Algoritmy pro rozpoznání obličeje

Face Recognition Algorithms

Marek Michalík

Bakalářská práce
2008

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2007/2008

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Marek MICHALÍK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Algoritmy pro rozpoznání obličeje**

Zásady pro vypracování:

1. **Literární rešerše k identifikaci obličejů.**
2. **Výběr optimálního algoritmu.**
3. **Napsání jednoduché aplikace v jazyce Java pro nalezení obličejů na hromadné forografii.**
4. **Návrh dalšího postupu při identifikaci.**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Uwechue, A. O. – Pandya, A. S.: Human face recognition using third-order synthetic neural networks. Kluwer academy, 1998.**
2. **Mihrosseini, A. R. – Hong, Y.: Human face image recognition: An evidence aggregation aproach. Australia, 1998.**
3. **Hinner, J.: Detekce a rozpoznávání obličejů osob a jejich identifikační význam. Kriminalistika, 3, 2003.**

Vedoucí bakalářské práce: **doc. RNDr. Petr Ponížil, Ph.D.**
Ústav fyziky a mater. inženýrství

Datum zadání bakalářské práce: **20. února 2008**

Termín odevzdání bakalářské práce: **5. května 2008**

Ve Zlíně dne 20. února 2008

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Práce se zabývá problémem vyhledávání a identifikace obličejů na fotografiích. V teoretické části jsou popsány metody detekce oblastí na fotografiích, které by mohly být lidským obličejem. Pozornost je věnována také postupům používaných pro identifikaci konkrétního člověka. V praktické části byl vyvinut program pro detekci obličejů na skupinových fotografiích vyhledáváním oblastí s barvou odpovídající barvě kůže.

Byl také napsán program pro zpracování obecných charakteristik obličejů z databáze 3061 fotografií studentů UTB použitelných dále pro detekci obličejů metodou porovnávání vzorů.

Klíčová slova: rozpoznávání obličeje, identifikace osob na fotografii

ABSTRACT

This bachelor work deals with the problem of searching and identification of faces in the photographs. In the theoretical part there are described methods for detection of areas with presumable face occurrence in the photograph. The attention was paid to the process for identification of the particular person as well. The program for face detection in the group photographs by identification of conformable skin colour areas was developed in the practical part.

The program for proccesing of common face characteristics from the database of 3061 TBU students photographs was worked up too. These photographs were consequently used for the face detection by the matching patterns method.

Keywords: face recognition, identification of persons in the photograph

Na tomto místě bych velmi rád poděkoval doc. RNDr. Petru Ponížilovi, Ph.D. za odborné vedení, cenné rady, připomínky a konzultace, které mi poskytoval během řešení mé bakalářské práce a tím mi pomohl k jejímu vytvoření. Za poskytnutí fotografií studentů děkuji Ing. Radomíru Chlupovi. Dále bych rád poděkoval svojí rodině za podporu při studiu.

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně 15.5.2008 …….

Podpis diplomanta

OBSAH

ÚVOD

Biometrické metody identifikace osob nacházejí uplatnění v různých oblastech bezpečnostních a přístupových systémů. Např. přístup do objektů, kdy jsou různé biometrické metody kombinovány s čipovými přístupovými kartami. Převážně se zde používají metody biometrie dlaně, daktyloskopické metody, analýza hlasu, porovnávání duhovky a rohovky apod. V poslední době se biometrické metody uplatňují i v zabezpečení přístupu do systémů personálních počítačů a přístupů do firemních informačních sítí (klávesnice se snímáním daktyloskopického otisku prstu).

V kriminalistice policie využívá řadu metod ke zjištění totožnosti zájmové osoby. Kromě standardní metody zjišťování totožnosti podle osobních dokladů, spočívají tyto metody v obecné rovině v nalezení, získání, určení a vyhodnocení jednoznačných identifikátorů, které mohou přispět k jednoznačnému určení totožnosti zájmové osoby. Ze sociologického hlediska je rozpoznávání známých tváří nejběžnější a nejčastější podvědomou činností lidského mozku. Automatizované rozpoznávání lidských obličejů je obtížný komplexní úkol z důvodů proměnlivosti základních fyzikálních veličin obrazu, jakosti a fotometrie, geometrie - úhlu natočení a přiblížení, morfologie změn - emoční výrazy obličeje a stárnutí - a "maskování" (čepice, brýle, vousy). Odtud vyplývá nutnost vytvoření normalizovaného modelu lidského obličeje tak, aby scénář rozpoznávání nebyl ovlivněn těmito reálnými, nicméně rušivými vlivy.

I. TEORETICKÁ ČÁST

PROCES ROZPOZNÁVÁNÍ OBLIČEJE

Automatizovaná identifikace osob může být řešena dvěma základními přístupy [1]. Řešení úlohy identfikace zájmových osob může být kombinací obou těchto metod.

- První přistup - strukturální - rozpoznávání jednotlivých dominantních částí obličeje (oči, ústa, nos...) předkládaného vzoru, změření antropometrických veličin, jejich normalizace vzhledem k předpokládaným rušivým vlivům (šum, rušení, poloha ve scéně, velikost...), porovnání s databází známých fotografií použitím klasifikačních algoritmů, statistické rozhodnutí o relativní podobnosti s takto vybranou množinou obrazů.
- Druhý přístup - holistický - porovnání - identifikace vzorku pomocí globálních reprezentací opět s následným statistickým vyhodnocením relativní pravděpodobnosti.

Samotnou oblast identifikace lze rozdělit do několika procesů a aktivit:

1. lokalizace - detekce obličeje ve scéně předložené fotografie
2. zpracování ohraničeného prostoru obličeje v obraze
3. rozpoznání dominantních částí obličeje
4. zjišťování charakteristických a jedinečných vlastností obličeje
5. identifikace - porovnání se vzorem známých fotografií

1 DETEKCE HLAVY

Detekce hranice hlavy je zpravidla prvním krokem v automatizovaných systémech rozpoznávání, slouží k určení tzv. oblasti zájmu (většinou oči, rty a okolí). Metody nalezení hlavy je možné rozdělit na dva základní přístupy. Metody založené na vyhledávání významných bodů a metody založené na vyhledávání tváří. Metody založené na vyhledávání významných bodů se orientují na nalezení očí, nosních dírek, úst, obočí a dalších relativně odlišných bodů lidské hlavy. Podle jejich umístění potom určují oblast zájmu. Metody vyhledávání tváří prohledávají jednotlivé části obrazu a porovnávají tyto části se vzory obličejů a pozadí získaných ze vzorových obrazů nebo využívají charakteristické barvy lidské kůže, která je ve speciální barevné reprezentaci YC_bC_r velice podobná pro velkou množinu lidí [2]. Složitost úlohy závisí na osvětlení či pozadí scény. Pozadí může být buď zaplněno celkovou scénou nebo prázdné, jako např. u fotografií pasů, identikitu nebo trojdílných fotografií známých pachatelů, kde je zároveň zajištěna určitá standardizace pohledů na obličej (čelní, boční a šikmý pohled) se standardní expozicí a pozadím.

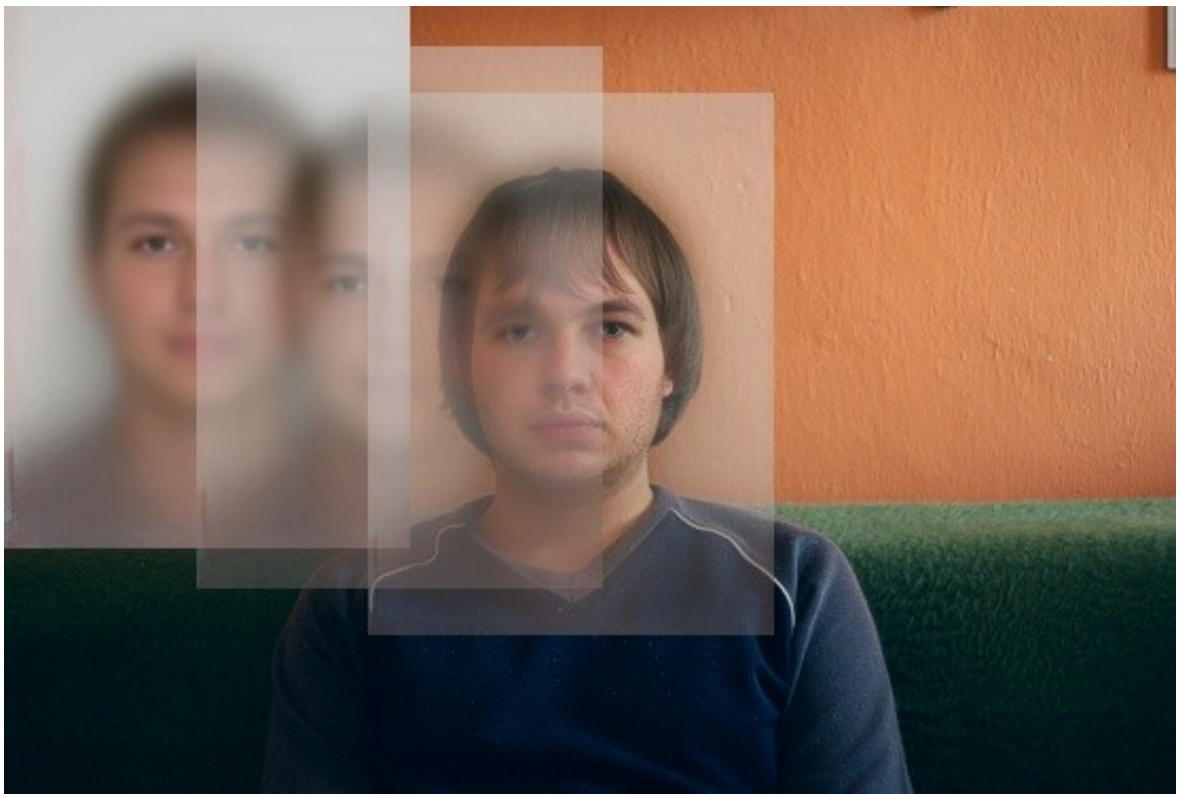
Detekce hlavy bývá zpravidla realizována ve dvouúrovňové detekční proceduře:

1. rozpoznání oblasti hlavy (a její orientace)
2. zvýraznění hranice obličeje.

Prvním krokem je hrubé rozpoznání možných kandidátů hlavy pomocí grafického vzoru nebo odstínů pleti. Poté následuje extrakce hranice možných tváří použitím aktivních hran.

1.1.1 Hledání pomocí vzorových obličejů

Grafickým vzorem může být sada vzorových obličejů, které jsou porovnávány se zkoumaným obrázkem. Méně náročnou metodou je použití šablony s „univerzálním“ obličejem, která postačí na základní rozpoznání zda se jedná o obličej či jiný objekt. Tato šablona se postupně posunuje po zkoumaném obrázku a hledá místo největší shody s pozadím. Pro dokonalejší vyhledávání je třeba k posunování šablony implementovat další afinní transformace – rotaci a změnu měřítka; tyto operace můžou u větších obrázků proces vyhledávání značně zpomalit.



Obr. 1. Lokalizace hlavy pomocí vzoru

1.1.2 Hledání pomocí barvy kůže

Metody detekce hlavy využívají podobnost barvy lidské kůže pro velkou množinu lidí v barevném prostoru označovaném jako YC_bC_r. Tento barevný prostor je převážně využíván v oblasti digitálního videa a to z důvodu jeho kompresních vlastností. Výhodou tohoto formátu je to, že informace o barvě je nesena pouze dvěma komponentami - C_b a C_r. Komponenta C_b určuje rozdíl mezi modrou složkou a referenční hodnotou, komponenta C_r pak obdobně určuje rozdíl mezi červenou složkou a referenční hodnotou. Komponenta Y obsahuje informaci o luminanci. Luminance je pojem zavedený skupinou CIE a udává sílu jasu, váhovanou funkcí spektrální citlivosti, která je charakteristická pro příslušný vizuální systém. Jasová složka je tedy reprezentována pouze jedinou komponentou Y. Informace o barvách je uložena ve zbývajících dvou komponentách, tedy ve složkách C_b a C_r.

Transformace ve smyslu z prostoru RGB do prostoru YC_bC_r je pak definována následujícím způsobem [3].

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \tag{1}$$

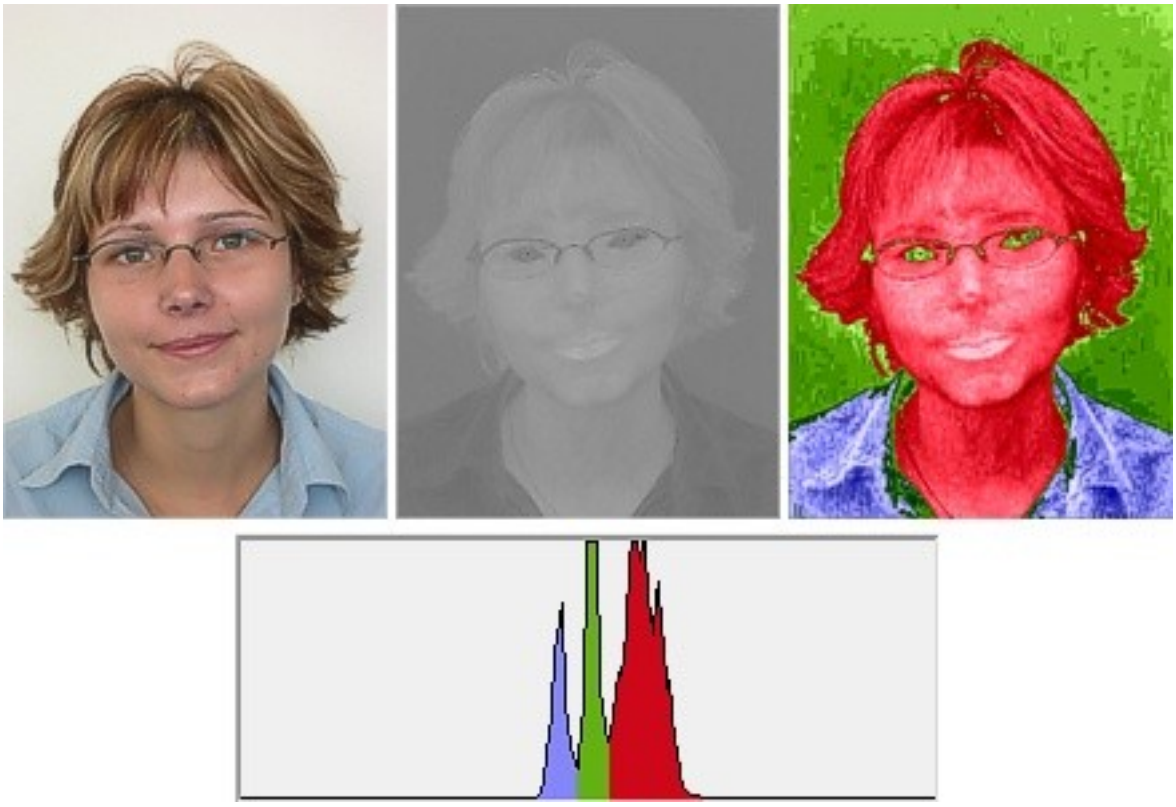
$$C_b = -0{,}1687 \cdot R - 0{,}3313 \cdot G + 0{,}5 \cdot B + 128 \tag{2}$$

$$C_r = 0{,}5 \cdot R - 0{,}4187 \cdot G - 0{,}0813 \cdot B + 128 \tag{3}$$

Obr. 2. Rozdělené kanály barevné fotografie

- *nahoře R(červený), G(zelený), B(modrý)*
- *dole Y(luminace), C_b(rozdíl modré složky), C_r(rozdíl červené složky)*

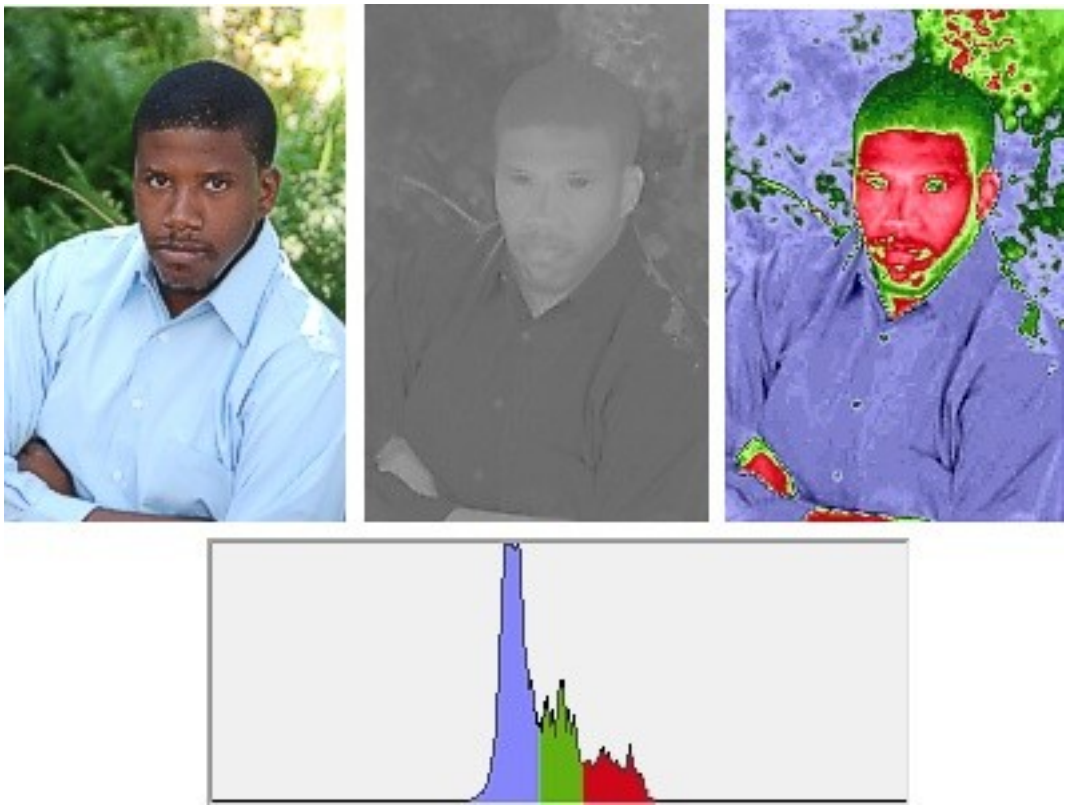
Rozdíl v červené složce mezi hlavou a okolím (pozadí, oblečení...) je poměrně velký, z histogramu je tento práh dobře určitelný.



Obr. 3. Histogram C_r kanálu u osoby se světlou pletí

- *nahoře barevná fotografie, její C_r kanál a zvýraznění částí podle histogramu*
- *dole histogram C_r kanálu s barevným vyznačením úrovní*

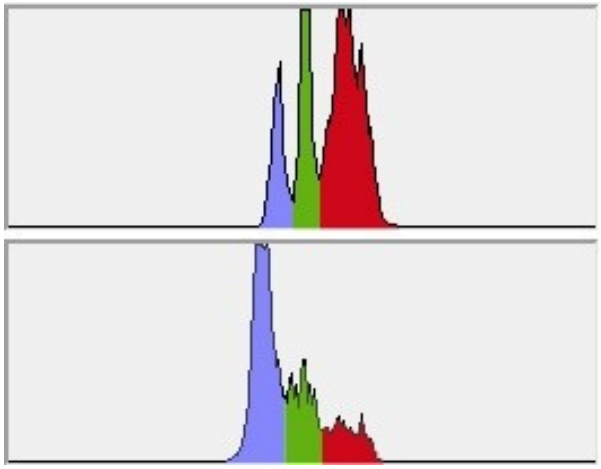
Metodu lze použít i u složitějších případů s různorodým pozadím nebo u osob s tmavou pletí, výrazněji červená složka představující oblast obličeje je stále dobře patrná.



Obr. 4. Histogram C_r kanálu u osoby s tmavou pletí a složitým pozadím

- *nahoře barevná fotografie, její C_r kanál a zvýraznění částí podle histogramu*
- *dole histogram C_r kanálu s barevným vyznačením úrovní*

Pokud porovnáme oba histogramy, je patrné, že velká část hlavy (především obličej) u osoby tmavé pleti má stejný rozsah jako hlava osoby se světlou pletí.



Obr. 5. Porovnání histogramů

- *nahoře osoba se světlou pletí*
- *dole osoba s tmavou pletí*

1.2 Detekce hranice obličeje

Obličej lze detekovat při apriorní znalosti jeho tvaru např. metodou „hadů“ (snakes), neboli metodou Aktivních kontur (Active Contours) [4]. Jde o postupné tvarování kontur až ke hraně objektu v obraze. Řízená uzavřená kontura se deformuje vlivem tzv. vnitřních, obrazových a vnějších sil. Vnitřní síly kontrolují hladkost průběhu, obrazové síly směrují tvarování kontury směrem ke hraně objektu a vnější síly jsou výsledkem počátečního umístění kontury. Obecně hovoříme o implementaci tzv. hada na jednotlivé snímky.

Jde o výpočet vnitřní a vnější energie „hada“ a nalezení jeho pozice s minimální celkovou energií. Konečný výběr „hadů“ tedy odpovídá lokálnímu minimu energetické funkce. Energetická funkce je definována:

$$\int_0^1 E_{snake}(v(s))\,ds = \int_0^1 [E_{internal}(v(s)) + E_{image}(v(s)) + E_{constrain}(v(s))]\,ds \qquad (4)$$

kde *v(s)* je parametrická křivka, *s* je délka křivky. $E_{internal}$ je vnitřní energie křivky způsobená zakřivením jako její první a druhé derivace.

$$E_{internal} = \alpha(t)\frac{\partial v(t)}{\partial t} + \beta(t)\frac{\partial^2 v(t)}{\partial^2 t} \tag{5}$$

Váhové funkce *α(t)* a *β(t)* definují hladkost a pružnost kontury. Mají vliv na tvar kontury např. v ostrých zlomech, je tak možné regulovat přilínání na hrany v obraze nebo naopak ladný tvar křivky. Vnější energie E_{image} se skládá z více složek. Složky E_{line}, E_{edge} a E_{term} kvantifikují matematicky snadno popsatelné detaily obrazu.

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{6}$$

E_{line} určuje přitažlivost kontury k světlejším nebo tmavším částem obrazu podle váhového faktoru w_{line}. Efektní je zejména v případě tenké hrany ve stejně barevné oblasti.

$$E_{line} = f(v(t)) \tag{7}$$

E_{edge} přitahuje kontury k hranám, tzn. k místům s vysokou hodnotou velikosti gradientu. Protože místo největšího gradientu (energetického minima) může být zašuměné, méně výrazné nebo příliš vzdálené od počáteční polohy kontury, je vhodné pro lepší konvergenci před samotným výpočtem obraz rozmazat nebo aplikovat speciální hranový filtr.

$$E_{edge} = -|\nabla f(v(t))|^2 \tag{8}$$

Nejsložitější částí je složka E_{term}, která má za úkol detekovat ostré rohy a konce hran pomocí zkoumáním křivosti. Kvůli předpokládanému šumu se křivost počítá na rozmazaném obraze. Jestliže kontura kopíruje rovnou nebo hladkou hranu, je celková hodnota E_{term} malá. Teoreticky lze zahrnout do E_{image} další libovolné části podle požadavků řešené úlohy.

$E_{constrain}$ reprezentuje vnější omezení jako aproximaci křivky její druhou derivací. Tento složitý algoritmus je vhodné pro zrychlení výpočtu zoptimalizovat dynamickými programovacími numerickými metodami.

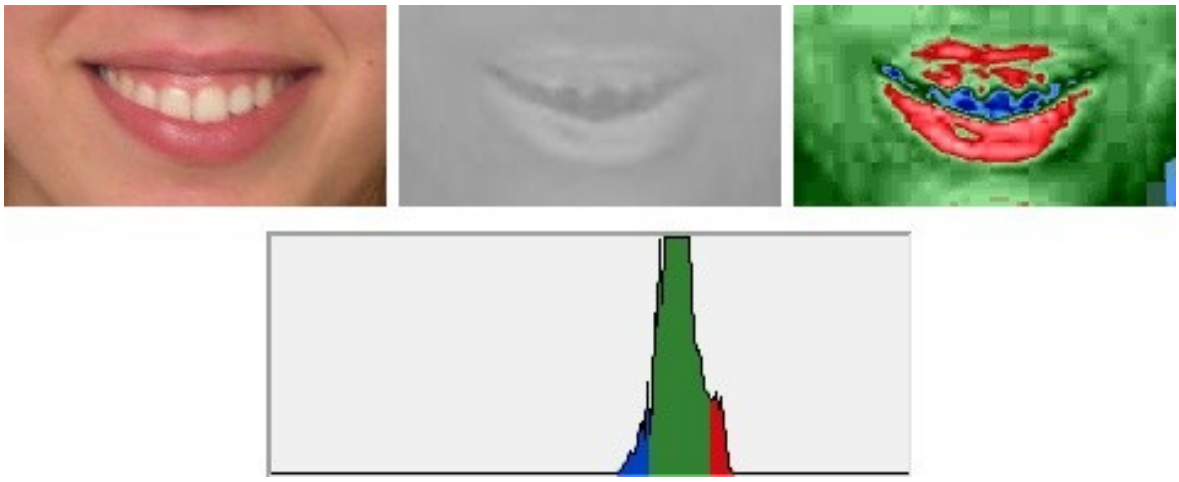
1.3 Detekce očí

Obrysy očí a rtů jsou určovány pomocí poměrného umístění vzhledem k hranicím hlavy. Oči mají obecně stabilní strukturu a tvar skládající se z duhovky a víčka. Tento fakt nabízí možnost jejich hledání pomocí pevného vzoru (šablony), podobně jako u modelu hlavy. Pomocí rastru obličeje se provede prohledávací fáze v celém obrazu.

1.4 Detekce úst (rtů)

Ústa mají velmi poddajnou formu, která určuje emoční vyjádření jednotlivce. Proto je pro vygenerování modelu úst zpravidla použit deformační model s hierarchickým adaptivním algoritmem. Běžné detektory hran nejsou schopny nalézt hrany takových přirozených útvarů, jako jsou např. ústa. Deformační modely jsou pro takové úlohy vhodné, protože mohou být specifikovány nastavením parametrů z apriorní znalosti tvaru objektu. Globální informace lokálních hran může být uspořádána do globálního vjemu, který spolehlivě určí umístění obrysu. Podle pozice očí je poměrně snadné určit přibližnou oblast úst. V této oblasti se stejným způsobem jako při zvýraznění ohraničení obličeje implementuje metoda Aktivních kontur.

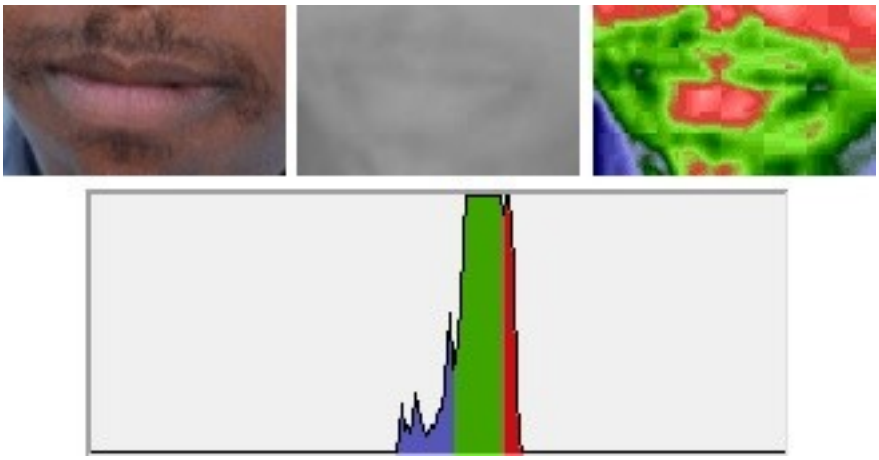
Podobně jako u detekce celého obličeje, lze i u hledání rtů využít výrazné barevné odlišnosti rtů od červeně méně výrazné pleťové barvy okolí [5]. Rovněž je nutné podle očí určit přibližnou oblast úst; v této oblasti se pak nacházejí jen dva barevně dominantní objekty: kůže a ústa. Těmto objektům odpovídá i rozložení histogramu, kde je nutné automaticky nalézt práh, oddělující oba objekty od sebe.



Obr. 6. Histogram oblasti úst u osoby se světlou pletí

- *nahoře barevná fotografie, její C_r kanál a zvýraznění částí podle histogramu*
- *dole histogram C_r kanálu s barevným vyznačením úrovní*

U člověka světlé pletí je ohraničení rtů velmi dobře patrné; problém může nastat u lidí s tmavší pletí, kdy barevný rozdíl rtů a okolní pleti není tak výrazný a v histogramu jsou rty rozloženy rovnoměrněji, nemusí být tedy práh dobře určitelný a ohraničení rtů bude jen částečné.



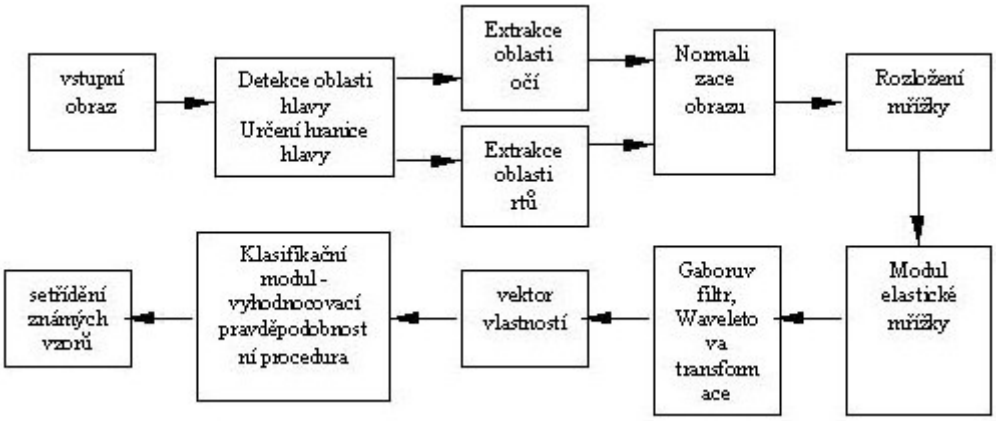
Obr. 7. Histogram oblasti úst u osoby s tmavou pletí

- *nahoře barevná fotografie, její C_r kanál a zvýraznění částí podle histogramu*
- *dole histogram C_r kanálu s barevným vyznačením úrovní*

2 EXTRAKCE VEKTORU RYSŮ (VLASTNOSTÍ)

Při identifikace osob metodou globálních reprezentací se využívá elastické mřížky prostorově proložené oblastí obličeje podle bodů zájmu obličeje (oči, ústa, nos, uši). V těchto metodách se stále více využívá tzv. waveletová transformace (vychází z Fourierovy transformace, digitalizovaný obraz je chápán jako signál ve dvourozměrném prostoru) k normalizaci obrazu ve smyslu fyzikálních vlastností fotometrických veličin obrazu (jas, kontrast, šum v obraze) a Gaboroův filtr pro analýzu okolí bodů proložené elastické mřížky. Jako jednoznačná reprezentace obrazu (jako výsledek waveletové transformace zpravidla aplikované na celý obraz k normalizaci obrazu a Gaborova filtru aplikovaného na každý bod elastické mřížky) se obdrží tzv. vektor vlastností, což je binární číslo reprezentující jedinečné vlastnosti zkoumaného obličeje.

Tento vektor vlastností se předloží klasifikačnímu pravděpodobnostnímu algoritmu k porovnání s množinou známých osob a vyhodnocení a setřídění podobnosti této množiny [1].



Obr. 8. Blokový diagram obličejové analýzy

Další metodou pro vyhodnocování a klasifikaci obrazu je použití neuronových sítí.

3 ROZPOZNÁVÁNÍ POMOCÍ NEURONOVÉ SÍTĚ

Neuronová síť zpravidla slouží k vyhodnocování a porovnávání hodnot získaných předzpracováním obrazů. Specifikace neuronových sítí se dá popsat takto:

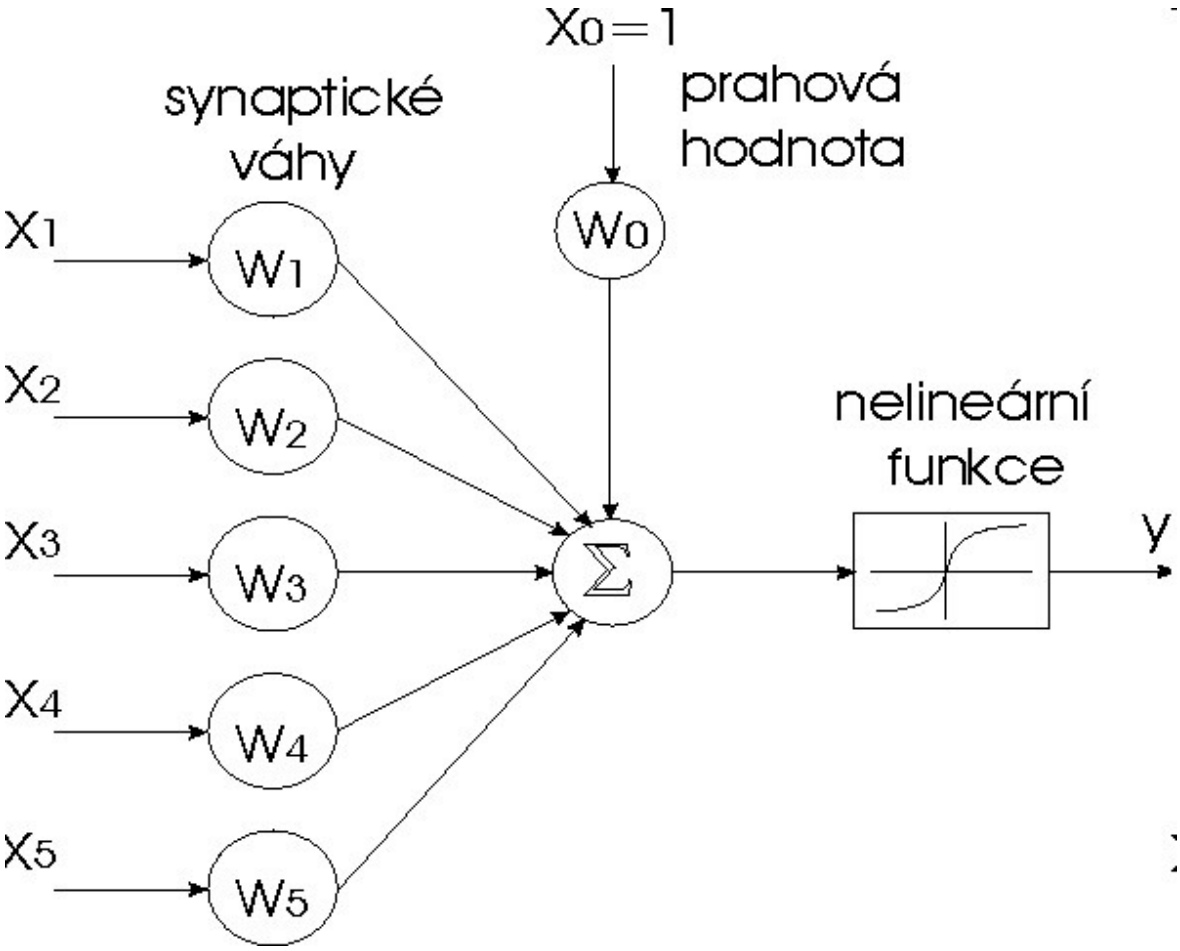
- Neuronové sítě vycházejí z principů biologické nervové sítě.
- Využívají distribuované, paralelní zpracování informace při provádění výpočtů.
- Znalosti do zvolené topologie jsou ukládány prostřednictvím síly vazeb mezi jednotlivými neurony.
- Učení je základní a podstatná vlastnost neuronové sítě.

Topologie umělé neuronové sítě je zpravidla zvolena podle úlohy, kterou má řešit. Pro rozpoznávání a analýzu obrazu se zpravidla používají Hopfieldovy sítě s dopředným učením, třívrstvé back propagation sítě se zpětným učením; je možné použít i Neocognitron sítě, Adaline sítě a další.

3.1 Neuron

Umělé neuronové sítě byly vytvořeny na základě jednoduchých modelů neuronů - funkčních buněk nervového systému živých organizmů. První matematický model neuronu vytvořili McCulloch a Pitts v roce 1943 a tento model se dodnes používá pro běžné aplikace. Tento matematický model se skládá ze tří hlavních částí. Obsahuje vstupní, výstupní a funkční část. Vstupní část se skládá ze vstupů a z přiřazených, nastavitelných vah (synaptické váhy). Na základě váhových koeficientů mohou být jednotlivé vstupy zvýhodňovány či potlačeny. Následující částí je výkonná jednotka, která zpracuje informace ze vstupu a vygeneruje výstupní odezvu. Třetí část je výstupní jednotka, která přivádí výstupní informace na vstup jiných neuronů. Z toho je patrná podoba mezi klasickými výpočetními systémy a umělými neurony. Oba systémy obsahují vstupní část, paměť, výkonnou jednotku a výstupní část. Velké rozdíly jsou ovšem v uspořádání těchto částí. Paměť umělého neuronu není samostatná jednotka, ale je rozprostřená ve vstupní části formou váhových koeficientů. Pomocí těchto koeficientů je systém schopný zapamatovat si informace. Jak je vidět na obrázku, výkonná jednotka

umělého neuronu je mnohem jednodušší než výkonná jednotka výpočetních systémů a je tvořena jednoduchou nelineární funkcí [6].



Obr. 9. Jednoduchý model neuronu

Vstupní hodnoty jsou vynásobeny příslušnými váhovými koeficienty a sečtou se. Na výsledek součtu se aplikuje funkce (obecně nelineární) a výsledná hodnota funkce je přivedena na vstup jiných neuronů pomocí výstupní části. Neuron má ještě jeden zvláštní vstup, který není připojený k výstupu žádného neuronu, ale přivádí konstantní veličinu do neuronu. Tato veličina funguje jako prahová hodnota při aktivování výstupu. Když suma váženého součtu vstupů nepřesahuje prahovou hodnotu, tak se neuron neaktivuje a jeho výstup zůstane nezměněný.

Matematicky lze funkci neuronu popsat následovně:

$$y = F\left(\sum_{i=1}^{n} x_i w_i + \Theta\right) \tag{9}$$

kde:

x_i - je hodnota na i-tém vstupu,

w_i - je váha i-tého vstupu,

Θ - je prahová hodnota,

n - je celkový počet vstupů,

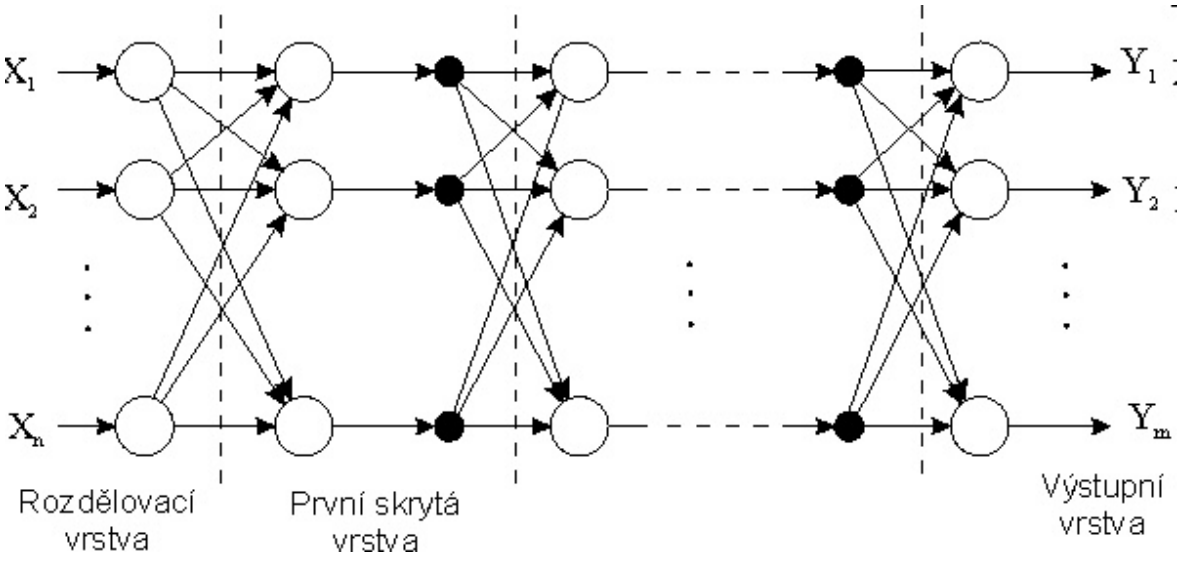
F - je obecná nelineární funkce,

y - je hodnota výstupu.

3.2 Neuronová síť

Jediný neuron není schopen vykonat příliš složitou funkci. Síla systému, využívající umělé neurony, je ve struktuře, v síti velkého počtu neuronů. Umělá neuronová síť je vlastně pole jednoduchých výkonných prvků - neuronů. Takovéto uspořádání má velkou flexibilitu a spolehlivost. Umožňuje různě propojovat vstupy a výstupy neuronů, zvýhodnit či potlačit některé vstupy a minimalizovat vliv nesprávně fungujícího neuronu na celkový výsledek. Samozřejmě i tento systém má nevýhody. Největší problémy se vyskytují při realizaci velmi složitých struktur, kde velký počet propojení mezi neurony se realizuje velmi obtížně. Dalším problémem je to, že neexistuje jednoznačný postup při syntéze složitějších struktur [6].

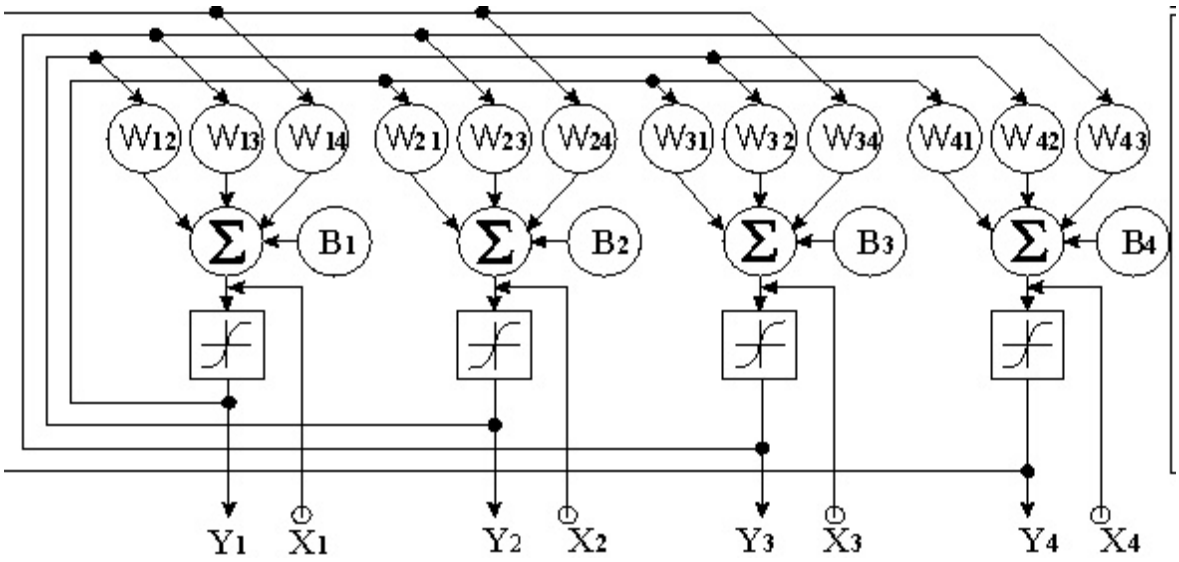
Neurony jsou většinou sdružovány do vrstev. Výstupy z n-té vrstvy jsou přivedeny na vstup obecně každého neuronu ve vrstvě n+1. První vrstva se nazývá vstupní či rozdělovací vrstva a má za úkol přijímat hodnoty z okolí pro zpracování a přivést je na vstup každého neuronu následující vrstvy. Poslední vrstva nese název výstupní a hodnoty na jejím výstupu jsou odezvou celého systému na vstupní vzorky. Vnitřní vrstvy se nazývají skryté vrstvy. Jejich počet závisí na složitosti funkce, kterou má síť vykonat a na zvoleném typu sítě.



Obr. 10. Vrstvová struktura umělé neuronové sítě

3.2.1 Hopfieldova neuronová síť

Hopfieldova síť představuje jednovrstvou síť, ve které jsou neurony propojeny způsobem „každý s každým, kromě sebe sama“ [7].



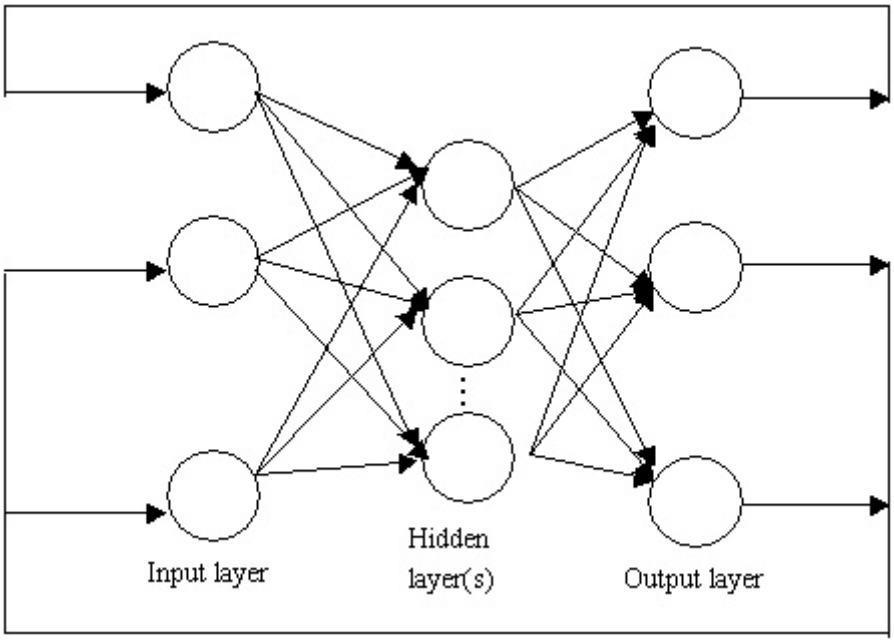
Obr. 11. Hopfieldova síť

Hopfieldova síť obsahuje tolik neuronů, kolik je vstupů resp. výstupů neuronové sítě. Přitom každý neuron je zároveň vstupním i výstupním neuronem. Výstup každého

neuronu je přes váhy spojení opětovně přiváděn na vstupy ostatních neuronů, čímž vzniká uzavřená smyčka (zpětná vazba). Hopfieldovu síť tedy řadíme do skupiny rekurentních (zpětnovazebních) neuronových sítí. Každý neuron přijímá jak vstupní signály, tak i interní výstupní signály od ostatních neuronů. Během učení dochází k iterativnímu procesu. Jedná se o případ učení bez učitele, při kterém se využívá pouze reakce na předkládané vstupy. Vstupní vektor způsobí reakci na výstupech sítě a hodnoty výstupů se ihned přivádějí jako vstupy sítě. Tento proces probíhá až do stavu, kdy jsou výstupní hodnoty identické s hodnotami vstupními. Změny synaptických vah mezi jednotlivými neurony probíhají dle Hebbova algoritmu učení. Hopfieldovy sítě dělíme na binární (s nelineární aktivační funkcí) a na spojité (např. se spojitou sigmoidální aktivační funkcí). V současné době existuje velké množství modifikací této sítě.

3.2.2 Třívrstvá síť s učením „Back Propagation“

Na rozdíl od sítí Hopfieldových jsou váhy ve vícevrstvých sítích jednosměrné a vedou ze vstupní vrstvy do skryté a ze skryté vrstvy do výstupní. Trénování sítě probíhá tak, že vytvoříme soubor trénovacích dat a síť tento soubor několikrát po sobě zpracovává. Trénovací soubor se skládá z dvojic vstupních a výstupních vzorů. Výstupní vzor může být považován za kategorii, do které má být správně zařazen vstupní vzor. Síť přijme na vstupní vrstvě vstupní vzor, ten sítí projde přes skrytou vrstvu na výstupní, na které se objeví jako výstupní vzor. Tento výstup je porovnán se správným výstupním vzorem načteným z trénovacího souboru. Informace o odchylce je sítí šířena směrem od výstupní vrstvy k vrstvě vstupní a jsou podle ní upravovány váhy sítě tak, aby až síť bude stejnou dvojici vzorů číst příště, byla odchylka na její výstupní vrstvě menší. Poté, co se síť správně naučí trénovací soubor, může být testována na jiném souboru dvojic vstupních a výstupních vzorů, abychom zjistili, jak dobře se síť naučila zobecňovat. U některých typů dat není výhodné, aby se síť naučila trénovací soubor co nejpřesněji, protože potom si pamatuje trénovací data příliš specificky a to jí brání v zobecňování neznámých dat [8].



Obr. 12. Síť s učením „Back Propagation“

II. PRAKTICKÁ ČÁST

LOKALIZACE OBLIČEJŮ NA SKUPINOVÉ FOTOGRAFII

Úkolem programu je rozpoznání obličejů na skupinové fotografii. Vstupní data jsou tedy reprezentována obrázkem (v jpg formátu) vybraným uživatelem. Výstupem by měly být výřezy z tohoto obrázku, kde se nacházejí jednotlivé obličeje, uloženy v samostatných souborech (obrázcích jpg). Pro platformní nezávislost je pro tvorbu programu zvolen objektově orientovaný jazyk Java. Z teoretické části uplatníme kombinaci metod hledání podle odstínů barvy kůže a hledání pomocí vzorových obličejů. Lokalizace obličejů by měla co nejpřesněji zohlednit různou velikost obličejů na fotografiích a také různé rasové odlišnosti. Základem přesné lokalizace ovšem je co nejlépe exponovaná fotografie.

4 PŘÍPRAVA DAT PRO DETEKCI HLAVY

Prvním krokem k lokalizaci obličeje je vytvoření vzoru, který bude použitý pro porovnání s obličeji na zkoumané fotografii. Vzor vytvoříme určením průměrného obličeje z dostatečně velké skupiny portrétů osob. Má-li být určení vzoru co nejpřesnější i nejjednodušší, je vhodné aby všechny portréty osob byly pořízeny při stejném osvětlení, ve stejné vzdálenosti a pokud možno s jednotným jednobarevným pozadím. Klasickým příkladem takových vhodných portrétů jsou pasové fotografie. I v tomto případě se však vytvoření vzoru bohužel neobejde bez zásahu uživatele. Aby byl vzor smysluplný, je potřeba všechny portréty nějakým způsobem zarovnat na sebe. V našem případě jsme zvolili zarovnání podle očí. Obě oči tedy musí být u všech portrétů na stejné pozici, tj. ve stejné vzdálenosti od okrajů, stejně vzdálené od sebe a vodorovně vyrovnány. Detekování pozice očí musí být v tomto případě provedeno ručně. Program postupně otvírá obrázkové soubory s portréty osob a obsluha označuje levé a pravé oko. Výsledné pozice horizontální (x) a vertikální (y) osy jsou uloženy v souboru společně s číslem souboru.

Skupinu portrétů jsme získali z fotografií studentů FT a FAI UTB ve Zlíně, které byly pořízeny studijním oddělením pro studentské průkazy při zápisu studentů v 1. ročníku v letech 2004, 2005 a 2007. Celkově se jedná o 3061 portrétů, které poměrně dobře splňují parametry dokladové fotografie. Zároveň jde o velmi různorodou skupinu, která se neskládá pouze ze studentů denního studia, ale velké zastoupení mají i studenti kombinovaného studia, částečně i zahraniční studenti. Vzorek tedy není tvořen z profilově úzké skupiny lidí. Fotografie jsou použity se souhlasem studijního oddělení UTB ve Zlíně.

Obr. 13. Studenti FAI a FT UTB ve Zlíně

5 URČOVÁNÍ VZORU

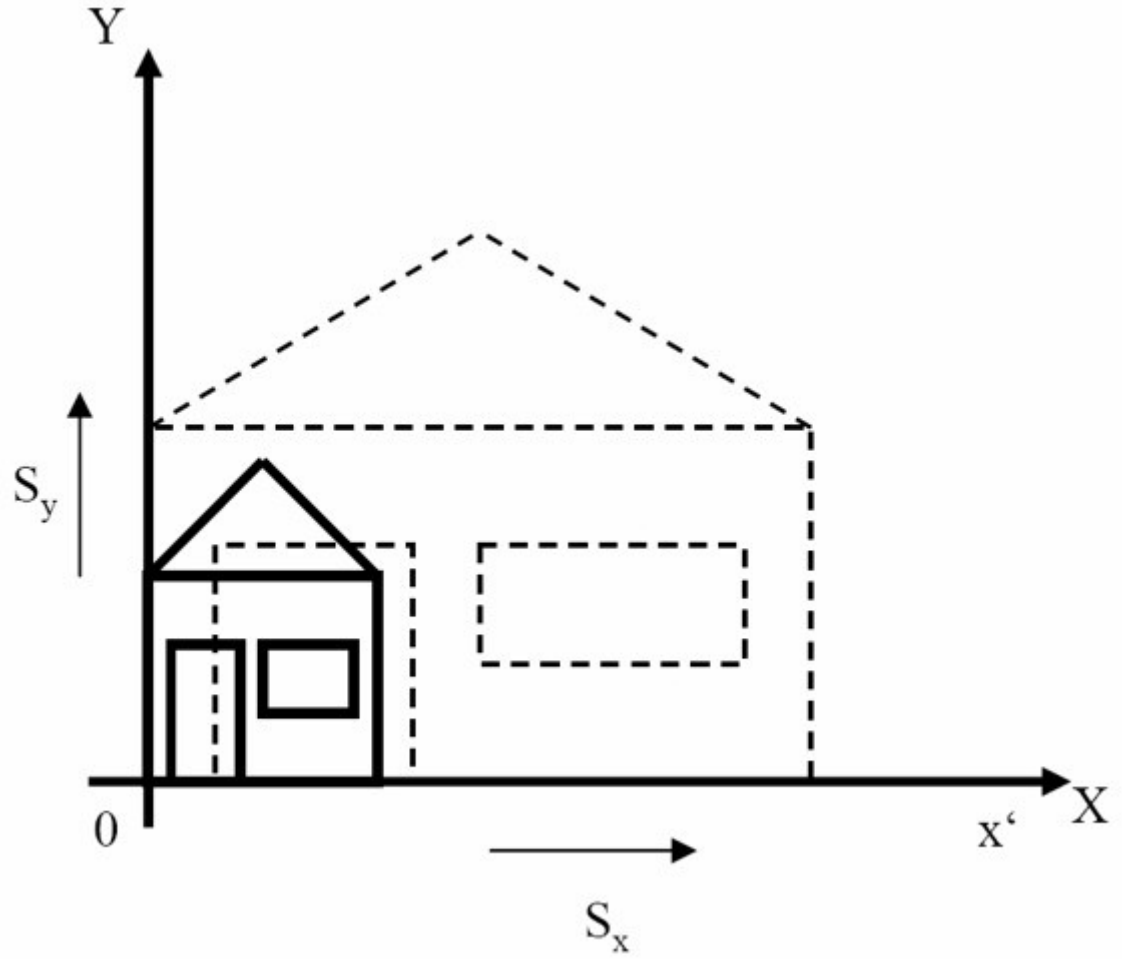
Po získání souřadnic očí u všech portrétů je potřeba upravit fotografie podle těchto souřadnic do jednotné podoby. Transformace obrázku musí přemístit oči všech osob na stejné souřadnice, přičemž souřadnice osy y jsou u každé osoby jednotné pro obě oči (oči jsou v jedné rovině). Takto upravené portréty zarovnáme podle výsledných souřadnic očí na sebe a určíme průměrný obličej, který použijeme při rozpoznávání obličejů jako vzor.

5.1 Afinní transformace

K určení výsledných souřadnic očí je nutné použít tři základní afinní transformace – **změna měřítka** (scale), **rotace** (rotation) a **posunutí** (translation). Každá tato operace je definovaná příslušnou transformační maticí (ve dvourozměrném prostoru maticí 3x3), která se násobí maticí se souřadnicemi jednotlivých bodů obrázku.

5.1.1 Změna měřítka (scale)

Změna měřítka je změnou velikosti objektu ve směru souřadnicových os. Pokud je absolutní hodnota koeficientu měřítkování v intervalu (0, 1), dochází ke zmenšení transformovaného objektu. Je-li absolutní hodnota koeficientu větší než jedna, dojde k prodloužení, je-li znaménko záporné, dochází k prodloužení či zmenšení v opačném směru (převrácení).



Obr. 14. Změna měřítka obrázku

Rovnice pro změnu měřítka bodu P mají tvar:

$$X' = S_X \cdot X \tag{10}$$

$$Y' = S_Y \cdot Y \tag{11}$$

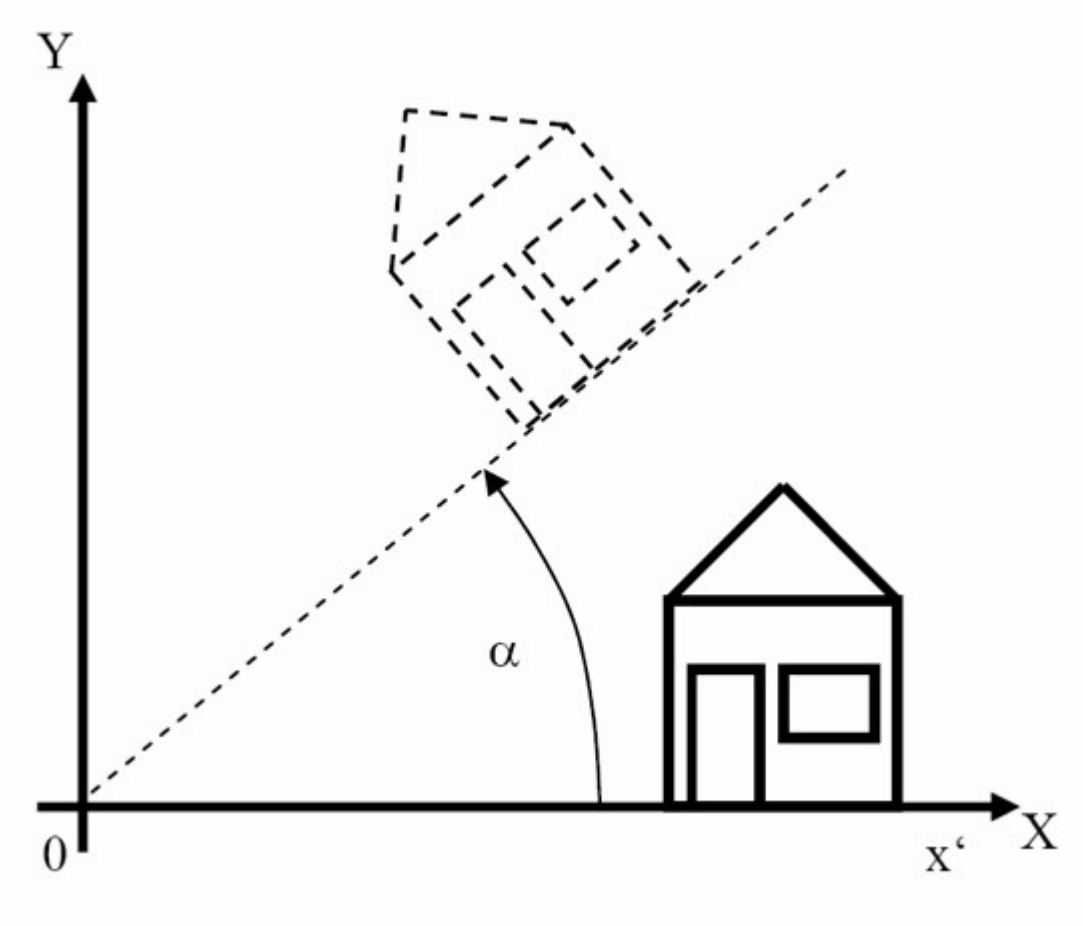
Kde S_x je koeficient změny měřítka ve směru souřadnicové osy x a S_y je koeficient změny měřítka ve směru souřadnicové osy y.

Odpovídající transformační matice má pro homogenní souřadnice tvar:

$$A_S = \begin{bmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

5.1.2 Rotace (rotation)

Rotace (otočení) je dána tím, že spojnice všech bodů s pevně zvoleným bodem, tzn. středem otočení, se změní o stejný úhel, tzv. úhel otočení a vzdálenost bodů od středu otáčení zůstává nezměněna.



Obr. 15. Rotace obrázku

Otáčení bodu P kolem počátku soustavy souřadnic O=[0,0] o orientovaný úhel α získáme bod P' o souřadnicích:

$$X' = X \cdot \cos\alpha - Y \cdot \sin\alpha \tag{13}$$

$$Y' = X \cdot \sin\alpha + Y \cdot \cos\alpha \tag{14}$$

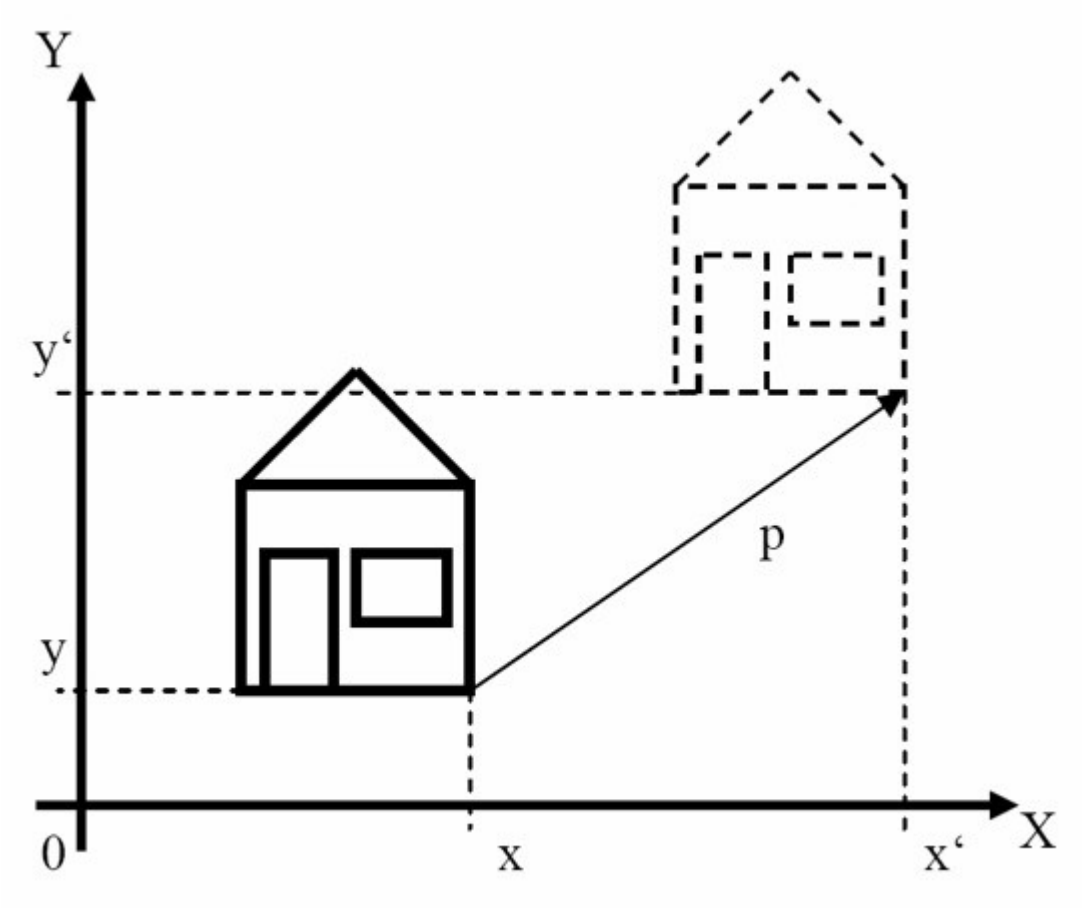
Pro rotaci kolem libovolného bodu je potřeba posunout tento bod do počátku, provést rotaci a inverzní maticí pro posunutí bod přesunout zpět (viz. 5.1.4).

Maticové vyjádření transformace otáčení má pro homogenní souřadnice tvar:

$$A_R = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

5.1.3 Posunutí (translation)

Posunutí je charakterizováno tím, že všechny body transformované množiny bodů změní své kartézské souřadnice o stejnou hodnotu. Všechny body dané množiny jsou tedy posunuty stejným směrem o stejnou vzdálenost.



Obr. 16. Posunutí obrázku

Směr a velikost tohoto posunutí lze charakterizovat vektorem posunutí.

$$\vec{p}=(X_T, Y_T)=(X'-X, Y'-Y) \tag{16}$$

Aplikací této transformace na bod P získáme bod P' o souřadnicích:

$$X'=X+X_T \tag{17}$$

$$Y'=Y+Y_T \tag{18}$$

Maticové vyjádření transformace posunutí má pro homogenní souřadnice tvar:

$$A_M=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X_T & Y_T & 1 \end{bmatrix} \tag{19}$$

5.1.4 Skládání transformací

Při postupném aplikování transformací na bod P záleží na pořadí, v jakém se transformace provádějí. Je rozdíl, jestliže bod posuneme a poté otočíme okolo počátku souřadnicového systému nebo zda bod nejprve otočíme a poté provedeme transformaci posunutí. Transformaci vzniklou složením z více transformací lze vyjádřit jedinou maticí, kterou získáme postupným násobením matic, reprezentujících dílčí transformace. Protože záleží na pořadí transformací, záleží i na pořadí násobení matic. Matice se násobí zprava.

$$A=A_S \cdot A_R \cdot A_M \tag{20}$$

5.1.5 Implementace v Javě

V jazyce Java je pro afinní transformace vytvořena speciální třída *java.awt.geom.AffineTransform*, která má v sobě implementovány výše uvedené transformace. Transformace se aplikují na obrázek načtený do paměti pomocí třídy *java.awt.image.BufferedImage*, která zajišťuje načtení a uchování obrázku v paměti. Ke každému nahranému obrázku jsou ze souboru načteny souřadnice očí, uložené předchozím programem. Z těchto souřadnic je podle pravidel pravoúhlého trojúhelníku určena vzdálenost očí od sebe, ze které je určena změna měřítka obrázku a úhel, o který je

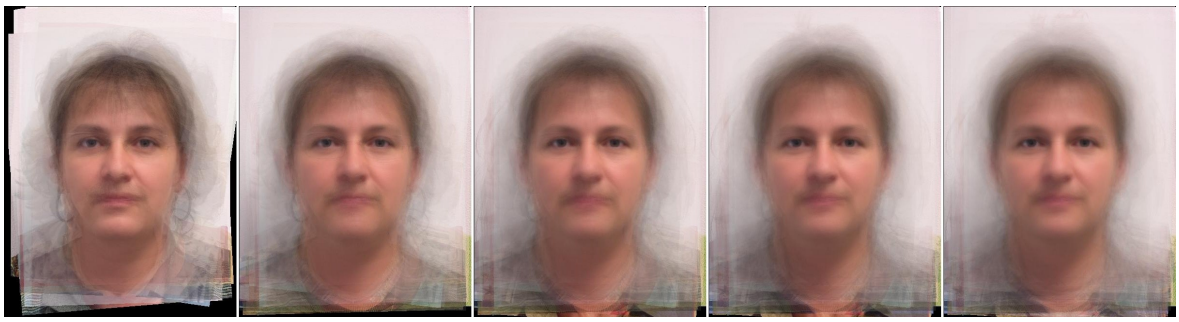
pravé oko vychýleno od roviny levého, z čehož plyne úhel rotace. Změna měřítka probíhá voláním funkce *scale* z uvedené třídy afinních transformací. Parametry této funkce jsou přímo hodnoty transformační matice S_X a S_Y (viz. 5.1.1). Rotace je uskutečněna voláním funkce rotate, jejíž parametry jsou úhel a středový bod rotace, v našem případě souřadnice levého oka (není tedy nutné zvlášť posouvat bod do počátku souřadnic a zpět). Následuje už jen posunutí do zvoleného bodu jednotného pro všechny transformované obrázky. Protože pro průměrování obličejů je nutné jednotlivé obrázky načíst do pole (viz. 5.2), je posunutí realizováno až v rámci tohoto pole.

Obr. 17. Aplikace výpočtu souřadnic a afinních transformací

5.2 Průměrování obličejů

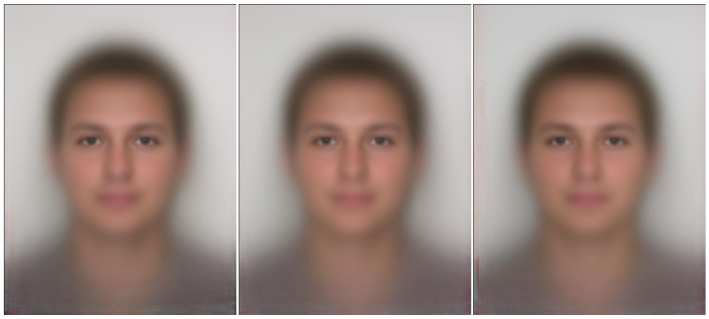
Výsledný vzorový obličej určíme zprůměrováním všech transformovaných obrázků. Je vytvořeno trojrozměrné pole o velikosti rovnající se rozměrům obrázku (všechny obrázky mají stejné rozměry) a tříhodnotové hloubky vycházející z barevného prostoru RGB. Každý pixel (jednotlivý bod obrázku) je v RGB prostoru tvořen třemi barvami – červená (red), zelená (green) a modrá (blue). Pomocí funkce *getRGB* třídy *java.awt.image.BufferedImage* jsou jednotlivé pixely rozebrány na tři barevné složky, které jsou jednotlivě přičteny do pozic pole. Protože při transformacích obrázků se vyskytují oblasti, ve kterých se skutečný obrázek nenachází, doplňuje tyto tři pozice barev ještě čtvrtá, která obsahuje počet uložených obrázků obsahujících tento pixel. Po přičtení hodnoty jednotlivých barevných složek každého pixelu všech obrázků jsou jednotlivé složky vyděleny počtem obsažených obrázků. Výsledné pole je uloženo do souboru, ale také převedeno zpět na hodnotu RGB pomocí funkce *setRGB* a uloženo jako obrázek ve formátu *jpg*.

Obr. 18. Prvních pět kroků počítání průměrného obličeje

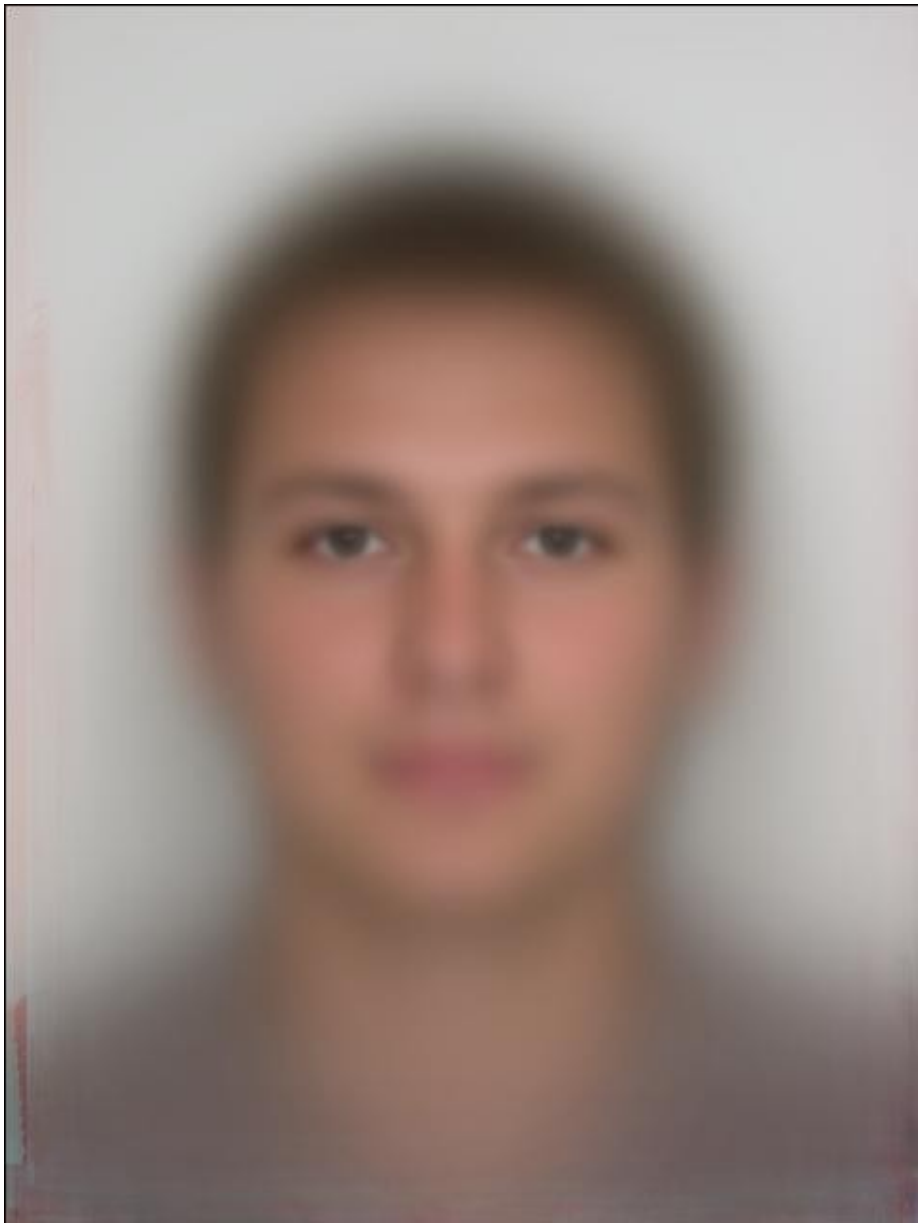


Obr. 19. Desátý, dvacátý, třicátý, čtyřicátý a padesátý krok počítání průměrného obličeje

Obr. 20. Stý, dvoustý, třístý, čtyřstý a pětistý krok počítání průměrného obličeje



Obr. 21. Tisící, dvoutisící a třítisící krok počítání průměrného obličeje



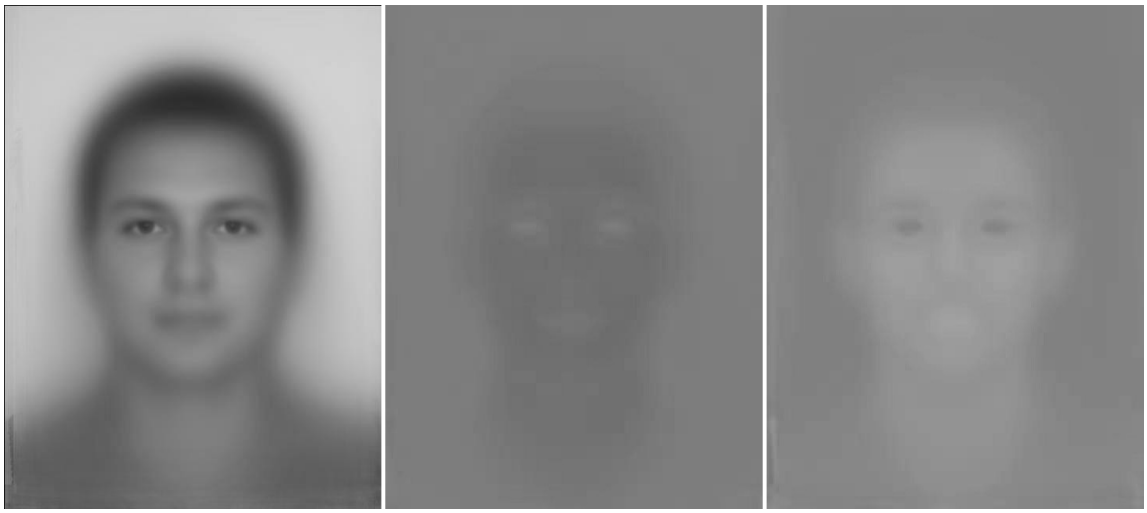
Obr. 22. Výsledný průměrný obličej (určeno z 3061 osob)

6 VYHLEDÁVÁNÍ OBLIČEJŮ NA FOTOGRAFII

Pro vyhledání obličejů na fotografii použijeme kombinaci metod s použitím vzoru a s využitím barevného odstínu kůže. Pro další zpracování je opět nutné stejným způsobem jako při průměrování obličejů (viz. 5.2) načíst do jednotlivých polí vzorový obličej a uživatelem vybranou fotografii se skupinou osob.

6.1 Převedení do prostoru YC_bC_r

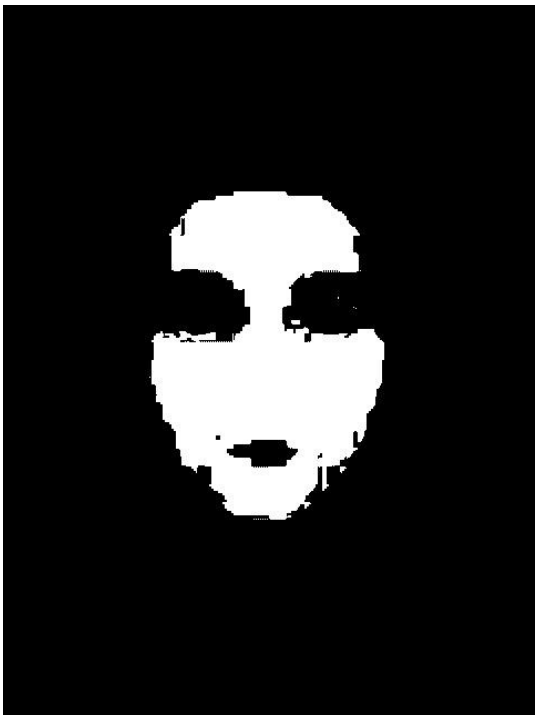
Tyto dva obrázky rozdělené v polích podle barevného prostoru RGB je potřeba převést do barevného prostoru YC_bC_r (viz. 1.1.2) a vložit do nových polí. Pro určení oblasti obličeje postačují pouze kanály C_b (reprezentující modrou barvu) a C_r (červená barva), jasová složka Y není využita a není ji tedy nutné počítat.



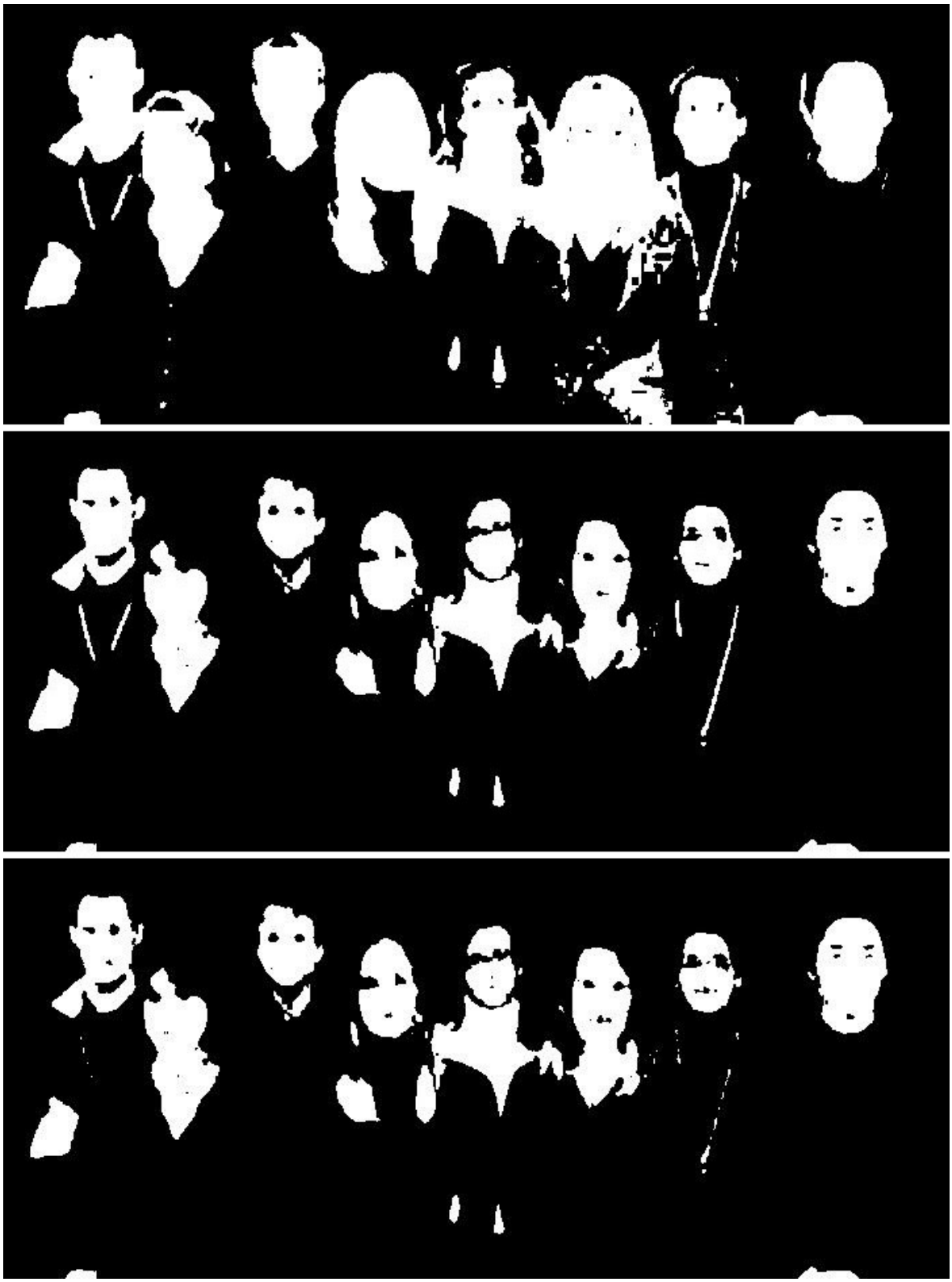
Obr. 23. Vzorový obličej rozložený do kanálů barevného prostoru YC_bC_r

Obr. 24. Skupinová fotografie rozdělená do kanálů barevného prostoru YC_bC_r

Z obrázků je patrné jak jsou v obou kanálech výrazné obličeje osob. U kanálu C_b jsou obličeje, které mají malé zastoupení modré složky výrazně tmavé, u kanálu C_r jsou díky převažující červené barvě výrazně světlé. Intervalem hodnot z obou kanálů (C_b i C_r) jsou automaticky vybrány oblasti ve kterých se pravděpodobně nachází barva lidské kůže. Interval musí být vhodně zvolený, aby eliminoval nejen místa s malým zastoupením červené barvy, ale naopak i místa s příliš velkým podílem červené (červené oblečení). Zároveň ale musí zohlednit co největší škálu barev pleti. Z hodnot ležících v intervalu jsou pro skupinovou fotografii i pro vzorový obličej vytvořeny binární obrazy.



Obr. 25. Binární obraz pleťových barev vzorového obličeje



Obr. 26. Binární obraz pleťových barev u skupinové fotografie – postupné snižování tolerance pro označení odstínu barvy kůže

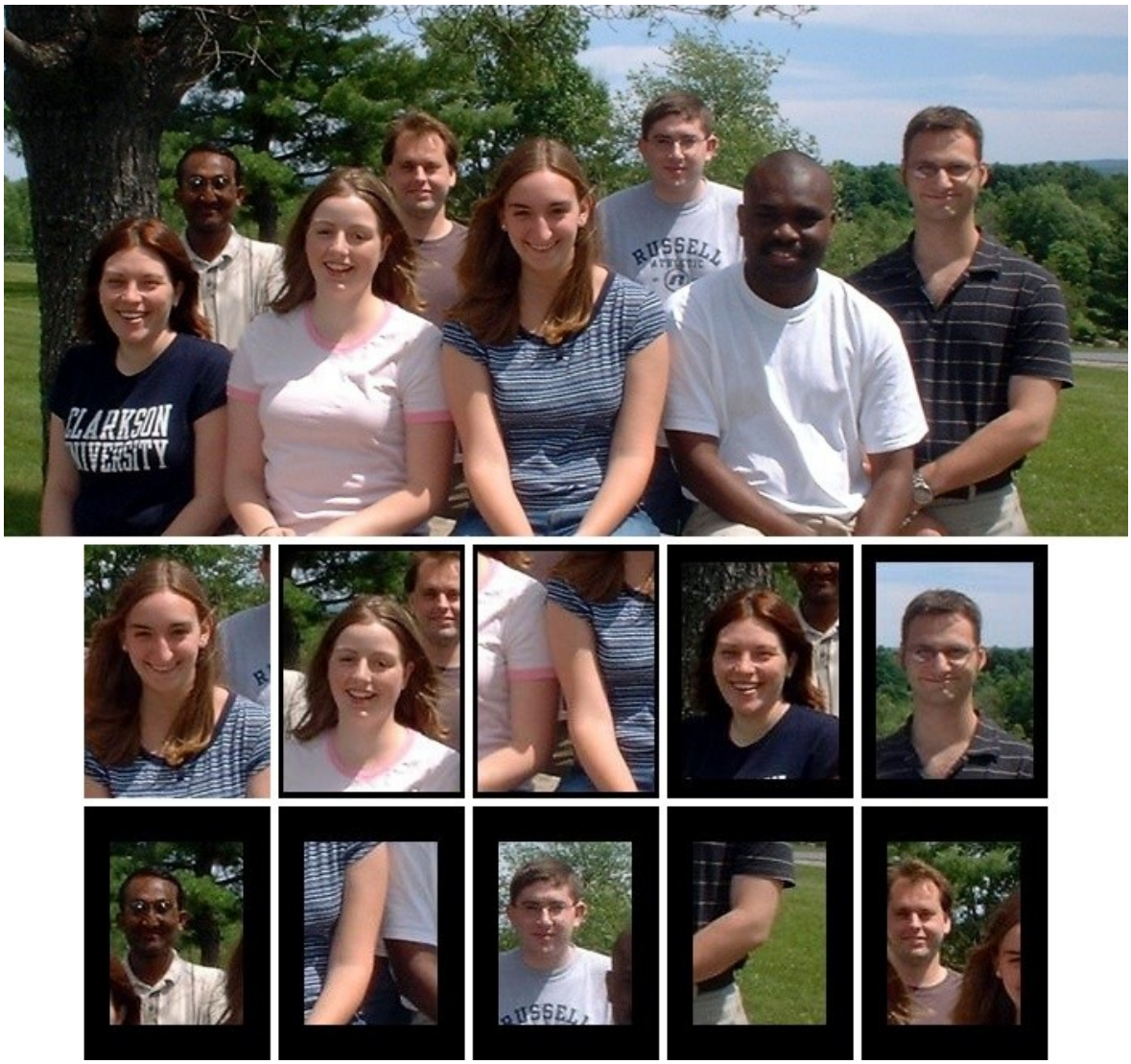
6.2 Lokalizace obličejů pomocí vzorů

Pole se vzorovým binárním obrazem je posunováno po poli obsahujícím binární obraz skupinové fotografie; v místě kde je nejmenší odchylka bílé oblasti vzoru od pozadí tvořeného výřezem ze zkoumané fotografie je lokalizován možný obličej. Tato oblast je u binárního obrazu „smazána“ (všechny body jsou převedeny na černou barvu). Pro přesnější identifikaci je tato oblast porovnána i v barevné variantě – původní barevný vzor je porovnán s lokalizovanou oblastí u načtené barevné fotografie. Pokud se od sebe výrazně neliší, je výřez fotografie uložen jako samostatný obrázek v jpg souboru. Není-li lokalizován žádný potencionální obličej nebo není v barevné formě určen jako skutečný obličej, je vzor pomoci afinní transformace zmenšen (viz. 5.1.1) a vyhledávání je opakováno až do okamžiku nejmenší určené velikosti vzoru.

Obr. 27. Výsledky lokalizace obličejů u skupinové fotografie s jednoduchým pozadím

Obr. 28. Výsledky lokalizace obličejů u skupinové fotografie s různorodým pozadím

Obr. 29. Výsledek lokalizace obličejů u skupiny osob stojících blízko sebe a s rušivými prvky (barva oblečení je podobná barvě kůže)



Obr. 30. Výsledek lokalizace obličejů u skupinové fotografie s odlišnou velikostí hlav osob a se zhoršenou expozicí (obličeje jsou stíněné)

ZÁVĚR

Předmětem práce je podrobné seznámení s problematikou rozpoznávání obličejů. Je zde nastíněn postup pro identifikaci osob podle fotografii. Nejprve je nutné lokalizovat oblast hlavy. Používanými metodami je porovnávání se vzorem, vhodnou metodou je také určení oblasti obličeje na základě barevných odstínů pleti. Pro přesnou detekci hlavy je nutné ji oddělit od pozadí. Metoda Aktivních kontur dokáže postupným obepínáním přesně zvýraznit hranice hlavy. Pro přesnou identifikaci jsou potřebné další identifikační prvky obličeje, oblasti zájmu se soustředí převážně na oči a ústa. Oči lze určit rovněž pomocí vzoru, pro lokalizaci úst je vhodné použít výrazného rozdílu odstínu rtů oproti odstínu kůže. K vyhodnocování a porovnávání hodnot získaných předzpracováním obrazů slouží neuronové sítě, které vycházejí z principu biologické nervové sítě a využívají distribuované, paralelní zpracování informace při provádění výpočtů.

V praktické části práce je popsán vytvořený program pro získání vzoru určeného k vyhledávání obličejů na hromadných fotografiích a jeho použití kombinované společně s metodou využívající pleťových odstínů. Výsledný program dokáže rozpoznat obličeje různé velikosti, různých barev pleti na různém pozadí. Velký podíl na úspěšné lokalizaci obličeje hraje kvalita expozice fotografie, při zhoršené expozici můžou být barvy výrazně posunuté a pleťové odstíny špatně identifikovatelné. I když se vzor postupně zmenšuje a dokáže rozpoznat různé velikosti obličejů, je pro co nejúspěšnější detekci vhodné, aby byla velikost obličejů co nejbližší velikosti vzoru. Při malých velikostech může být obličej detekován příliš brzy, u velkých velikostí můžou být nesprávně detekovány nežádoucí objekty (především obnažené ruce). Nejlepších výsledků při lokalizaci obličejů může být dosáhnuto při stejných podmínkách fotografování (stejné osvětlení, stejná vzdálenost skupiny...), v takovémto případě mohou být tolerance při výběru potencionálních obličejů co nejnižší.

RESUME

The detailed identification of face recognition problem has been described in this work. The detection of the head area is the first step for the person identification in the photograph. There can be used or matching pattern methods or identification of conformable skin colour areas methods.

The separation of the head from the background has to be done for the precise head detection necessary. The Active Contours Method achives precise head detection by step by step shrinking. The others face identification elements there are the eyes and the mouth. The eyes identification is possible by the pattern; for the mouth the distinct difference in colour nuance between mouth and skin can be used. For the plotting and matching of the values from the pictures preprocessing the neuron nets are exploiting. These neuron nets are based on the biological nerves net and they use the distributed, parallel information processing for computation.

The program for face detection in the group photographs by identification of conformable skin colour areas was developed in the practical part. The program can recognize the faces of different size and colours on different background. The quality of the exposure is important; the colours could be markedly offset and the complexion nuances could be misidentifiable in the case of worse exposure. For the most sucessfull detection the size of the face should be closest to pattern size. In the case of the little size, the face could be detected too early, in the case of the large size there could be detected undesirable objects (especially naked arms). The best results in the face localization we obtain by the same conditions of the photographing (the same lighting and distance of the group). In that case the toleration for the selection of the possible faces can be the lowest.

SEZNAM POUŽITÉ LITERATURY

1: Ing. Jiří Hinner, Detekce a rozpoznávání obličejů osob a jejich identifikační význam, 2003, Ministerstvo vnitra České republiky,

http://www.mvcr.cz/casopisy/kriminalistika/2003/03_01/hinner.html

2: Ing. Petr Hujka, Nová segmentační metoda, principem založená na Gaussově rozdělovací funkci a její aplikace na segmentaci lidské kůže, 2005, Ústav Telekomunikací VUT v Brně, Fakulta elektrotechniky a komunikačních technologií, VUT v Brně,

http://www.elektrorevue.cz/clanky/05054/index.html

3: Pavel Tišnovský, Programujeme JPEG: transformace a podvzorkování barev, 2006, root.cz,

http://www.root.cz/clanky/programujeme-jpeg-transformace-a-podvzorkovani-barev/

4: Václav Krajíček, Měření objemu v 3D datech, 2007

5: Audiovisual Speech Recognition, , Fakulta aplikovaných věd, Katedra kybernetiky, Západočeská univerzita v Plzni, http://www.kky.zcu.cz/en/research-fields/audio-visual-speech-recognition

6: Ing. Karol Molnár, Úvod do problematiky umělých neuronových sítí, 2000, ÚTKO FEI, VUT Brno,

7: O. Drábek, P. Seidl, I. Taufer, Umělé neuronové sítě - teorie a aplikace, 2006, CHEMagazín, Univerzita Pardubice,

8: Ing. Dana Nejedlová, Fonetocká transkripce češtiny pomocí třívrstvé neuronové sítě, 2000, Fakulta mechatroniky a mezioborových inženýrských studií, Technická Univerzita v Liberci,

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: APLIKACE PRO RUČNÍ ZÍSKÁNÍ SOUŘADNIC OČÍ

```
//spusteni okna - class MainObrazek

import javax.swing.*;
public class MainObrazek extends Jframe
{
        public MainObrazek()
        {
        Okno ok= new Okno(2*496, 676);
        //spusteni okna,nastaveni rozmeru
        }
        public static void main (String [] args)
        {
        try
                {
                UIManager.setLookAndFeel(UIManager.
                getSystemLookAndFeelClassName());
                }
        catch (Exception e)
                {
                e.printStackTrace();
                }
        new MainObrazek();
        }
}
```

```
//aplikace pro ziskani souradnic oci - class Okno

import java.awt.*;                  //imporotvane graficke a vst./vyst. tridy
import java.io.*;
import java.awt.event.*;
import javax.swing.*;
public class Okno extends JFrame
{
        private Obrazek o;          //prom. tridy Obrazek - slouzi k nacteni a zobrazeni fotek
        private JLabel l;           //panel pro zobrazovani informaci uzivateli
        private String str;         //prom. pro nazev souboru (nacitane fotky)
        private int x, y, a=1, c=0;   //pomocne prom.
        private OutputStream vystup = null;
        //vystupni proud pro ulozeni cisla souboru a souradnic oci
        public Okno(int vyska, int sirka)
        {       //parametry okna
        this.setSize(vyska, sirka);
        this.setTitle("Načítání obrázku");
        this.setLayout(new GridLayout(1,1));
        JFileChooser chooser = new JFileChooser();
//výběr adresář ze kterého se budou načítat fotografie
    chooser.setCurrentDirectory(new java.io.File("."));
    chooser.setDialogTitle("Vyberte adresář s fotkama");
    chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    chooser.setAcceptAllFileFilterUsed(false);
        if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
        {
        str=(String.valueOf(chooser.getSelectedFile()));
        //ziskani adresy adresare pro nacteni obrazku
        }
        else
        {
        System.out.println("chyba cteni");          //pokud nemuze byt nacten
        }
        o=new Obrazek();                            //inicializace prom. tridy "Obrazek"
        o.loadPicture(480, 640, str+a+".jpg");  //volani funkce nacteni obrazku v tride "Obrazek"
        l=new JLabel("leve"); //v informacniom panelu se zobrazuje ktere oko ma uzivatel oznacit
        l.setSize(50, 20);
        this.getContentPane().add(o);               //pridani objektu na panel
        this.getContentPane().add(l);
                try {                               //vytvoreni souboru pro ulozeni souradnic
                      vystup = new FileOutputStream("data");
                      }
                       catch (IOException f)
                       {
                       System.out.println("Soubor data.dat se nepodarilo
        vytvorit!");
                      return;
                       }
o.addMouseListener(new MouseListener()                    //akce na pouziti mysi
```

```
	{
		public void mouseClicked(MouseEvent e)		//akce po kliknuti mysi
		{
			DataOutputStream dos = new DataOutputStream(vystup);
		if (c<1)			//pokud  nebylo prvni oko oznaceno
			{
			x=e.getX();		//ziskani souradnic
			y=e.getY();
			l.setText("prave");	//inf. panel signalizuje druhe oko
			try
				{
				dos.writeInt(a);
//ulozeni cisla souboru a souradnic leveho oka - volani funkce write
				dos.writeInt(x);
				dos.writeInt(y);
				}
			catch (IOException e1)
				{
				// TODO Auto-generated catch block
				e1.printStackTrace();
				}
			c++;
			}

		else			//pokud uz bylo prvni oko oznaceno
			{
			x=e.getX();	//ziskani druheho oka
			y=e.getY();
				try	//ulozeni souradnic - volani funkce write
				{
					dos.writeInt(x);
					dos.writeInt(y);
				}
				catch (IOException e1)
				{
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}
				l.setText("leve");
				//nastaveni inf. panelu na leve oko dalsiho souboru
				a++;				//pocitadlo na dalsi soubor
				o.loadPicture(480, 640, str+a+".jpg");
								//nacteni noveho souboru
				c=0;
				repaint();			//překreslení panelu
			}
		}
		@Override
		public void mouseEntered(MouseEvent arg0) {  //dalsi akce mysi
			// TODO Auto-generated method stub
		}
```

```
		@Override
		public void mouseExited(MouseEvent arg0) {
			// TODO Auto-generated method stub
		}
		@Override
		public void mousePressed(MouseEvent arg0) {
			// TODO Auto-generated method stub
		}
		@Override
		public void mouseReleased(MouseEvent arg0) {
			// TODO Auto-generated method stub
		}
	}
	);
	this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);	//zavreni souboru
	this.setVisible(true);
	//zobrazeni objektu na panelu
	}
	public void write(int st)
	//funkce pro zapis hodnoty INT do souboru
	{
		try
		{
			vystup.write(st);
		}
		catch (IOException e1)
		{
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}
	}
}
```

```
//třída pro nacteni a zobrazeni obrazku

import java.awt.Graphics;     //importovane graficke a vst./vyst. tridy
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.JComponent;
public class Obrazek extends JComponent
{
        private int width, height;                              //rozmery obrazku
        private String strl;                                    //nazev obrazku
        private BufferedImage im;                               //obrazek
        public Obrazek()
        {
        }
        public void loadPicture(int width, int height, String str)
//funkce pro nacteni obrazku - vst. rozmery a nazev
        {
                this.width=width;
                this.height=height;
                this.strl=str;
                File f=new File(strl);        //nacteni souboru s obrazkem
                try
                {
                        im=ImageIO.read(f);
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
        public void paintComponent(Graphics g)       //vykrtesleni obrazku
        {
                g.drawImage(im,00,00, width,height, this);
        }
}
```

PŘÍLOHA P II: APLIKACE PRO VÝPOČET VZORU

```
//spusteni okna - class „main“

import javax.swing.*;
public class main extends JFrame
{
	public main()
	{
		Nacti ok= new Nacti(200, 80);
//spusteni okna,nastaveni rozmeru
	}
	public static void main (String [] args)
	{
		try
		{
			UIManager.setLookAndFeel(UIManager.
			getSystemLookAndFeelClassName());
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}
		new main();
	}
}
```

```
//trida pro pocitani prumerneho obliceje

import java.awt.*;            //importovane graficke a vst./vyst. tridy
import java.io.*;
import java.awt.event.*;
import javax.swing.*;
public class Nacti extends JFrame  implements ActionListener{
        private JButton b;                          //tlacitko pro START
        private String str="1";                     //nazev souboru
        private InputStream vstup = null;           //vstupni proud
        private OutputStream vystup = null;         //vystupni proud
        private long vel;     //velikost soubor (pocet obrazku v ulozenem souboru se souradnicemi)
        private int [] x,y;                  // pole pro souradnice oci
        private int c=0,a=0,p=0;             //pomocne prom.
        private Obrazek o;                   //prom. tridy Obrazek
        private int [][][] poleA, poleB;     //pole pro nacteni obrazku
        public Nacti(int vyska, int sirka){
                this.setSize(vyska, sirka);          //hodnoty okna
                this.setTitle("Načítání souboru");
                this.setLayout(new GridLayout(1,1));
                b=new JButton("stiskni");            //tlacitko pro start
                x=new int[2];                        //inicializace poli pro souradnice obo oci
                y=new int[2];
                poleA=new int[480][640][3];          //inicializace poli pro nacteni obrazku
                poleB=new int[480][640][6];
                vel=velikost();                      //zjisteni poctu obrazku

                try
                 {                           //nacteni souboru se souradnicemi oci u obrazku
                   vstup = new FileInputStream("..\\Obrazek\\data");
                 }
                 catch (IOException e)
                 {
                   System.out.println("Soubor DATA se nepodarilo otevrit!");
                      return;
                 }
                 try
                 {
//vytvoreni souboru pro ulozeni vysledne matice s vypoctenym prumernym oblicejem
                            vystup = new FileOutputStream("matrix");
                 }
                 catch (IOException f)
                 {
        System.out.println("Soubor data.dat se nepodarilo vytvorit!");
                             return;
                 }
        JFileChooser chooser = new JFileChooser();                  //vyber adresare s fotkama
        chooser.setCurrentDirectory(new java.io.File("."));
        chooser.setDialogTitle("Vyberte adresář s fotkama");
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
```

```
        chooser.setAcceptAllFileFilterUsed(false);
        if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
                        {
                         str=(String.valueOf(chooser.getSelectedFile()));
                        }
                else
                        {
                        System.out.println("cyhba v adresari s fotkama");
                        }
                    str=str+"\\";          //pripojeni lomitka k ceste k souboru
                 o=new Obrazek();          //inicializace prom. tridy Obrazek
                    b.addActionListener(this);          //reakce na tlacitko
                this.getContentPane().add(b);
                this.getContentPane().add(o);
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                this.setVisible(true);
        }
        public void actionPerformed(ActionEvent e)          //stisknuti tlacitka
        {
                 for (int h=0;h<vel;h++)                    //dokud neni posledni fotka
                 {
                        c=0;
                        a=cti();              //volana funkce pro nacteni cisla
                          x[c]=cti();         //volana funkce pro nacteni souradnic leveho oka
                          y[c]=cti();
                         c++;                 //presun v poli
                   x[c]=cti();                //volana funkce pro nacteni souradnic praveho oka
                   y[c]=cti();
                   System.out.println(a);
                   o.loadPicture(480, 640,str+a+".jpg",x[0],x[1],y[0],y[1]);
//nacteni obrazku tridou Obrazek, funkci loadPicture
                   poleA=o.loadMatrix();              //nacteni obrazku do poleA po transformacich
                   for (int x=0;x<480;x++)            //bod po bodu obrazku
                   {
                                for (int y=0;y<640;y++)
                                {   p=0;
                                        for (int z=0;z<3;z++)         //tri barvy RGB
                                {
                                poleB[x][y][z]=poleB[x][y][z]+poleA[x][y][z]; //pricteni poleA k poliB
                                        if (poleA[x][y][z]!=0) poleB[x][y][z+3]++;
//pokud poleA obsahuje slozku pricteni do pocitadla obsazenych obrazku v subbodu
                  }}}
        }
                   for (int x=0;x<480;x++)                          //bod po bodu obrazku
                   {
                                for (int y=0;y<640;y++)
                                {
                                        for (int z=0;z<3;z++)
                                        {
                                                if (poleB[x][y][z+3]>0)
```

```
//vypocet prumeru subbodu podle pocitadla obsazenych obrazku
                            poleB[x][y][z]=poleB[x][y][z]/poleB[x][y][z+3];
                                    }
                            }
                    }
                o.paintMatrix(poleB);                   //vykresleni vysledneho obliceje
                o.savePicture(str);                     //ulozeni vysledneho obliceje do jpg
                 DataOutputStream dos = new DataOutputStream(vystup);
            //ulozeni pole s vyslednym oblicejem do souboru "matrix"
                  try {
                     for (int x=0;x<480;x++){
                                        for (int y=0;y<640;y++)
                                        {
                                                for (int z=0;z<3;z++)
                                                {
                                        dos.writeInt(poleB[x][y][z]);
                                                }
                                        }
                     }
                            } catch (IOException e1) {
                                    // TODO Auto-generated catch block
                                    e1.printStackTrace();
                            }
        repaint();                          //prekresleni panelu
        }
        public int cti()
        {                                   //funkce pro nacteni cteni vstupni fotky
                int f=0;
                DataInputStream zes = new DataInputStream(vstup);
                try
                {
                        f = zes.readInt();
                }
                catch (IOException e1)
                {
                        System.out.println("Nectu!");
                        e1.printStackTrace();
                }
                return f;
        }

        public long velikost()
        {               //funkce pro zjistni poctu fotek ze souradnicoveho souboru
                long f=0;
                File zes = new File("..\\Obrazek\\data");
                        f = zes.length();
                return f/20;
        }
}
```

```
//operacde s obrazkem (nacteni, zobrazeni, afinni transf.)- class Obrazek

import java.awt.Color;
import java.awt.Graphics;
import java.awt.geom.AffineTransform;
import java.awt.image.*;
import java.awt.image.BufferedImage;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import javax.imageio.ImageIO;
import javax.swing.JPanel;

public class Obrazek extends JPanel{
private int width, height, xB,yB,c,z;        //rozmery, pom. prom.
private String strl;                         //cesta k obrazku
private BufferedImage im;                    //nacteny obrazek
private double x1,x2,y1,y2,x3,y3;            //souradnice oci + x3 a y3  pomocna prom.
private double uhel,prepona;                 //pocitani uhlu a prepony
private OutputStream vystup = null;          //vystupni proud pro ulozeni obrazku v jpg
public int[][][] poleB=new int [480][640][3];       //pole pro obrazek
public int[][][] poleA=new int [480][640][3];       //pomocne pole pro transf.
public Obrazek()
{
}
public void loadPicture(int width, int height, String str,double x1, double x2, double y1, double
y2)
//funkce pro nacteni transformaci obrazku - rozmery obrazku, cesta k obrazku, souradnice obou oci
{
        this.width=width;
        this.height=height;
        this.strl=str;
        this.x1=x1;
        this.x2=x2;
        this.y1=y1;
        this.y2=y2;
        File f=new File(strl);        //nacteni souboru s obrazkem
        try
                {
                        im=ImageIO.read(f);
                }
        catch (Exception e)
                {
                        e.printStackTrace();
                }
        uhel=Math.tan((y1-y2)/(x2-x1));      //urceni uhlu otoceni
        prepona=(x2-x1)/Math.cos(uhel);
//vypocet prepony trojuhelniku z oci - vzdalenost oci od sebe pro zmenu meritka
```

```
        prepona=110*480/prepona;
        prepona=prepona/480;
        im=filter(im);          //funkce pro zpracovani (transformace) obrazu
        int x=0,y=0,b=0;                //prom pro pohyb v poli
        for (x=0;x<480;x++)
        {
                for (y=0;y<640;y++)
                {
                        c=im.getRGB(x,y);               //rozdeleni RGB na tri kanaly
                        int red = (c & 0x00ff0000) >> 16;
                        int green = (c & 0x0000ff00) >> 8;
                        int blue = c & 0x000000ff;
                        poleA[x][y][0]=red;
                        poleA[x][y][1]=green;
                        poleA[x][y][2]=blue;
                }
        }
        for (x=0;x<480;x++)                             //vynulovani poleB
                {
                        for (y=0;y<640;y++)
                        {
                                poleB[x][y][0]=0;
                        }
                }
        x3=x1;
        y3=y1;
        x3=(int)(x1*prepona);                   //vypoet noveho bodu oka
        y3=(int)(y1*prepona);
        for (x=0;x<480;x++)                                     //posunuti
                {
                        for (y=0;y<640;y++)
                        {
                                xB=(int) (x+180-x3);            //vypocet nove souradnice
                                yB=(int) (y+280-y3);
                                if (xB<480 && yB<640 && xB>0 && yB>0)
                                        //podminky nevypadnuti z obrazu
                                        {
                                                for (z=0;z<3;z++)
                                                {
                                               poleB[(int)xB][(int)yB][z]=poleA[x][y][z];
                //transformace subbodu poleA do poleB
                }}}}
        for (x=0;x<480;x++)
        {
                for (y=0;y<640;y++)
                {
                        for (z=0;z<3;z++)
                        {
                                poleA[x][y][z]=poleB[x][y][z];
//navrat transformovaneho obrazku z pomocne matice
                        }
```

```
                }
        }
}
public void paintComponent(Graphics g)                      //vykresleni obrazku
{
    g.drawImage(im,0,0, width,height, this);
}
public BufferedImage filter(BufferedImage src)              //nacteni souboru do "im" - obrazku
{
    AffineTransform at = new AffineTransform();             //inicializace afinnich transf.
    BufferedImage bif = new BufferedImage(480, 640, src.getType());
    at.scale(prepona,prepona);              //zmena meritka
    at.rotate(uhel,x1,y1);                  //rotace
    AffineTransformOp ato = new AffineTransformOp(at, null);
    ato.filter(src,bif);                    //provedeni transformaci na obrazku
    return bif;
  }
public int[][][] loadMatrix()
{               //vraci poleA - nacteni pretransf. pole s obrazkem
        return poleA;
}
public  void paintMatrix(int[][][] poleC)           //vykresleni pole s obrazkem
        {
                for (int x=0;x<480;x++)
                {
                        for (int y=0;y<640;y++)
                        {
                        Color ce=new Color(poleC[x][y][0],poleC[x][y][1],poleC[x][y][2]);
//urceni hodnoty RGB s barevnych kanalu R,G a B - subbodu
                        im.setRGB(x,y, ce.getRGB());        //nastaveni hodnoty RGB
        }}}
public void savePicture(String str)         //ulozeni obrazku do jpg souboru - vysledneho obliceje
{
        try {
              vystup = new FileOutputStream(str+"universal.jpg");
              } catch (IOException f) {
             System.out.println("Soubor data.dat se nepodarilo vytvorit!");
               return;
    }
              DataOutputStream dos = new DataOutputStream(vystup);
                        try {
                                ImageIO.write(im, "jpg", dos);

                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
}
}
```

PŘÍLOHA P III: APLIKACE PRO LOKALIZACI OBLIČEJŮ

```
//spusteni okna

import javax.swing.JFrame;
import javax.swing.UIManager;
public class mainY extends JFrame
{
	public mainY()
	{
		rgbconvert ok= new rgbconvert (600, 500);    //spusteni okna,nastaveni rozmeru
	}
	public static void main (String [] args)
	{
		try
		{
			UIManager.setLookAndFeel(UIManager.
					getSystemLookAndFeelClassName());
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}
		new mainY();
	}
}
```

```
//hledani obliceju, prevod RGB na YCbCr - class rgbconvert

import java.awt.GridLayout;
import java.io.*;
import javax.swing.*;
import math.Matrix;
public class rgbconvert extends JFrame
{
        private InputStream vstup = null;
        private double [][][] poleA, poleVz;        //pole pro vysledny obrazek a pro CB vzor
        int[][][] poleB, poleVzB,poleTisk;
                                    //pomocne pole, pole pro barevny vzor a pole pro zobrazeni
        private zobraz o;                           //prom. tridy ZOBRAZ - prace s obrazkem
        private String str,vzorBr="vzor4.jpg";      //nazev souboru, nazev vzoru
        private int width, height,widthVz, heightVz,widthVzB, heightVzB,MINX,MINY,widthPoc,
heightPoc;                                          //rozmery obrazku a poli, max a min pri hledani
private Matrix MVelikost = new Matrix( 3, 3), MBod = new Matrix( 3, 1);
                                                    //matice pro vypocet zmenseni
        private double xM;
        public rgbconvert(int vyska, int sirka)
        {
                this.setSize(vyska, sirka);                 //vlastnosti okna
                this.setTitle("Vyhledavac");
                this.setLayout(new GridLayout(1,1));
                int x,y = 0,z;                                      //pom. prom.
                double pCb=0,pCr=0;
        JFileChooser chooser = new JFileChooser("");        //vyber fotky pro zkoumani
            chooser.setDialogTitle("Vyberte hledanou fotku");
            if (chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
            {
                str=(String.valueOf(chooser.getSelectedFile()));
            }
            else
            {
                System.out.println("spatna grupa");
            }
            MVelikost.Set(1, 2, 0); MVelikost.Set(1, 3, 0);   //nastaveni matoce pro vypocet zmenseni
            MVelikost.Set(2, 1, 0); MVelikost.Set(2, 3, 0);
            MVelikost.Set(3, 1, 0); MVelikost.Set(3, 2, 0); MVelikost.Set(3, 3, 1);
                o=new zobraz();
                poleVzB=o.loadPicture(vzorBr);//nacteni vzoroveho obrazku
                widthVz=o.getWidth();         //zjisteni rozmeru vzoru
                heightVz=o.getHeight();
                widthPoc=widthVz;
                heightPoc=heightVz;
                poleB=o.loadPicture(str);     //nacteni skupinoveho fota
                width=o.getWidth();           //zjisteni jeho rozmeru
                height=o.getHeight();
                poleA=new double[width][height][3];         //inicializace poli
                poleVz=new double [widthVz][heightVz][3];
```

```
                for (x=0;x<widthVz;x++)              //prevod Vzoru RGB na YCbCr
                {
                        for (y=0;y<heightVz;y++)
                        {
                                poleVz[x][y][0]= (0.299*poleVzB[x][y][0]+0.587*poleVzB[x][y]
[1]+0.114*poleVzB[x][y][2]);
                                poleVz[x][y][1]= (128-0.1687*poleVzB[x][y][0]-0.3313*poleVzB[x][y]
[1]+0.5*poleVzB[x][y][2]);
                                poleVz[x][y][2]= (128+0.5*poleVzB[x][y][0]-0.4187*poleVzB[x][y]
[1]-0.0813*poleVzB[x][y][2]);
                                pCb=pCb+poleVz[x][y][1];
                                        //celkova hodnota kanalu Cb a Cr
                                pCr=pCr+poleVz[x][y][2];
                        }
                }
                pCr=pCr/(widthVz*heightVz);   //prumerna hodnota kanalu Cb a Cr
                pCb=pCb/(widthVz*heightVz);
                for (x=0;x<widthVz;x++)
//zjisteni pletove barvy, z intervalu prumernych Cb a Cr
                {
                        for (y=0;y<heightVz;y++)
                        {
                                if ((poleVz[x][y][2]>1.1*pCr) && (poleVz[x][y][1]<0.95*pCb) &&
(poleVz[x][y][2]<1.4*pCr)) poleVz[x][y][0]=255;
                                else poleVz[x][y][0]=0;
                        }
                }
                for (x=0;x<width;x++)         //prevod Skupiny RGB na YCbCr
                {
                        for (y=0;y<height;y++)
                        {
poleA[x][y][0]=0.299*poleB[x][y][0]+0.587*poleB[x][y][1]+0.114*poleB[x][y][2];
poleA[x][y][1]=128-0.1687*poleB[x][y][0]-0.3313*poleB[x][y][1]+0.5*poleB[x][y][2];
poleA[x][y][2]=128+0.5*poleB[x][y][0]-0.4187*poleB[x][y][1]-0.0813*poleB[x][y][2];
                pCb=pCb+poleA[x][y][1];//celkova hodnota kanalu Cb a Cr
                pCr=pCr+poleA[x][y][2];
                        }
                }
                pCr=pCr/(width*height);       //prumerna hodnota kanalu Cb a Cr
                pCb=pCb/(width*height);
                o.paintMatrix(poleA,str,0);   //zobrazeni a ulozeni kanalu Y
                o.savePicture(str+" Y");
                o.paintMatrix(poleA,str,1);   //zobrazeni a ulozeni kanalu Cb
                o.savePicture(str+" Cb");
                o.paintMatrix(poleA,str,2);   //zobrazeni a ulozeni kanalu Cr
                o.savePicture(str+" Cr");
                o.paintMatrixBarv(poleA,str,0);
                                        //zobrazeni a ulozeni kanalu Y s barevnou pleti
                    o.savePicture(str+" Y2");
                  for (x=0;x<width;x++)
                  {
                                for (y=0;y<height;y++)
```

```
                                {
                                        if ((poleA[x][y][2]>1.1*pCr) && (poleA[x][y][1]<0.95*pCb) &&
(poleA[x][y][2]<1.4*pCr)) poleA[x][y][0]=255;
                                        else poleA[x][y][0]=0;
                                }
                        }
                        o.paintMatrixBW(poleA,str);
                                        //binarniho obrazku Skupiny s vyznacenou barvou pleti
                        o.savePicture(str+" BW3");
                        int X,Y,p=0,MINXD=0,MINYD=0;
                        double d=0,MIN = 999999999,MINc=0,D=0,b=0,B = 0,de=0,t=0,te=0,bila=0,wh=0,
MIND=999999999;
                        widthVzB=widthVz;
                        heightVzB=heightVz;
                        for (double k=1;widthVzB>35;k=k-0.05)
                                        //dokud je strana vzoru vetsi nez minimum, zmensi k
                        {
                                zmensiVz(k);          //zmenis CB vzor o k
                                    zmensiVzB(k);             //zmensi barevny vzor o k
                               for (MINc=0;MINc<10000;MINc=MIN)
                                        //dokud neni prekrocen limit urcujici potencialni oblicej
                               {
                                        MIN=999999999;       //nastaveni minim na maximum
                                        MIND=999999999;
                                        for (X=0;X<width-(int)widthVz;X+=2)
                                        //skenovani skupinove CB fotografie po dvou pixelech
                                        {
                                                for (Y=0;Y<height-(int)heightVz;Y+=2)
                                                {
                                                        for (x=0;x<widthVz;x++)
                                        //prochazeni CB Vzoru bod po bodu
                                                        {
                                                                for (y=0;y<heightVz;y++)
                                                                {
                                                                if (poleVz[x][y][0]>128)
                                        //pokud je bod bily
                                                                        {
d+=(poleVz[x][y][0]-poleA[x+X][y+Y][0])*(poleVz[x][y][0]-poleA[x+X][y+Y][0]);
                                        //odecteni od podkladu (Skupiny)
                                                                        bila++;
                                        //pocitadlo bilych bodu
                                                                        }
                                                                }
                                                        }
                                                d=(d/(bila));
                                        //ziskani prumerneho rozdilu mezi Vzorem a Skupinou
                                                if (MIN>d)
                                        //zjisteni zda rozdil neni minimalni
                                                {
                                                        de=d;
                                                        wh=bila;
                                                        MIN=d;MINX=X;MINY=Y;
                                        //ulozeni minim a pocatecnich souradnic Vzoru na Skupine v minimu
                                                }
```

```
                                        d=0;            //vynulovani rozdilu a pocitadla bile
                                        bila=0;
                                        }
                                }
                                    if(MIN<10000)           //pokud je minimum dostatecne male
                                    {
                                for (x=0;x<widthVz-10;x++)
        //zacerneni stredu mista, kde byla nalezena potencionalni hlava
                                {
                                                for (y=0;y<heightVz-10;y++)
                                                {
if ((x+MINX+0)>0 && (y+MINY+0)>0 && (x+MINX+0)<width && (y+MINY+0)<height)    //je-li uvnitr obrazu
                                poleA[x+MINX+10][y+MINY+10][0]=0;
                                                }
                                }
                                for (x=0;x<widthVzB;x++)
//porovnani barevneho Vzoru na miste minima s barevnou Skupinou
                                    {
                                      for (y=0;y<heightVzB;y++)
                                      {
                                                for (z=0;z<3;z++)
                                                {
                                                        if (poleVzB[x][y][z]>0)
                                                        {
D+=(poleVzB[x][y][z]-poleB[x+MINX][y+MINY][z])*(poleVzB[x][y][z]-poleB[x+MINX][y+MINY][z]);
                                                                b++;
                                                        }
                                                }
                                      }
                                    }
                                        MIND=D/(b);
                                                if (MIND>0 && MIND<12000)
                //kdyz je minimum dostatecne male
                                                {
                                                        b=0;
                                poleTisk=new int[widthVzB][heightVzB][3];
                                                for (x=0;x<widthVzB;x++)
                //ulozeni vyrezu barevne Skupiny pod Vzorem do pole
                                                {
                                                        for (y=0;y<heightVzB;y++)
                                                        {
                                                                for (z=0;z<3;z++)
                                                                {
                                                                        poleTisk[x][y]
[z]=(int) poleB[x+MINX][y+MINY][z];
                                                }}}
        o.paintMatrixInt(poleTisk,vzorBr,widthPoc,heightPoc,widthVzB,heightVzB);
                //zobrazeni a ulozeni lokalizovaenho obliceje
                                        o.savePicture("_"+p+" tisk");
                                                }
                                D=0;
                                p++;
```

```
                                    }
                                }
                    }
                      this.getContentPane().add(o);
                      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                         this.setVisible(true);
}
                      public void zmensiVz(double xM)          //zmenseni CB Vzoru
                      {
                         int x,y, widthZm,  heightZm;
                         int [][] poleZm;
MVelikost.Set(1, 1, xM);              //vyplneni hodnot do matice pro zmenseni
 MVelikost.Set(2, 2, xM);
MBod.Set(1, 1, widthVz); MBod.Set(2, 1, heightVz); MBod.Set(3, 1, 1);      //vyplneni matice bodu
MBod = MBod.MulMat( MVelikost);
//vypocet vvysledne velikosti - max pozice obrazku
                            widthZm=(int)(MBod.Get(1,1));
                            heightZm=(int)(MBod.Get(2,1));
                            poleZm=new int[widthZm][heightZm];
                            for (x=0;x<widthVz;x++)          //transformace bod po bodu
                               {
                                       for (y=0;y<heightVz;y++)
                                       {
        MBod.Set(1, 1, x); MBod.Set(2, 1, y); MBod.Set(3, 1, 1);
                                              MBod = MBod.MulMat( MVelikost);
                                          int widthM=(int)(MBod.Get(1,1));
                                          int heightM=(int)(MBod.Get(2,1));
if (widthM<widthZm && heightM<heightZm)              //pokud se vleze do vysledneho obrazku
                                          {
                                          if (poleVz[x][y][0]<256 && poleVz[x][y][0]>200)
poleZm[widthM][heightM]=255;
                                       else poleZm[widthM][heightM]=0;
                                       }}}
                            for (x=0;x<widthZm;x++)
                               {
                                       for (y=0;y<heightZm;y++)
                                       {
                                                poleVz[x][y][0]=poleZm[x][y];
                               }}
                            widthVz=widthZm ;          //nastaveni rozmeru zmenseneho pole
                               heightVz=heightZm;
                      }
                      public void zmensiVzB(double xM) //zmenseni barevneho Vzoru
                      {
                         int x,y,z, widthZmB,  heightZmB;
                         int [][] [] poleZm;
                          MVelikost.Set(1, 1, xM);
                          MVelikost.Set(2, 2, xM);
MBod.Set(1, 1, widthVzB); MBod.Set(2, 1, heightVzB); MBod.Set(3, 1, 1);
                            MBod = MBod.MulMat( MVelikost);
                            widthZmB=(int)(MBod.Get(1,1));
```

```
                        heightZmB=(int)(MBod.Get(2,1));
                        poleZm=new int[widthZmB][heightZmB][3];

                        for (x=0;x<widthVzB;x++)
                            {
                                    for (y=0;y<heightVzB;y++)
                                    {
MBod.Set(1, 1, x); MBod.Set(2, 1, y); MBod.Set(3, 1, 1);
                                    MBod = MBod.MulMat( MVelikost);
                                            int widthM=(int)(MBod.Get(1,1));
                                            int heightM=(int)(MBod.Get(2,1));
                                            for (z=0;z<3;z++)
                                                {
                                                if (widthM<widthZmB && heightM<heightZmB)
                                                {
if (poleVz[x][y][0]<256 && poleVz[x][y][0]>200) poleZm[widthM][heightM][z]=poleVzB[x][y][z];
                            }}}}
                        for (x=0;x<widthZmB;x++)
                            {
                                    for (y=0;y<heightZmB;y++)
                                    {
                                            for (z=0;z<3;z++)
                                            {
                                                    poleVz[x][y][z]=poleZm[x][y][z];
                            }}}
                        widthVzB=widthZmB ;
                            heightVzB=heightZmB;
                    }
}
```

```
//zpracovani obrazku - load, save - class zobraz

import java.awt.Color;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import javax.imageio.ImageIO;
import javax.swing.JPanel;
public class zobraz extends JPanel{
	private int width1, height1,width, height,c;
	private String strl;
	private BufferedImage im,img;
	public int[][][] poleA;
	private OutputStream vystup = null;
	public  void paintMatrix(double[][][] poleA, String str,int b)
//funkce pro vykresleni obrazku z pole
	{
		File f=new File(str);
		try
		{
			im=ImageIO.read(f);
		}
		catch (Exception e)
		{
		e.printStackTrace();
		}
		int [][][] poleB=new int [width][height][3];	//prekopirovani do poleB
		for (int i=0;i<width;i++)
		{
			for (int j=0;j<height;j++)
			{
				for (int k=0;k<3;k++)
				{
					poleB[i][j][k]=(int) poleA[i][j][k];
		}}}
		for (int i=0;i<width;i++)		//slouceni kanalu do hodnoty RGB
		{
			for (int j=0;j<height;j++)
			{
			Color ce=new Color(poleB[i][j][b],poleB[i][j][b],poleB[i][j][b]);
			im.setRGB(i,j, ce.getRGB());
		}}
	}
	public  void paintMatrixBW(double[][][] poleA, String str)
						//funkce pro vykresleni binarni Skupiny
	{
```

```
                File f=new File(str);
                try
                {
                        im=ImageIO.read(f);
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
                int [][][] poleB=new int [width][height][3];
                for (int i=0;i<width;i++)
                {
                        for (int j=0;j<height;j++)
                        {
                                poleB[i][j][0]=(int) poleA[i][j][0];
                }}
                for (int i=0;i<width;i++)
                {
                        for (int j=0;j<height;j++)
                        {
                        Color ce=new Color(poleB[i][j][0],poleB[i][j][0],poleB[i][j][0]);
                                im.setRGB(i,j, ce.getRGB());
                }}
        }
public  void paintMatrixInt(int[][][] poleA, String st,int widthVz,int heightVz,int widthVzB,int
heightVzB)
//vykresleni vysledneho lokalizovaneho obliceje
                {
                                File f=new File(st);
                                try
                                {
                                        im=ImageIO.read(f);
                                }
                                catch (Exception e)
                                {
                                        e.printStackTrace();
                                }
                                for (int i=0;i<widthVz;i++)
                                        {
                                                for (int j=0;j<heightVz;j++)
                                                {
                                                im.setRGB(i,j, 0);

                                        }}
                                double x=(widthVz-widthVzB)/2;
                                double y=(heightVz-heightVzB)/2;
                                for (int i=(int) x;i<(widthVzB+x);i++)
                                        {
                                                for (int j=(int) y;j<(heightVzB+y);j++)
                                                {
```

```
Color ce=new Color(poleA[(int) (i-x)][(int) (j-y)][0],poleA[(int) (i-x)][(int) (j-y)][1],poleA[(int) (i-x)][(int) (j-y)][2]);
                                                    im.setRGB(i,j, ce.getRGB());
                                    }}
                }
        public  void paintMatrixBarv(double[][][] poleA, String str,int b)
        //zobrazeni kanalu Y s barenou pleti
        {
                File f=new File(str);
                try
                {
                        im=ImageIO.read(f);
                }
                catch (Exception e)
                {
                        e.printStackTrace();}
                        int [][][] poleB=new int [width][height][3];
                for (int i=0;i<width;i++)
                {
                        for (int j=0;j<height;j++)
                        {
                                for (int k=0;k<3;k++)
                                {
                                        poleB[i][j][k]=(int) poleA[i][j][k];
                }}}
                for (int i=0;i<width;i++)
                {
                        for (int j=0;j<height;j++)
                        {
                        if (poleB[i][j][2]>149 && poleA[i][j][2]<168){      //vybarveni pleti
                                        poleB[i][j][0]=poleB[i][j][0]+30;
                                        poleB[i][j][1]=poleB[i][j][0]-30;
                                        poleB[i][j][2]=poleB[i][j][0]-40;
                                }
                                else { poleB[i][j][2]=poleB[i][j][0];
                                poleB[i][j][1]=poleB[i][j][0];
                                poleB[i][j][0]=poleB[i][j][0];
                }}}
                for (int i=0;i<width;i++)
                {
                        for (int j=0;j<height;j++)
                        {
                Color ce=new Color(poleB[i][j][0],poleB[i][j][1],poleB[i][j][2]);
                                im.setRGB(i,j, ce.getRGB());
                }}

        }
        public void paintComponent(Graphics g)                      //vykresleni obrazku
        {
                g.drawImage(im,0,0,width,height, this);
        }
```

```
public int[][][] loadPicture(String strl)           //nacteni souboru jpg do pole
{
        File f=new File(strl);
        try
        {
                img=ImageIO.read(f);
        }
        catch (Exception e)
        {
                e.printStackTrace();
        }
        width=img.getWidth();
        height=img.getHeight();
        poleA=new int [width][height][3];
        for (int x=0;x<width;x++)
        {
                for (int y=0;y<height;y++)           //rozdeleni RGB na kanaly
                {
                        c=img.getRGB(x,y);
                        int red = (c & 0x00ff0000) >> 16;
                        int green = (c & 0x0000ff00) >> 8;
                        int blue = c & 0x000000ff;
                        poleA[x][y][0]=red;
                        poleA[x][y][1]=green;
                        poleA[x][y][2]=blue;
                }
        }
return poleA;
}
public void savePicture(String str)          //ulozeni obrazku do jpg souboru
{
        try
        {
            vystup = new FileOutputStream(str+".jpg");
        }
        catch (IOException f)
        {
            System.out.println("Soubor data.dat se nepodarilo vytvorit!");
            return;
    }
        DataOutputStream dos = new DataOutputStream(vystup);
        try
        {
                ImageIO.write(im, "jpg", dos);
        }
        catch (IOException e1)
        {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
        }
```

```
        }
        public int getWidth()
        {
                return width;
        }
        public int getHeight()
        {
                return height;
        }
}
```
