky byly použity Haarovy vlnky s rozšířením o vlnky rotované o 45°. Použité Haarovy vlnky jsou zobrazeny na obrázku 11. Trénování probíhalo pro velikost detekčního okénka 20x20 pixelů, což je také zároveň minimální velikost, kterou jsou klasifikátory schopny detekovat. Rysy o menší velikosti nejsou nalezeny. Obraz je při detekci postupně podvzorkováván, čímž je možné detekovat rysy v různých velikostech.

Výsledkem jsou tedy čtyři klasifikátory, jenž byly trénovány s následujícími parametry:

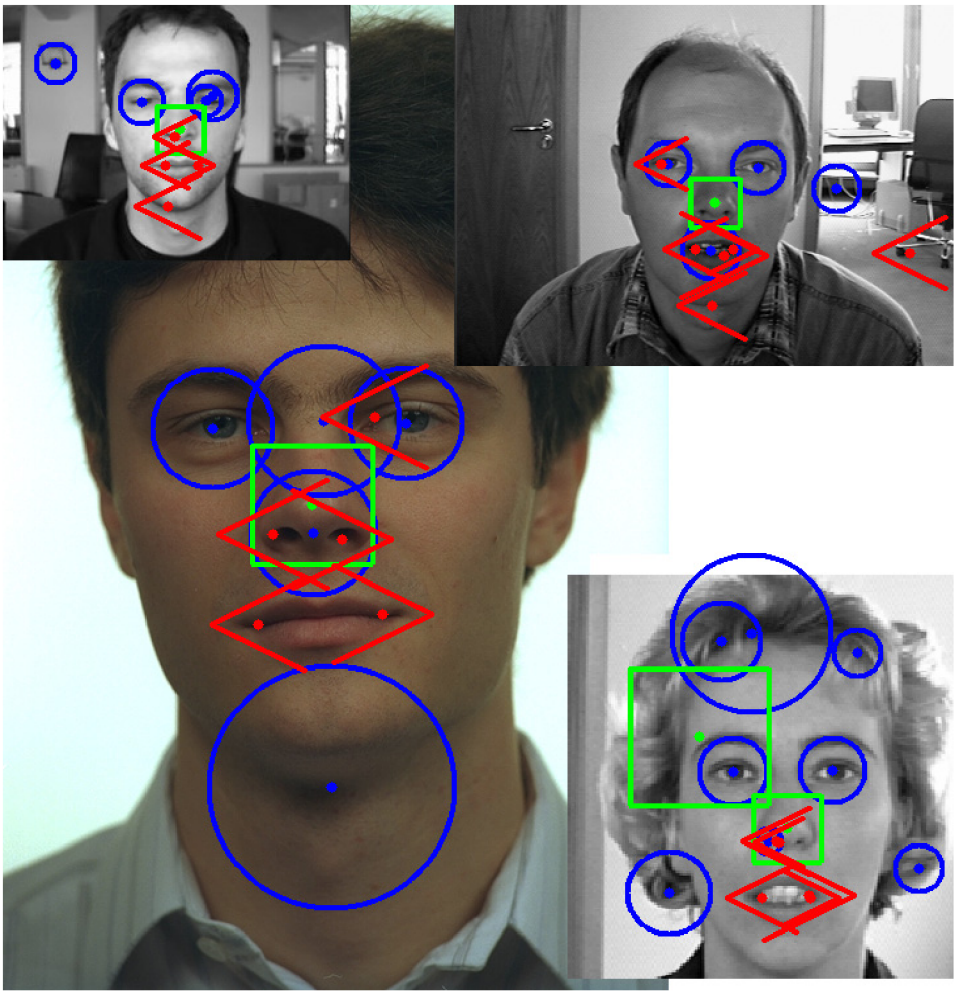
Rys	P	N	Stages	Min Hit Rate	Max False Alarm
Oko	1521	6000	24	0,995	0,5
Nos	1521	5000	15	0,995	0,5
Levý koutek úst	1521	5000	19	0,995	0,5
Pravý koutek úst	1521	4000	16	0,995	0,5

Tabulka 1 - Parametry detektorů obličejových rysů

1 V celé práci je za levou (pravou) stranu považována levá (pravá) strana z pohledu obličeje na obrázku

P je počet pozitivních vzorků pro trénování, *N* je počt negativních vzorků pro trénování, Stages výsledný počet detektorů zapojených v kaskádě (stage), *Min Hit Rate* – je minimální požadovaná hodnota Hit Rate (True Positive Rate) pro každou stage v kaskádě a *Max False Alarm* maximální požadovaná hodnota False Alarm Rate (False Positive Rate) pro každou stage v kaskádě.

Detektory byly trénovány na množinách o malé velikosti, tomu také odpovídá jejich relativně malá přesnost. Výsledky jednotlivých detekcí ilustruje obrázek 28:



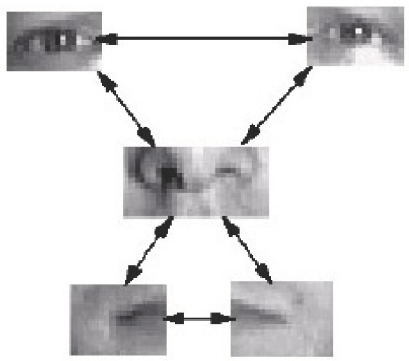
*Obrázek 28 - Výsledky detekce jednotlivých obličejových rysů - modrá kružnice představuje **oko**, zelený čtverec **nos** a červené šipky levý (pravý) **koutek úst***

6.2 Spojování detekcí

Ke spojení detekcí jednotlivých obličejových rysů do výsledného celku, tedy obličeje, je možné použít různé postupy. V této práci je použita metoda, kdy je pro každý rys vyjádřen jeho vztah ke středu obličeje, a to pomocí dvourozměrných Gaussových funkcí. Při samotné detekci jsou pak na základě odezev detektorů akumulovány příslušné Gaussovy funkce a poté po vyprahování určena poloha obličeje.

Vytvoření Gaussových funkcí

Jak již bylo řečeno, je potřeba pro každý obličejový rys vytvořit jeho vztah ke středu obličeje. Jednotlivé rysy mohou ležet pouze v určité oblasti obličeje, např. levý koutek úst může ležet pouze v malé oblasti vlevo dole od středu obličeje. Polohy jednotlivých obličejových rysů naznačuje obrázek



Obrázek 29 -Rozložení obličejových rysů

Jako vyjádření vztahu obličejového rysu vůči středu obličeje byla zvolena dvourozměrná Gaussova funkce, která odpovídá normálnímu rozdělení pravděpodobnosti, což vhodně modeluje situaci.

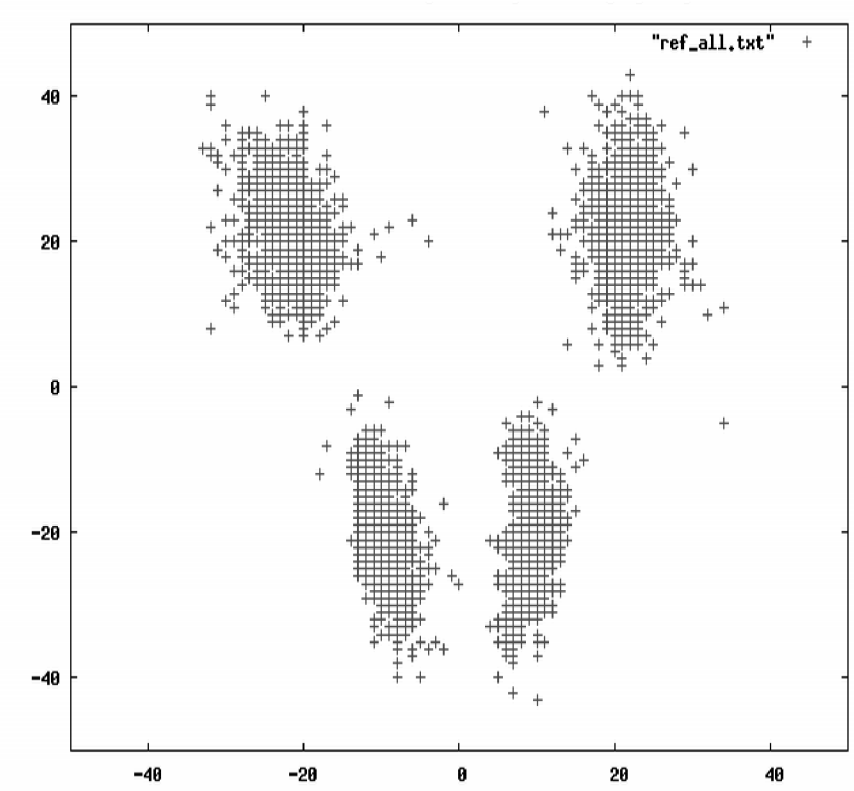
Parametry Gaussovy funkce vyjádříme na základě anotací použité obličejové databáze – BioID [17]. Anotace obsahuje souřadnice jednotlivých obličejových rysů. Jako vztažný bod bereme souřadnici špičky nosu. Vypočteme tedy vzdálenost příslušného obličejového rysu od souřadnice špičky nosu a tuto vzdálenost normalizujeme velikostí obličeje.

Tímto postupem získáme množinu bodů, z níž již můžeme jednoduše získat parametry Gaussovy funkce $N(\boldsymbol{x};\boldsymbol{\mu},\Sigma)$, kde $\boldsymbol{\mu}$ je střední hodnota a Σ je kovarianční matice. Vzorce pro výpočet parametrů následují:

$$\mu = \sum_{x} \boldsymbol{x}\, p(\boldsymbol{x}), \tag{19}$$

$$\sum = \sum_{x} (\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T p(\boldsymbol{x}). \tag{20}$$

Obrázek 30 představuje množiny bodů, vyjadřující vzdálenosti jednotlivých obličejových rysů od středu obličeje. Referenční vzdálenosti středu nosu vůči středu obličeje na obrázku uvedené nejsou, neboť jako referenční bod pro výpočet vzdáleností byla brána právě špička nosu, a tudíž jsou všechny vzdálenosti nulové.



Obrázek 30 - Referenční vzdálenosti jednotlivých obličejových rysů - levý horní shluk bodů představuje pravé oko (z pohledu obličeje na obrázku), pravý horní shluk levé oko, levý dolní shluk pravý koutek úst a pravý dolní shluk levý koutek úst

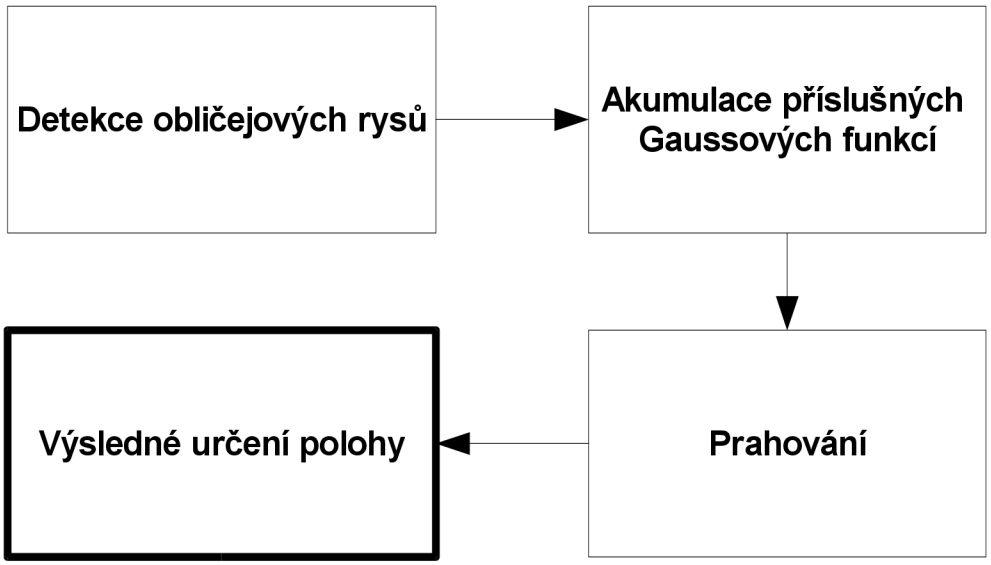
Z těchto množin bodů byly odhadnuty Gaussovy funkce s parametry uvedenými v tabulce 2. Gaussova funkce pro nos byla z výše uvedených důvodů vyjádřena jako křivka se stejným rozptylem hodnot ve směru x i y a střední hodnotou v bodě [0,0].

Rys	Střední hodnota	Kovarianční matice
Levé oko	$[-21,21]$	$\begin{bmatrix} 9 & -5 \\ -5 & 32 \end{bmatrix}$
Pravé oko	$[21,21]$	$\begin{bmatrix} 6 & 0 \\ 0 & 47 \end{bmatrix}$
Nos	$[0,0]$	$\begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$
Levý koutek úst	$[-9,-19]$	$\begin{bmatrix} 4 & -4 \\ -4 & 35 \end{bmatrix}$
Pravý koutek úst	$[9,-20]$	$\begin{bmatrix} 3 & 2 \\ 2 & 38 \end{bmatrix}$

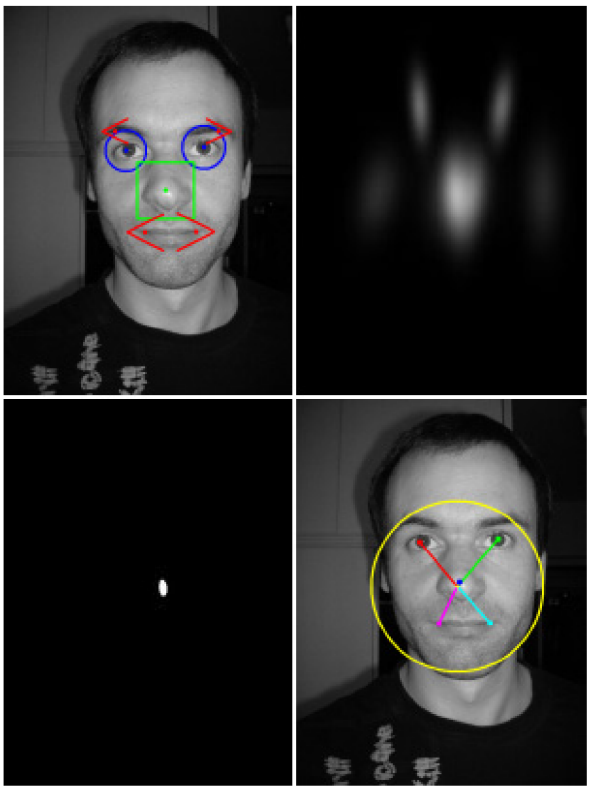
Tabulka 2 - Parametry Gassových funkcí pro jednotlivé obličejové rysy

6.3 Detekce

Princip samotné detekce je nažnačen na obrázku 31. Následuje detailní popis jednotlivých kroků spolu s demonstrací na vybraném obrazu.



Obrázek 31 - Schéma procesu detekce



Obrázek 32 - Jednotlivé kroky detekce

Detekce obličejových rysů

Nejprve je tedy obraz analyzován všemi detektory obličejových rysů (oči, nos, levý, pravý koutek úst). Výsledek takovéto detekce je naznačen na obrázku 32 v levé horní části. Jak je možné vidět, detektory mohou (a často mívají) špatnou odezvu. Zde například detektory koutků úst reagují také na okraj obočí, jehož tvar je do značné míry podobný koutku úst.

Akumulace Gaussových funkcí

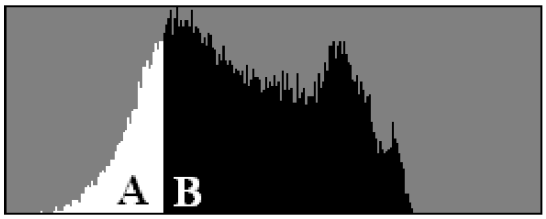
Ke každému detekovanému obličejovému rysu patří jedna Gaussova funkce (pro oči dvě, neboť máme pouze jeden detektor pro obě oči, čili nevíme, které oko jsme detekovali). Ta nám vyjadřuje, kde by měl ležet střed obličeje k tomuto obličejovému rysu. Pokud tedy budou nadetekovány všechny obličejové rysy, součet příslušných Gaussových funkcí bude vysoký a právě v bodě tohoto maxima by měl ležet obličej.

V této fázi je vytvořeno dvourozměrné pole (obraz) o stejné velikosti jako zpracovávaný obraz, v kterém se sčítají hodnoty jednotlivých Gaussových funkcí. Toto pole nutně nemusí mít velikost původního obrazu, v mé implementaci je pole podvzorkováno desetkrát, neboť tato velikost nám stačí pro relativně přesnou detekci obličeje.

Spočítáme tedy součet průběhů všech Gaussových funkcí. S tím omezením, že funkce od rysů stejných druhů nesčítáme. To proto, že pokud by např. detektor oka měl několik špatných odezev blízko sebe, průběhy Gaussových funkcí by se nesprávně sčítali a docházelo by k výskytu nesprávných lokálních maxim. Krok akumulace Gaussových funkcí ilustruje pravá horní část obrázku 32.

Prahování

Pro hledání lokálních maxim v obraze vytvořeném v předchozím kroku byla zvolena velmi jednoduchá metoda. Z obrazu je vytvořen histogram. Hodnota prahu je poté stanovena tak, aby rozdělovala plochu histogramu na dvě části v určitém poměru (např. 0,1%). Toto vyjadřuje následující obrázek 33. Jedná se tedy o poměr ploch *A* a *B*. Ve výsledné aplikaci byl tento poměr po testech stanoven na 0,1%. Vyšší či nižší hodnoty znamenaly snížení True Positive Rate a zvýšení False Positive Rate. Po vyprahování obrazu touto hodnotou získáme binární obraz, ve kterém bílé hodnoty představují oblasti, kde se teoreticky nachází obličej, jak můžeme vidět na obrázku 32 v levé dolní části. Toto je také vstupem do další části metody, která z těchto oblastí získá konkrétní souřadnice.



Obrázek 33 - Histogram

Výsledné určení polohy obličeje

V této fázi máme tedy binární obraz, ve kterém bílá místa představují oblasti teoreticky obsahující obličej. Oblasti jsou segmentovány pomocí semínkového vyplňování a výsledné souřadnice, které již představují polohu obličeje, jsou spočítány jako střed každé této oblasti.

7 Implementace

Implementačním jazykem byl zvolen programovací jazyk C++. Pro usnadnění práce s obrazem byla použita knihovna OpenCV verze 1.0 [18]. Tuto knihovnu jsem použil zejména proto, že obsahuje třídy umožňující trénování a detekci pomocí metody AdaBoost.

Na přiloženém CD je projekt do Microsoft Visual Studia 2005, který je možné přeložit a spustit, CD navíc obsahuje již přeložený program, spustitelný na platformě Windows.

Vytvoření detektorů

K natrénování detektorů byl využit trénovací software dodávaný s instalací OpenCV. Tento software obsahuje programy pro vytvoření trénovacích množin a následné trénování dle zadaných požadavků. Výstupem je xml soubor s natrénovaným klasifikátorem.

Pro vytvoření trénovacích množin byla použita obličejová databáze BioID [17], která obsahuje 1521 obličejů. Snímky obsahují anotaci jednotlivých význačných obličejových bodů.

Jednotlivé klasifikátory byly natrénovány s parametry, které uvádí tabulka 1.

Třída pro detekci obličeje

Pro účel prezentace navrhované metody byly vytvořeny jednoduché třídy umožňující detekci obličeje. Struktura tříd je popsána v následující části:

Třídy popisující obličejový rys a obličej:

```
class FacialFeature
{
  public:
    FacialFeature();
    ~FacialFeature();

    TFacialFeatureType type; // typ oblicejoveho rysu (nos, leve oko,...)
    int x;                   // souradnice rysu
    int y;
    int width;               // velikost rysu
};

class Face
{
  public:
    Face();
    ~Face();
    int x;
    int y;                    // souradnice obliceje
    int width;                // velikost obliceje

    bool rEFound;             // zda byly jednotlive rysy nalezeny
    bool lEFound;
    bool nFound;
    bool rMCFound;
    bool lMCFound;
```

```
    FacialFeature rEFeature;  // oblicejove rysy patrici k danemu obliceji
    FacialFeature lEFeature;
    FacialFeature nFeature;
    FacialFeature rMCFeature;
    FacialFeature lMCFeature;
};
```

Třída představující samotný detektor:

```
class Detector
{
  public:
    Detector();
    ~Detector();

    bool m_bClassifiersLoaded;    // jsou nactene klasifikatory

    CvHaarClassifierCascade *m_EyeClassifier;
    CvHaarClassifierCascade *m_NoseClassifier;
    CvHaarClassifierCascade *m_LeftMouthCornerClassifier;
    CvHaarClassifierCascade *m_RightMouthCornerClassifier;


  int DetectFace(const char *imgSourceFile, T_FaceVector &faces,
                 T_FacialFeatureVector &facialFeatures, double thresh);
};
```

Metoda `DetectFace` provede detekci v obrazu imgSourceFile, ve vektoru `faces` uloží nadetekované obličeje a do vektoru `facialFeatures` všechny nadetekované obličejové rysy. Hodnota `thresh` slouží pro nastavení dělící hodnoty při prahování (viz kapitola 6.3).

Výsledný program

Výsledný program je konzolová aplikace, která výsledek detekce vykresluje do okna vytvořeného pomocí OpenCV.

Parametry programu jsou následující:

```
ffd.exe -files <images_file> [-features]
```

kde `<images_file>` je soubor obsahující seznam souborů, na kterých chceme detekci provést. Pokud je zadán volitelný parametr `-features`, jsou do výsledného okna vykresleny i všechny nadetekované obličejové rysy (i ty, které nejsou součástí žádného obličeje).

8 Výsledky

Pro testování obličejového detektoru byla použita část databáze FERET. Souřadnice detekovaného obličeje byla porovnána s údajem uvedeným v anotaci a pokud byla vzdálenost mezi těmito dvěma body menší než určitá stanovená konstanta, byla detekce brána za správnou.

Test byl proveden na celkem 2000 obrázcích, 1000 z nich obsahovalo obličej (každý obrázek právě jeden obličej) a 1000 z nich obličej neobsahovalo. Byly naměřeny tyto hodnoty:

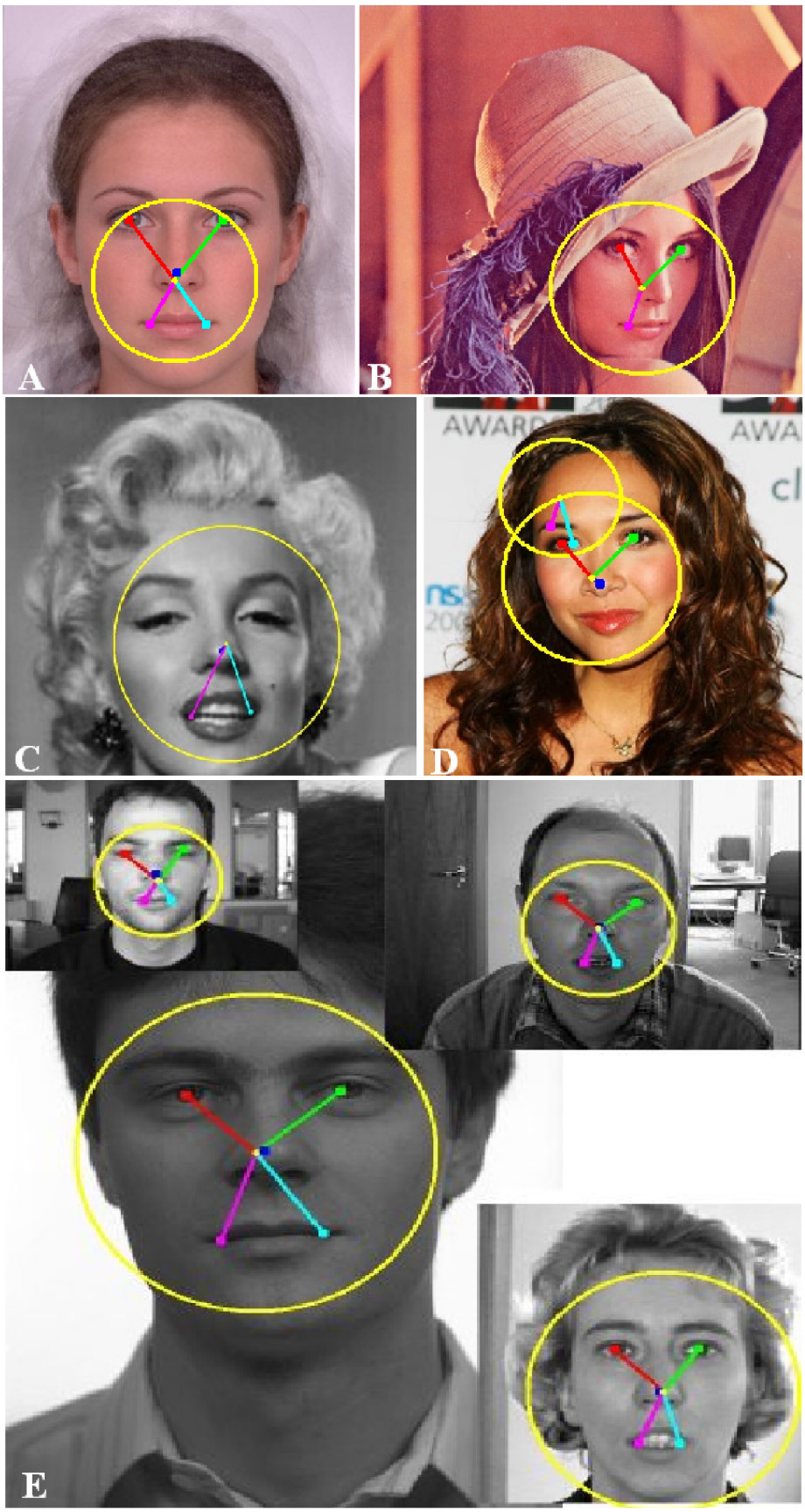
P	N	True Positive Rate	False Positive Rate
1000	1000	0,864	0,094

Ukázky detekcí vidíme na obrázku 34. Detekovaný obličej je vykreslen žlutou kružnicí včetně svého středu. Dále jsou pak znázorněny jednotlivé obličejové rysy (včetně vztahu ke středu obličeje), pokud jsou detekovány. Červeně je vykresleno pravé oko, zeleně oko levé, modře pak nos, fialová barva představuje pravý koutek úst a světle modrá koutek levý.

Obrázek A ukazuje správnou detekci, kdy byly nalezeny všechny obličejové rysy. Na obrázcích B, C a D došlo vždy k detekci pouze některých obličejových rysů , přesto byl obličej i tak detekován. Na obrázku D navíc můžeme vidět chybnou detekci, kdy koutek oka byl nesprávně detekován jako koutek úst a roh obočí jako druhý koutek úst. Výskyt těchto chybných detekcí v takovéto vzájemné poloze však má za výsledek, že je oblast označena nesprávně jako obličej. Obrázek E pak demonstruje detekci obličeje v různých velikostech.

K detekci obličeje nemusí vždy nutně být nalezeny všechny obličejové rysy. Metoda funguje i v případě, že jsou nalezeny např. jen tři, či čtyři obličejové rysy.

Jak je vidět, i z jednoduchých detektorů obličejových rysů lze spojováním vytvořit systém, který dosahuje relativně dobré přesnosti.



Obrázek 34 - Ukázky detekce obličeje

9 Závěr

Tato diplomová práce si kladla za cíl navrhnout a implementovat detektor obličeje. Je zde uveden přístup, který nejprve detekuje jednotlivé obličejové rysy (levé a pravé oko, nos, levý a pravý koutek úst). Výsledky těchto detekcí jsou pak podle určitých pravidel spojovány do finální detekce celého obličeje.

Klasifikátory jednotlivých obličejových rysů byly natrénovány algoritmem AdaBoost, jako příznaky pro klasifikaci byly použity Haarovy vlnky. Tyto klasifikátory byly natrénovány na relativně malé trénovací sadě, přesto je vidět, že díky spojování jednotlivých detekcí je i pomocí těchto jednoduchých klasifikátorů možné vytvořit systém, který je schopen detekovat obličej s relativně dobrou přesností.

Jak je vidět z výsledků, ke správné detekci obličeje není nutné nalezení všech obličejových rysů. Metoda funguje i v případě, že jsou nalezeny např. jen tři, či čtyři obličejové rysy.

Jako možné rozšíření do budoucna se nabízí použití složitějších klasifikátorů obličejových rysů (trénované na více datech), které by jistě úspěšnost detektoru zvýšily.

Literatura

[1] Viola, P. A., Jones, M. J., *Robust Real-time Object Detection*. International Conference on Computer Vision, pp. 747, 2001.

[2] Freund, Y., Schapire, R., *A decision-theoretic generalization of on-line learning and an application to boosting*. In 2nd European Conference on Computational Learning Theory, 1995.

[3] Yang, M., Kriegman, D. J. and Ahuja, N., *Detecting Faces in Images: A Survey*. In IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, January 2002.

[4] Pietikäinen, M., *Image Analysis with Local Binary Patterns*. SCIA 2005: 115-118

[5] Papageorgiou, C., Oren, M. and Poggio, T., *A general framework for object detection*. In International Conference on Computer Vision, 1998.

[6] Lienhart, R., Maydt, J., *An Extended Set of Haar-like Features for Rapid Object Detection*. Submitted to ICIP2002.

[7] Lee, T.S., *Image representation using 2D Gabor wavelets*. IEEE Transection of Pattern Analysis and Machine Intelligence. Vol. 18, No. 10, (1996) 959-971.

[8] wikipedia.org, *Support vector machine* [online] [cit. 23.1.2009], dostupný z URL http://en.wikipedia.org/wiki/Support_vector_machine

[9] wikipedia.org, Artificial neural network [online] [cit. 23.1.2009], dostupný z URL http://en.wikipedia.org/wiki/Artificial_neural_network

[10] Šochman J., Matas J., *WaldBoost - Learning for Time Constrained Sequential Detection*, Center for Machine Perception, Dept. of Cybernetics, Faculty of Electrical Engineering, CVUT Prague, 2005

[11] Šochman, J., Matas, J., *AdaBoost with Totally Corrective Updates for Fast Face Detection*. Sixth IEEE International Conference on Automatic Face and Gesture Recognition (2004) p. 445

[12] Li, S. et al., *FloatBoost learning for classification*. In: S. Thrun S. Becker and K. Obermayer, editors, NIPS 15. MIT Press (2002)

[13] Leibe, B., Leonardis, A., Schiele, B., *Robust object detection by interleaving categorization and segmentation.* International journal of computer vision, 2008, vol. 77, no. 1/3, pp. 259-289.

[14] A. Leonardis and S. Fidler., *Learning hierarchical representations of object categories for robot vision.* 13th International Symposium of Robotics Research (ISRR), 2007.

[15] Hradiš M., Herout A., Zemčík P., *Local Rank Patterns - Novel Features for Rapid Object Detection*, In: Proceedings of International Conference on Computer Vision and Graphics 2008, Heidelberg, DE, 2008, s. 1-12

[16] Španěl M., *Rozpoznávání Gest ve video sekvencích*, Vysoké Učení Technické v Brně, 2003

[17] BioID Face DB, [online] [cit. 21.1.2009], dostupná na URL http://www.bioid.com/downloads/facedb/index.php

[18] OpenCV knihovna, [online] [cit. 21.1.2009], dostupná na URL http://opencvlibrary.sourceforge.net