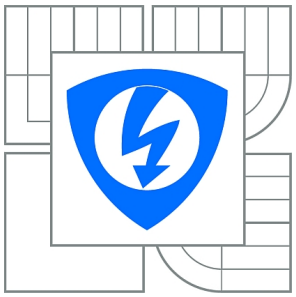
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ROZPOZNÁVÁNÍ OBLIČEJŮ V OBRAZE

FACE RECOGNITION IN DIGITAL IMAGES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

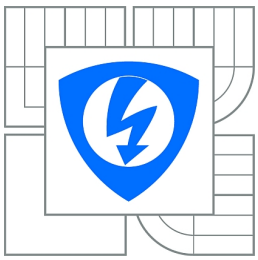
AUTOR PRÁCE — Bc. VÁCLAV HAUSER

AUTHOR

VEDOUCÍ PRÁCE — Ing. KAMIL ŘÍHA, Ph.D.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Václav Hauser	***ID:***	98497
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Rozpoznávání obličejů v obraze

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte současné přístupy pro rozpoznávání obličejů v digitálních obrazech či video sekvencích. Vybrané algoritmy implementujte pomocí vhodného vývojového prostředí. Zaměřte se na vytvoření ucelené vzorové aplikace a teoretických podkladů pro výukové účely v laboratorních cvičeních předmětu MPZO. Hlavním cílem je vytvoření vzorové aplikace pro demonstraci základních postupů při rozpoznávání obličejů včetně kompletace teoretických podkladů s přehledem moderních trendů v této oblasti. Doporučené vývojkové nástroje: MS Visual C++ 2010 a knihovny pro implementaci algoritmů počítačového vidění OpenCV s C++ API.

DOPORUČENÁ LITERATURA:

[1] GONZALEZ R. C., WOODS R. E.: Digital Image Processing, Prentice Hall, New Jersey, 2002
[2] FISHER R. B.: CVonline: The Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision, http://homepages.inf.ed.ac.uk/rbf/CVonline/
[3] LI S. Z., JAIN A. K.: Handbook of face recognition, Springer, New York, 2005

Termín zadání:	6.2.2012	***Termín odevzdání:***	24.5.2012

Vedoucí práce: Ing. Kamil Říha, Ph.D.
Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce se zabývá problematikou detekce a rozpoznávání obličejů v obraze. Obsahem práce je popis využívaných metod detekce a rozpoznávání obličejů. Podrobněji je popsána metoda analýzy hlavních komponent (PCA), která je následně využita při implementaci rozpoznávání obličejů ve videosekvenci. Ve spojení s implementací je v práci popsán balíček knihoven OpenCV, který byl pro realizaci využit, konkrétně jeho C++ API. Závěrem je provedeno testování vzniklé aplikace na dvou rozdílných videosekvencích.

Abstract

This master thesis deals with the detection and recognition of faces in the image. The content of this thesis is a description of methods that are used for the face detection and recognition. Method described in detail is the principal component analysis (PCA). This method is subsequently used in the implementation of face recognition in video sequence. In conjunction with the implementation work describes the OpenCV library package, which was used for implementation, specifically the C ++ API. Finally described application tests were done on two different video sequences.

Klíčová slova

Detekce obličeje, rozpoznání obličeje, trénovací sekvence, testovací sekvence, eigenfaces, extrakce příznaků, klasifikátor, PCA, Viola & Jones, OpenCV

Keywords

Face detection, face recognition, train sequence, test sequence, eigenfaces, feature extraction, classifier, PCA, Viola & Jones, OpenCV

HAUSER, V. *Rozpoznávání obličejů v obraze.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 60 s. Vedoucí diplomové práce Ing. Kamil Říha, Ph.D..

Prohlášení

Prohlašuji, že svůj semestrální projekt na téma Rozpoznávání obličejů v obraze jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....
podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Kamilu Říhovi, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne ....
podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

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

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Obsah

Seznam obrázků:

Seznam tabulek:

Seznam kódů:

Úvod

V moderní době jsou na počítače kladeny stále větší požadavky a lidé se snaží ulehčit si a zpříjemnit co nejvíce činností, při kterých počítače využívají. Jedním z rychle se rozvíjejících odvětví je počítačové vidění a s ním související detekce a rozpoznávání objektů v obrazech. S tímto odvětvím jde také ruku v ruce obor biometrie. Jde o způsoby identifikace člověka, se kterými se dnes setkáváme téměř na každém kroku, od vlastních notebooků přes kamerové dohledové systémy až k autentizaci osob v hotelových pokojích. Tato práce se zaměřuje na metody detekce a rozpoznávání obličejů v obrazech a jejich praktické využití.

1 Biometrie

1.1 Co je biometrie

Biometrie slouží k jednoznačné identifikaci osob na základě jejich jedinečných fyziologických nebo behaviorálních znaků. Biometrické systémy mají za úkol automaticky identifikovat nebo autentizovat určité fyzické osoby. Vstupem obou režimů biometrických systémů jsou osobní biometrická data dané osoby. Při identifikaci se vstupní data porovnávají s širokou databází příznaků, dokud není nalezena shoda, a při autentizaci se data porovnají pouze s příznaky přiřazenými k autentizované osobě [22]. Na rozdíl od ostatních způsobů identifikace u biometrie nehrozí zapomenutí hesla, či jeho uhodnutí cizí osobou, nebo ztráta, či odcizení identifikační karty. Jedná se o určitou formu elektronického klíče, kde klíčem je samotný člověk [9].

Rozlišují se dvě hlavní skupiny sbíraných biometrických dat [16]:

1) **Fyziologická data** – obličej, žíly na rukou, pach, otisk prstu, otisk dlaně, ušní boltec, geometrie ruky, oční duhovka, oční sítnice, DNA, termogram
2) **Behaviorální data** – hlas, podpis, stisk klávesnice, chůze

Aby mohla být uvedená fyziologická a behaviorální data využita v praxi, musejí splňovat určité teoretické a praktické požadavky [22] [16]:

a) **Teoretické požadavky:**
   - *Univerzálnost* – charakteristiku by měla mít každá osoba
   - *Jednoznačnost* – charakteristika by nikdy neměla být pro dvě rozdílné osoby stejná
   - *Stálost* – charakteristika by se v průběhu života neměla měnit nebo podléhat náhlým změnám
   - *Shromáždítelnost* – charakteristika by měla být hromadně měřitelná

b) **Praktické požadavky:**
   - *Výkon* – dosažitelná přesnost a rychlost jaké může systém dosáhnout

- *Přijatelnost* – míra přijatelnosti využívání systému pro uživatele
- *Bezpečnost* – míra zabezpečení systému proti podvodným útokům

Tab 1.1: Porovnání biometrických metod [22] [16]

Biometrická charakteristika	Univerzálnost	Jednoznačnost	Stálost	Shromáždítelnost	Výkon	Přijatelnost	Bezpečnost
Obličej	vysoká	nízká	střední	vysoká	nízká	vysoká	nízká
Žíly na rukou	střední	střední	střední	střední	střední	střední	nízká
Pach	vysoká	vysoká	vysoká	nízká	nízká	střední	nízká
Otisk prstu	střední	vysoká	vysoká	vysoká	vysoká	střední	vysoká
Otisk dlaně	střední	vysoká	vysoká	střední	vysoká	střední	střední
Ušní boltec	střední	střední	vysoká	střední	střední	vysoká	střední
Geometrie ruky	střední	střední	střední	vysoká	střední	střední	střední
Oční duhovka	vysoká	vysoká	vysoká	střední	vysoká	nízká	vysoká
Oční sítnice	vysoká	vysoká	střední	nízká	vysoká	nízká	vysoká
DNA	vysoká	vysoká	vysoká	nízká	vysoká	nízká	nízká
Termogram	vysoká	vysoká	nízká	vysoká	střední	vysoká	nízká
Hlas	střední	nízká	nízká	střední	nízká	vysoká	nízká
Podpis	nízká	nízká	nízká	vysoká	nízká	vysoká	nízká
Stisk klávesy	nízká	nízká	nízká	střední	nízká	střední	střední
Chůze	střední	nízká	nízká	vysoká	nízká	vysoká	střední

1.2 Fyziologické metody

1.2.1 DNA

DNA (Deoxyribonukleová kyselina) je nukleová kyselina, která je nositelkou genetické informace všech buněčných organismů na zemi. Základní jednotkou DNA jsou nukleotidy, řetězec nukleotidů je gen a celková informace zahrnutá v DNA se nazývá genom [17]. Lidský genom je pro každého jedince unikátní s výjimkou jednovaječných dvojčat, která mají shodný vzorec DNA. Toto je jeden z důvodů, proč se tato metoda využívá velice málo. Dalším důvodem mohou být například vysoké náklady na testování vzorků, ochrana soukromí a celkem vysoká snadnost získání cizího vzorku DNA a jeho zneužití [22]. V praxi se tato metoda využívá především v kriminalistice.

1.2.2 Obličej

Rozpoznávání obličeje je jedna z nejpřijatelnějších metod, jak z hlediska uživatelského, tak z hlediska praktického. Jednak nijak nenarušuje soukromí uživatele a je vcelku jednoduše shromážditelná. Metoda je založena na měření vzájemné polohy specifických bodů obličeje a probíhá ve třech krocích:

1) Zjištění zda se na obrázku nachází tvář
2) Přesná lokalizace tváře
3) Rozpoznání tváře

V praxi se tato metoda potýká s několika problémy, jako jsou různé podmínky při pořizování fotografií a stárnutí identifikovaných osob [22]. Je však hojně využívána například k vyhledávání nežádoucích osob v davu na fotbalových stadionech či na letištích nebo při hledání pohřešovaných.

1.2.3 Otisk prstu

V dnešní době asi nejrychleji se rozvíjející metoda autentizace a identifikace. Funkční princip se skládá ze tří fází:

1) Získání obrazu otisku
2) Vyhodnocení otisku
3) Identifikace otisku

Pro sejmutí obrazu otisku se v dnešní době využívají tři druhy snímačů:

1) Optické – založené na změně odrazivosti skla v místě dotyku linie prstu. Snímání se uskutečňuje CCD čipem (Charge-Couple Device chip).
2) Kapacitní – měří kapacitu jednotlivých částí otisku. Měření se provádí lineárním kapacitním senzorem.
3) Ultrazvukové – pro zjištění povrchu prstu se využívá ultrazvuk [9].

Při následném vyhodnocování jsou detekovány hřebeny a údolí na sejmutém otisku a podle jejich poloh je provedena identifikace. Nevýhodou, například oproti detekci obličeje, je nutnost spolupráce s identifikovaným [22]. V praxi je

dnes tato metoda asi nejrozšířenější, kromě využití v kriminalistice se snímače implementují i do nejrůznějších druhů spotřební elektroniky.

1.3 Behaviorální metody

Behaviorální metody jsou všeobecně považovány za méně bezpečné, proto se využívají spíše jako sekundární autentizační metody.

1.3.1 Stisk klávesy

Rozlišovacím znakem jedinců u této metody je rozdílnost dynamiky úhozů při psaní na klávesnici. Jednak jsou sledovány mezery mezi jednotlivými úhozy a doba, po kterou je klávesa stisknuta. Nevýhodou je, že dynamika se může u člověka postupem času měnit, a není nijak zaručeno, že neexistuje další člověk se shodnou dynamikou úhozů [18].

1.3.2 Podpis

Tato metoda je založena na kombinaci fyziologických a behaviorálních vlastností člověka. Principem metody je detekce dynamických vlastností podpisu, jako je rychlost, akcelerace, časování, tlak a směr tahu. Vlastnosti jsou zaznamenávány v trojrozměrném souřadnicovém systému. Výhodou je vysoká bezpečnost, nevýhodou je využitelnost pouze k verifikaci [18].

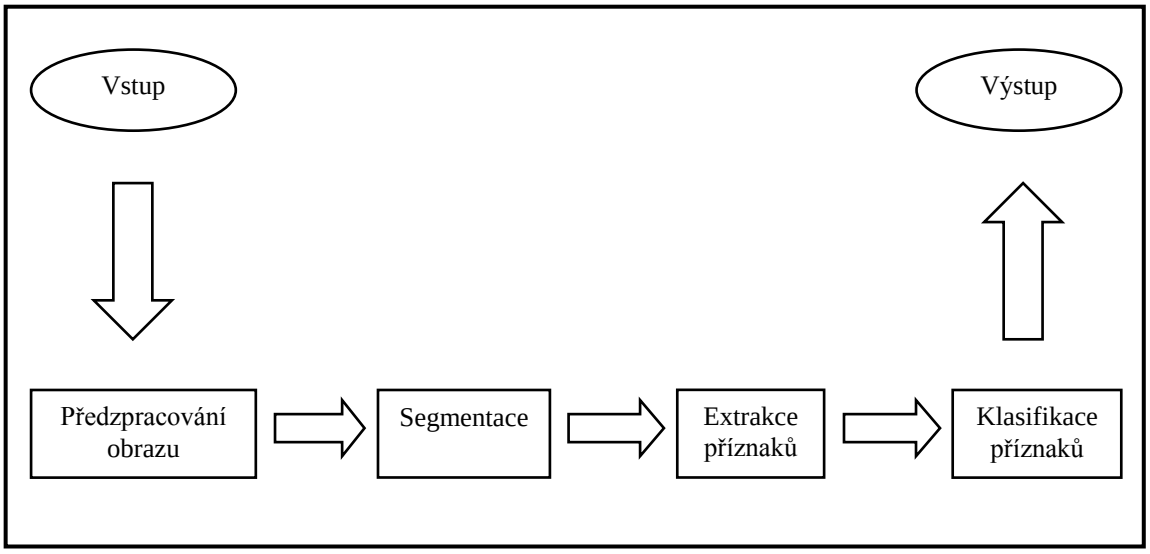
1.3.3 Chůze

Identifikační metoda, jejímž sledovaným znakem je dynamika pohybu určitých bodů na lidském těle během chůze, například těžiště. Svalový a kosterní systém každého člověka je unikátní, čímž je zaručena nezaměnitelnost. Není tedy možné tuto metodu obejít pomocí převleků [18].

2 Detekce a rozpoznávání obličejů v obraze

2.1 Obecný algoritmus

Při detekci jakéhokoliv objektu v obraze se jedná o problematiku rozhodování, zda se hledaný objekt na obrázku nachází, a jeho následná lokalizace. Každá metoda určená k detekci objektu se skládá ze čtyř základních částí (viz obr. 2.1).



Obr 2.1: Blokové schéma detekce objektu

1) Předzpracování obrazu – aby byl obraz využitelný pro danou metodu, je nutné jeho předzpracováni. Mezi úkony předzpracování patří filtrace šumu, vyvážení bílé barvy, úprava jasu a úprava histogramu
2) Segmentace – jedná se o jeden z nejtěžších kroků. Při segmentaci oddělujeme hledané objekty od jejich pozadí za účelem dalšího zpracování. Segmentace může být částečná (získávání oblastí s lidskou kůží) nebo kompletní (rozdělení celého obrazu do částí).
3) Extrakce příznaků – příznakem jsou myšleny charakteristické rysy hledaného předmětu, jako například barva, vzdálenost od referenčního bodu v prostoru.
4) Klasifikace příznaků – proces třídění příznaků získaných v předcházejícím kroku do předem definovaných skupin.

2.2 Základní rozdělení metod

Metody využívané pro detekci obličejů v obraze jsou rozdělovány do dvou hlavních skupin, které se dále dělí do dalších dvou podskupin. Některé metody však mohou být zařazeny do více uvedených skupin [16] [28] [25] [10]:

1) **Strukturální přístup** – přístup založený na rozpoznávání charakteristických vlastností obličeje. Metody dále dělíme:
    a) **Metody založené na znalosti (Knowledge-based methods)** – obličej je vyhledáván na základě znalosti „typického obličeje“ Obličej je reprezentován typickými částmi, jako jsou oči, nos, ústa a mezi nimi jsou dány určité geometrické vztahy. Úspěšnost těchto metod je velice závislá na natočení obličeje v obraze a na výrazu tváře. Pro každé nové natočení a výraz je potřeba zavádět další pravidla, čímž stoupá složitost a robustnost algoritmů. Tyto metody jsou určené především k lokalizaci obličejů.
    b) **Metody založené na invariantních rysech (Feature invariant approaches)** – na rozdíl od znalostních metod, tyto metody využívají pro vyhledání obličeje takové rysy, které nejsou závislé na světelných podmínkách ani na natočení obličeje, tedy jsou neměnné (invariantní). Problémy může způsobit například špatné vyvážení barev nebo šum. Tyto metody jsou určené především k lokalizaci obličejů.
2) **Holistický přístup** – přístup založený na porovnávání vstupního obrazu s určitým vzorem. Metody dále dělíme:
    a) **Metody založené na srovnávání šablon (Template matching methods)** – tato metoda využívá předem vytvořené šablony obličeje nebo jeho částí a jejich korelace se vstupním obrazem. Šablony mohou být ručně vytvořeny, což je velice pracné a časově náročné, nebo jsou parametrizovány vhodnou funkcí.
    b) **Metody založené na zjevu (Appearance-based methods)** – metody pracující na podobném principu, jako metody založené

na srovnávání šablon, ale modely obličeje jsou získávány učením z trénovacích obrazů. Detekce probíhá stejně jako v předchozí metodě srovnáváním obrazu s modelem obličeje.

Dále se bude práce zabývat pouze dvěma metodami, které v současnosti vykazují nejlepší výsledky a jsou tudíž nejvyužívanější. Jedná se o jednu metodu od každého přístupu. Ze strukturálního přístupu jde o metody založené na invariantních rysech a z holistického přístupu o metody založené na zjevu.

3 Metody založené na invariantních rysech

Původní myšlenka těchto metod byla, že musí existovat určité shodné znaky pro všechny obličeje, na jejichž základě jsou rozpoznávány člověkem. Další nutnou vlastností byla stálost rysů při změně úhlu pohledu a světelných podmínek. Základním neměnným znakem je přítomnost očí, nosu a úst a jejich podobná poloha v obličeji u všech osob. Barva kůže, obočí, textura a tvar obličeje mohou být dalšími invariantními znaky využitelnými pro detekci obličeje. Některé metody mohou tyto přístupy kombinovat. Například nejdříve dojde k detekci barvy kůže a poté se podle pozice očí či úst určí, kde přesně se obličej nachází. Společným problémem pro tyto metody je velká citlivost na deformaci obrazu šumem, příliš velkým jasem a zakrytí detekovaného objektu. [25] [20].

3.1 Obličejová textura

Technika využívající poznatku, že obličej má v porovnání s předměty v našem okolí specifickou strukturu (texturu). Na základě tohoto poznatku mohou být obličeje od okolí separovány. Této metody se ale využívá spíše u složitějších realizací [28] [25].

3.2 Obličejové rysy

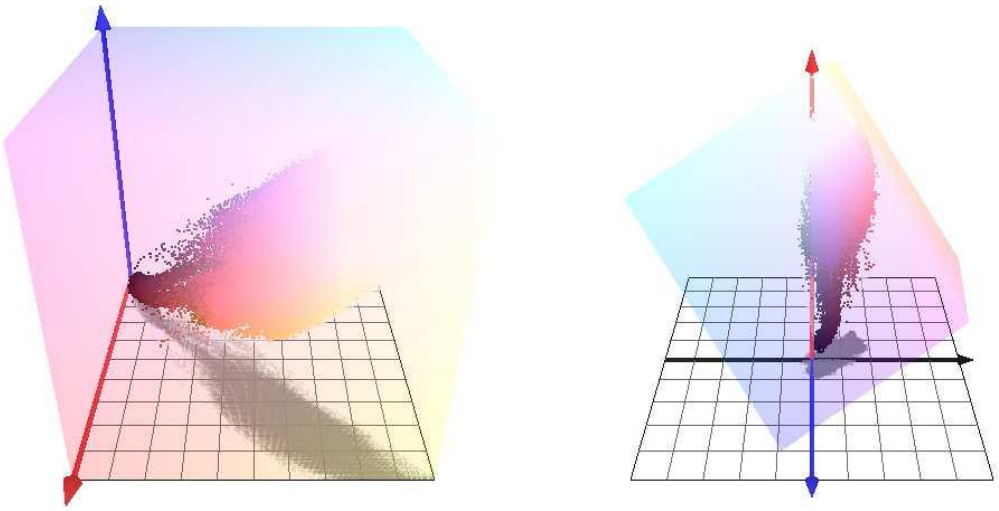
Technika zaměřující se na vyhledávání základních rysů obličeje, jako jsou oči, obočí, nos a ústa. Tyto rysy jsou z obrazu získány pomocí různých filtračních metod, hranových detektorů, morfologických operací nebo prahování. Pokročilejší techniky dosahují rychlejší detekce díky znalosti polohy jednotlivých rysů na obličeji, jako například, že nos se nachází mezi očima a ústy. Pokud tyto podmínky nejsou splněny, objekt nemůže být klasifikován jako obličej. Problémy nastávají v prostředích s velkým počtem hran, které mohou způsobit velké množství chybně pozitivních detekcí hranového detektoru [10] [20].

3.3 Barva obličeje

Pro svoji jednoduchost, rychlost a odolnost vůči změnám natočení obličeje, stínům, částečným zakrytím a komplexním prostředím je tato technika hojně využívána v mnoha detektorech. Vedle detekce obličeje byla využita například pro detekci nahých lidí, sledování pohybu rukou ad. Zásadní otázkou

při detekci barvy obličeje, je výběr barevného prostoru. Barevných prostorů bylo vyzkoušeno mnoho, například RGB, normalizovaný RGB, HSV, HLS, YCbCr a YIQ. Který z modelů je však ten nejvhodnější, nelze jednoznačně určit kvůli odlišnému chování klasifikátoru u každého barevného modelu (viz obr. 3.1) [16] [28] [8].

a) b)



Obr 3.1: Podprostor barvy kůže v barevném prostoru a) RGB b) YCbCr [25]

Po výběru barevného modelu je každý pixel v obraze porovnáván se zvolenými kritérii a označen jako obličejový nebo neobličejový. Tato kritéria představují většinou vzdálenost barvy určovaného pixelu k barvě kůže. Kritéria mohou být volena následujícími způsoby [26]:

1) **Explicitně definovaná oblast** – interval podprostoru barvy kůže je jasně definovaný. Interval je určen na základě rozložení barev v předložené šabloně (viz obr. 3.2). Klasifikátor není flexibilní při náhlých změnách podmínek osvětlení.
2) **Neparametrický model** – v tomto případě není podprostor barvy kůže striktně daný intervalem, ale je určen pravděpodobnostní mapou. Každému pixelu je následně přiřazena hodnota pravděpodobnosti, že se jedná o kůži. Pravděpodobnostní mapy jsou opět určovány na základě rozložení barev v trénovacích šablonách. Mezi neparametrické metody patří například Normalizovaná tabulka

Lookup Table, Bayesův klasifikátor, nebo metoda založená na Kohonenových neuronových sítích.

3) **Parametrický model** – interval podprostoru barvy kůže je určen na základě matematického modelu, jehož parametry jsou získávány z trénovacích šablon. Ve většině modelů má tento podprostor eliptický tvar a normální rozložení. Oproti neparametrickým modelům jsou schopné interpolovat chybějící data trénovací šablony. Nevýhodou je větší výpočetní náročnost.

Obr 3.2: Šablona barvy kůže [26]

Při detekci barvy kůže nám detektor najde veškerou kůži v obraze a případně i některé odstíny barvy dřeva, což je samozřejmě nežádoucí. Drobnější nechtěné detekce je možné odstranit pomocí morfologických operací, avšak u větších ploch nastávají problémy. Proto většina systémů pro detekci obličeje využívá kombinaci detekce barvy kůže s detekcí obličejových rysů, případně jeho textury

3.4 Vícenásobné rysy

Tyto metody jsou kombinací metod uvedených v předchozích kapitolách. Nejdříve jsou na základě detekce barvy kůže a znalosti tvaru obličeje určeny kandidátní oblasti, a poté je ověřena správnost nalezených kandidátních oblastí pomocí detekce obličejových rysů.

4 Metody založené na zjevu

Na rozdíl od metod založených na srovnávání šablon, kde jsou šablony definovány na základě expertních znalostí, je klasifikace obrazu realizována pomocí strojového učení a statistické analýzy. Klasifikátor určuje, zda se ve vybrané oblasti obrazu vyskytuje obličej či ne. Klasifikace probíhá formou porovnávání vstupního obrazu s naučeným modelem obličeje, který je uchováván ve formě diskriminantních funkcí nebo pravděpodobnostních modelů. S těmito modely je spojena snaha o redukci dimenzí příznakového prostoru za účelem zvýšení účinnosti výpočtu a tím i samotné detekce [28].

4.1 Extrakce příznaků

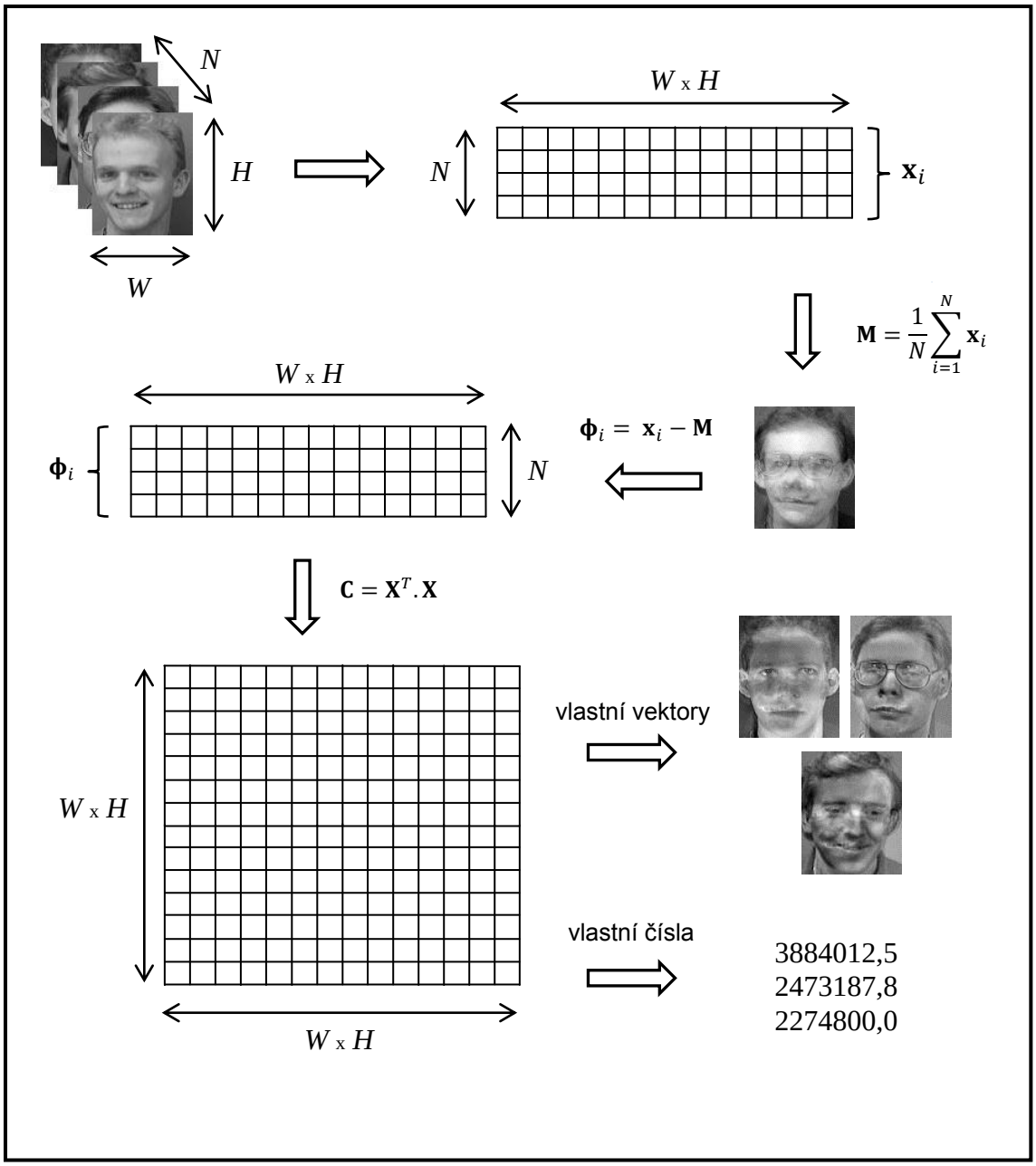
4.1.1 PCA (Principal Component Analysis)

Technika analýzy hlavních komponent je statistická metoda redukce dimenze příznakového prostoru se snahou o co nejmenší ztrátu informace. Hlavní myšlenkou této metody je extrakce příznaků ze vstupní sady dat, jejich následné promítnutí do vytvořeného podprostoru a porovnání s již promítnutou trénovací databází. V matematické terminologii lze myšlenku shrnout jako snahu o nalezení vlastních vektorů kovarianční matice vytvořené ze sady trénovacích obrazů. Tyto vlastní vektory představují soubor vlastností charakterizujících odlišnosti jednotlivých obličejů, a protože jsou podobné obrazům obličeje, říká se jim eigenfaces.

Rozpoznávání obličejů metodou PCA se skládá ze tří částí. Prvním krokem je vytvoření podprostoru PCA, dále pak promítnutí trénovací sady dat do vytvořeného podprostoru a následné rozpoznávání testovaných obrázků.

1) Vytvoření podprostoru

V první fázi jsou nejdříve všechny obrázky trénovací sady dat převedeny na řádkové vektory $\mathbf{x}_i$ a je z nich složena matice o počtu řádků N (počet obrázků v trénovací sadě) a počtu sloupců $n = W \times H$ (rozměr jednoho obrázku).



Obr 4.1: Výpočet podprostoru PCA

Podle vztahu 4.1 je vypočítán průměrný obrázek ve vektorovém tvaru, kde $\mathbf{M}$ je řádkový vektor o počtu sloupců n. Tento vektor je reprezentací průměrné tváře z trénovací sady dat a od každého trénovacího obrázku je odečten (viz rovnice 4.2), čímž vznikne sada vektorů $\boldsymbol{\phi}_i$, přičemž $\mathrm{i} = [1,2,\ldots,N]$.

$$\mathbf{M} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{4.1}$$

$$\boldsymbol{\phi}_i = \mathbf{x}_i - \mathbf{M} \tag{4.2}$$

Těmito operacemi vznikne matice $\mathbf{X}$, jejíž každý z N řádků obsahuje jeden vektor $\boldsymbol{\phi}_i$ délky n. Z matice $\mathbf{X}$ je podle vztahu vypočtena

$$\mathbf{S} = \mathbf{X}.\mathbf{X}^T = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\phi}_i.{\boldsymbol{\phi}_i}^T \tag{4.3}$$

kovarianční matice $\mathbf{S}$ o rozměru $N \times N$.

Jak bylo zmíněno na začátku této pod kapitoly, úkolem je najít vlastní vektory kovarianční matice. Hledané vlastní vektory však nejsou z výše uvedené kovarianční matice $\mathbf{S}$, ale z kovarianční matice $\mathbf{C}$, která vznikne prohozením násobených matic $\mathbf{X}$, tedy

$$\mathbf{C} = \mathbf{X}^T.\mathbf{X} \tag{4.4}$$

Tato kovarianční matice má však rozměr $n \times n$ a je pro výpočty kvůli své velikosti naprosto nevhodná. K výpočtu vlastních vektorů této kovarianční matice je možné dojít i pomocí kovarianční matice $\mathbf{S}$, a to následujícím způsobem. Z kovarianční matice $\mathbf{S}$ vypočítat pomocí vztahu

$$\lambda_i.\mathbf{v_i} = \mathbf{S}.\mathbf{v_i} \tag{4.5}$$

vlastní vektory $\mathbf{v_i}$ a vlastní čísla λ_i. Vlastní čísla jsou pro matice $\mathbf{S}$ i $\mathbf{C}$ shodná [21]. Dále je možné podle [21] odvodit následující vztah mezi vlastními čísly obou kovariančních matic:

$$\mathbf{u_i} = \mathbf{v_i}.\mathbf{X} \tag{4.6}$$

Kde $\mathbf{u_i}$ je i-tý vlastní vektor kovarianční matice $\mathbf{C}$ o rozměru $1 \times n$ a $\mathbf{v_i}$ je i-tý vlastní vektor kovarianční matice $\mathbf{S}$ o rozměru $1 \times N$. Vypočtené vektory $\mathbf{u_i}$ je dále nutné seřadit podle velikosti vlastních čísel od největšího po nejmenší a normalizovat podle rovnice 4.7 na jednotkovou velikost.

$$\mathbf{u_i} = \frac{\mathbf{u_i}}{\|\mathbf{u_i}\|} \tag{4.7}$$

Podle hodnot vlastních čísel λ_i (eigenvalues) jednotlivých vlastních vektorů $\mathbf{u_i}$ (eigenvectors) je poznat, nakolik daný vektor přispívá k popisu odchylek mezi obrázky. Malá vlastní čísla je proto možné ignorovat a ponechat pouze k vybraných vlastních vektorů. Tímto je dosaženo potřebné redukce z n-dimenzionálního prostoru do prostoru k-dimenzionálního. Složením všech vektorů $\mathbf{u_i}$ po řádcích do matice je získána hledaná matice vlastních vektorů $\mathbf{U}$ o rozměru $N \times n$. Jak již bylo řečeno, tyto jednotlivé vektory jsou nazývány eigenfaces a slouží jako báze hledaného podprostoru PCA, takzvaného eigenspace [4] [9] [21] [3].

2) Promítnutí trénovací sady

Je-li vytvořen podprostor PCA s redukovanou dimenzí, nic nebrání promítnutí trénovacích obrázků do tohoto podprostoru. Od promítaného obrázku $\mathbf{x}_i$ je nejdříve odečten průměrný obrázek $\mathbf{M}$ a následně je vynásoben transponovanou maticí vlastních vektorů $\mathbf{U}^T$ viz rovnice 4.8.

$$\mathbf{z}_i = (\mathbf{x}_i - \mathbf{M}).\mathbf{U}^T \tag{4.8}$$

Výsledkem těchto operací je matice $\mathbf{Z}$, jejíž každý řádek obsahuje vektor $\mathbf{z}_i$ o rozměru $1 \times k$, přičemž $\mathrm{i} = [1,2,\ldots,N]$. Každý z těchto vektorů reprezentuje jeden z trénovacích obrázků a obvykle bývá nazývaný jako vektor tvaru vah [3]. Jednotlivé sloupce každého vektoru představují příspěvek daného k-tého vlastního vektoru pro reprezentaci i-tého trénovacího obrázku [9] [21] [3].

3) Rozpoznávání testovaných obrazů

V závěrečné fázi je možné přistoupit k samotnému procesu rozpoznávání neznámých obličejů o počtu M. První kroky jsou totožné s kroky při promítání trénovací sady obrázků. Nejdříve je od každého rozpoznávaného obrázku $\mathbf{y}_\mathrm{j}$,

kde j $= [1,2, \ldots, M]$, odečten průměrný obrázek $\mathbf{M}$ a následně je vynásoben s transponovanou maticí $\mathbf{U}^T$, obdobně jak je uvedeno v rovnici 4.8. Takto vzniklé vektory $\mathbf{p}_\mathrm{j}$ o rozměru $1 \times k$ jsou poskládány do matice $\mathbf{P}$ o rozměru $M \times k$.

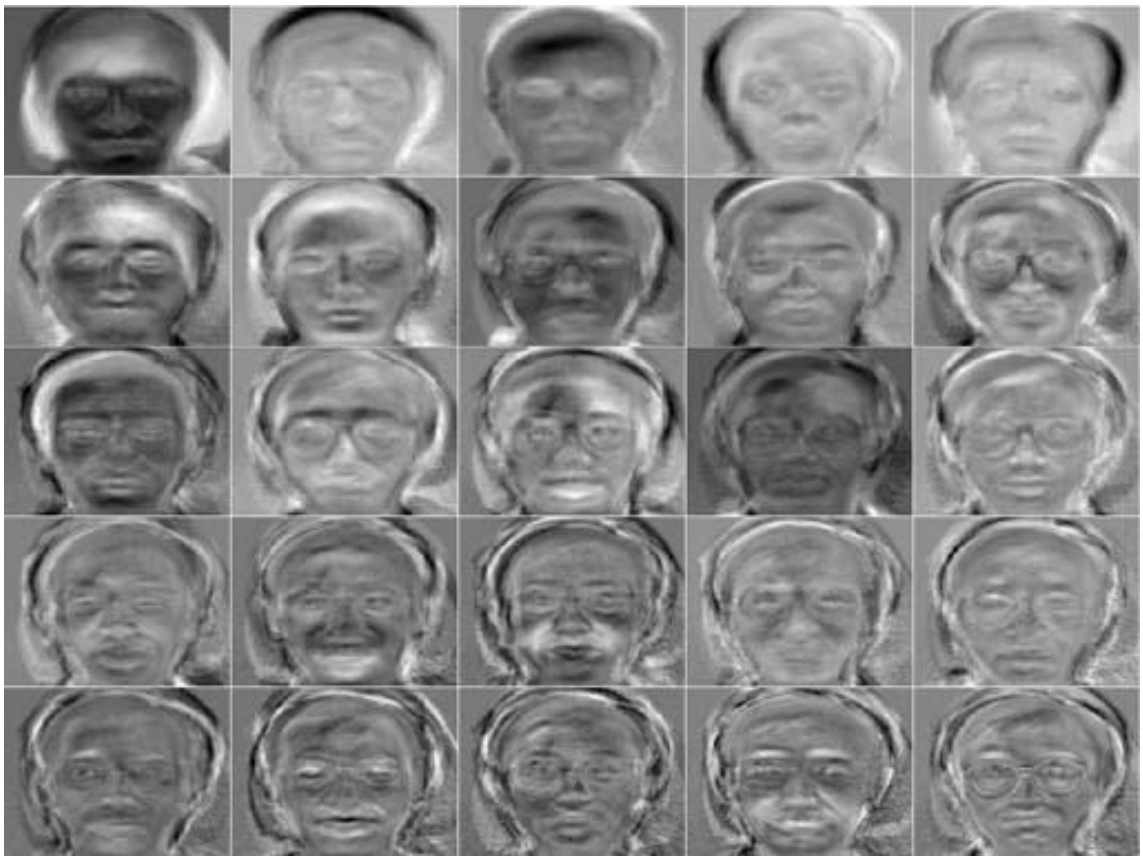
Nyní je nutné nalézt minimální vzdálenost mezi projekcí $\mathbf{p}_j$ rozpoznávaného obrázku y_j a promítnutými trénovacími obrázky $\mathbf{z}_i$ v matici $\mathbf{Z}$. Pro výpočet minimální vzdálenosti lze využít například vzorec pro Euklidovskou vzdálenost (rov. 4.9), Hammingovu vzdálenost (rov. 4.10), nebo vzdálenost na šachovnici (rov. 4.11) [14].

$$D_e(\mathbf{p}_j, \mathbf{z}_i) = \sqrt{\sum_{l=0}^{k-1} \left(\mathbf{p}_j[l] - \mathbf{z}_i[l]\right)^2} \tag{4.9}$$

$$D_4(\mathbf{p}_j, \mathbf{z}_i) = \sum_{l=0}^{k-1} \left|\mathbf{p}_j[l] - \mathbf{z}_i[l]\right| \tag{4.10}$$

$$D_8(\mathbf{p}_j, \mathbf{z}_i) = max \left|\mathbf{p}_j[l] - \mathbf{z}_i[l]\right| \tag{4.11}$$

Podle nalezené minimální vzdálenosti je určen obličej x_i z trénovací sady nejlépe odpovídající rozpoznávanému obličeji y_j. Aby bylo co nejefektivněji předcházeno chybnému rozpoznání obličeje, je nutné porovnat nalezenou hodnotu minimální vzdálenosti s hodnotou prahu T. Pokud platí, že hodnota $D <$ T, pak je možné předpokládat, že hledaný obličej byl natrénován a s největší pravděpodobností i správně rozpoznán. Pokud však nalezená minimální vzdálenost nesplňuje tuto podmínku, je vysoce pravděpodobné chybné rozpoznání, a proto je daný obličej označený jako „neznámý“ [9] [21] [3].



Obr 4.2: Ukázka eigenfaces trénovací sady [15]

4.1.2 LDA (Linear Discriminant Analysis)

Tato statistická metoda, na rozdíl od metody PCA, během transformace maximalizuje rozptyl dat mezi jednotlivými třídami (různé osoby) a minimalizuje rozptyl dat uvnitř daných tříd (stejná osoba).

Vstupem je sada N obrázků, které se dají rozdělit podle různých vlastností do L tříd. Tato vstupní sada dat je, stejně jako u metody PCA, převedena na N jednorozměrných vektorů. Dále jsou vypočteny průměrné obrázky jednotlivých tříd $\mathbf{M}_l$ a celkový průměrný obrázek $\mathbf{M}$, vše ve vektorovém tvaru:

$$\mathbf{M}_l = \frac{1}{J}\sum_{j=1}^{J}\mathbf{x}_{l,j} \tag{4.12}$$

$$\mathbf{M} = \sum_{l=1}^{L} p_l.\mathbf{M}_l \tag{4.13}$$

Hodnota J udává množství obrázků v dané třídě l, $x_{l,j}$ je j-tý vektor z třídy l a p_l udává pravděpodobnost výskytu třídy l. Matice rozptylu mezi třídami je definována:

$$\mathbf{S}_\mathrm{b} = \frac{1}{L}\sum_{l=1}^{L}(\mathbf{M}_l - \mathbf{M})\,(\mathbf{M}_l - \mathbf{M})^T \tag{4.14}$$

A matice rozptylu uvnitř tříd:

$$\mathbf{S}_w = \frac{1}{LJ}\sum_{l=1}^{L}\sum_{j=1}^{J}\left(\mathbf{x}_{l,j} - \mathbf{M}_l\right)\left(\mathbf{x}_{l,j} - \mathbf{M}_l\right)^T \tag{4.15}$$

Požadovaného efektu je dosaženo maximalizací poměru determinantu matice $\mathbf{S}_b$ a determinantu matice $\mathbf{S}_w$, přičemž matice $\mathbf{S}_w$ musí být regulární (čtvercová matice s nenulovým determinantem a lineárně nezávislými řádky a sloupci). Maximalizace poměru determinantů je realizována maticí ortonormálních vektorů $\mathbf{W}_{opt}$,

$$\mathbf{W}_{opt} = argmax\,\frac{\mathbf{W}^T\mathbf{S}_b\mathbf{W}}{\mathbf{W}^T\mathbf{S}_w\mathbf{W}} = [w_1, w_2, \ldots w_m] \tag{4.16}$$

která znázorňuje optimální projekci, w_i je množina vlastních vektorů (eigenvectors) $\mathbf{S}_B$ a $\mathbf{S}_w$ a matice $\mathbf{W}$ obsahuje vlastní vektory získané z trénovací sady metodou PCA. Z matic $\mathbf{W}^T\mathbf{S}_b\mathbf{W}$ a $\mathbf{W}^T\mathbf{S}_w\mathbf{W}$ jsou vypočteny zobecněné vlastní vektory, které se následně sestupně seřadí. Projekce trénovacích a testovacích dat je převedena do této množiny vektorů a po klasifikaci je obdobně jako u metody PCA vypočítá nejkratší vzdálenost.

LDA dosahuje lepších výsledků než PCA například při drastických změnách osvětlení a výrazu tváře [9] [16] [12].

4.2 Klasifikátory

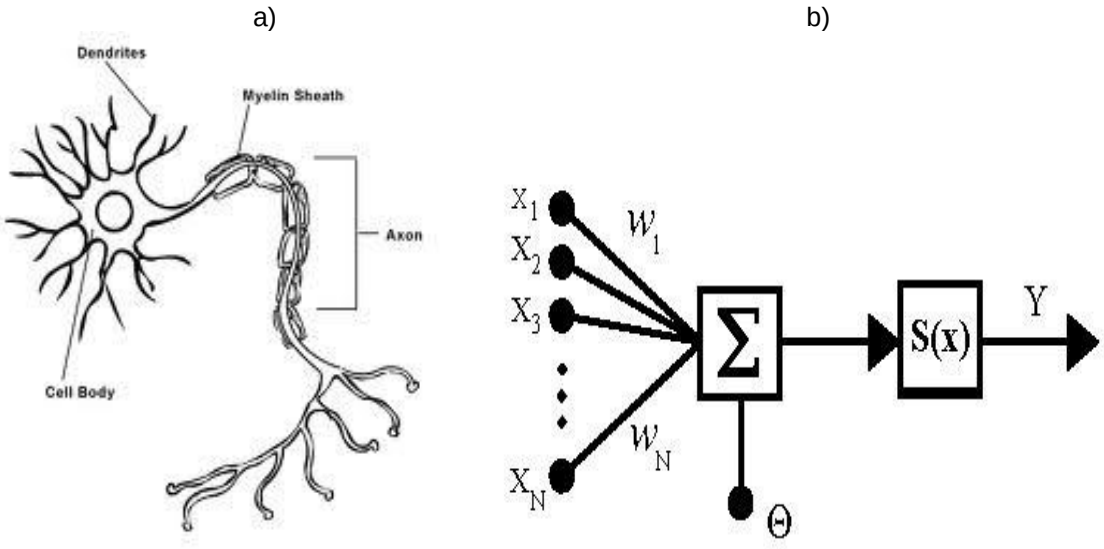
Jedná se o algoritmy hodnotící pravděpodobnost, s jakou se ve vstupních datech vyskytuje hledaný objekt, a následně tato data rozřadí do předem definovaných kategorií. Nejčastěji využívané klasifikátory jsou binární, jejichž výstupem je pouze informace, zda se ve vstupu hledaný objekt nachází či ne. Vícestupňové klasifikátory rozdělují data podle míry pravděpodobnosti výskytu hledaného objektu.

4.2.1 Neuronové sítě

Jedním z oblíbených klasifikátorů jsou neuronové sítě. Jedná se o paralelní systém vyvinutý na základě znalostí chování biologických neuronových sítí. Umělý neuron je postaven na stejném principu jako neuron biologický (viz obr. 4.3). Skládá se z N vstupů (dendridů) označených x_N, sumy (jádra), výstupu Y (axonu), synaptických vah w_N, váhy ovlivňující práh rozhodování θ a přenosové funkce neuronu $S(x)$ [13].

Neuronová síť je tvořena množstvím vzájemně propojených neuronů, které vytvářejí tři druhy vrstev:

a) Vstupní vrstva – neurony přijímající vstupní data ke zpracování
b) Skryté vrstvy – neurony spojující vstupní a výstupní vrstvu
c) Výstupní vrstva – neurony vysílající zpracovaná data



Obr 4.3: a) Biologický neuron a b) umělý neuron [13]

Samotné neuronové sítě se dělí podle způsobu propojení výše zmíněných vrstev na [10]:

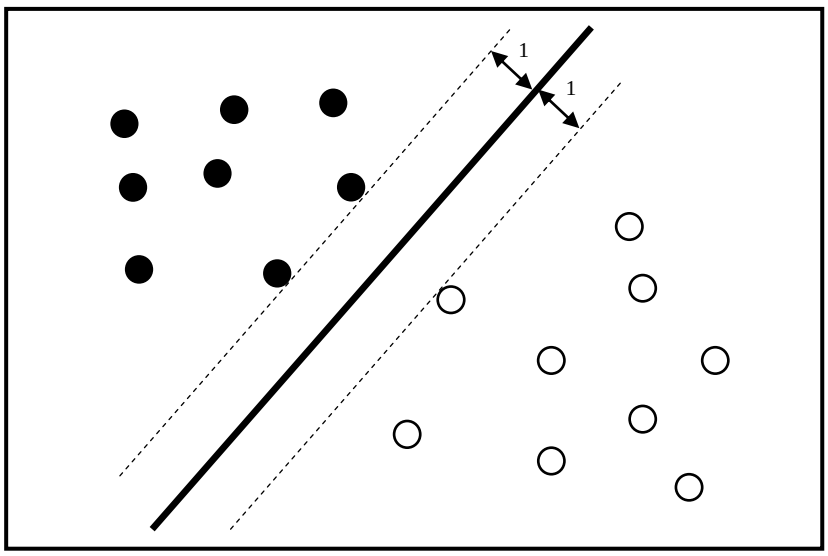
a) Dopředné sítě – v těchto sítích se data šíří přímo ze vstupu na výstup, neexistují zde smyčky. Nejjednodušším typem je například jednovrstvý perceptron
b) Rekurentní sítě – zde se již mohou vyskytovat smyčky a zpětné vazby, data se tudíž nemusí pohybovat pouze ze vstupu na výstup. Příkladem je Hopfieldova síť.

Trénování neuronových sítí probíhá formou úpravy jednotlivých vah neuronů tak, aby bylo dosaženo co nejlepších výsledků. V oboru rozpoznávání tváří je asi nejznámější trénovací technika Back-propagation ve spojení s PCA [11].

4.2.2 SVM (Support Vector Machine)

Dalším často využívaným klasifikátorem je SVM. Jedná se o metodu klasifikace lineárních dat, která pracuje na principu co nejlepšího rozdělení *n*-rozměrných dat pomocí *n-1* rozměrné hyperplochy.

Na obrázku 4.4 je zobrazen příklad rozdělení dat do dvou tříd pomocí SVM. Plnou čarou je zobrazena dělící hyperplocha, která maximalizuje vzdálenost jednotlivých tříd. Přerušovanou čarou jsou znázorněné Support Vectory, které dokazují, že podmínka maximální vzdálenosti dělící hyperplochy od nejbližších dat jednotlivých tříd je splněna [9].

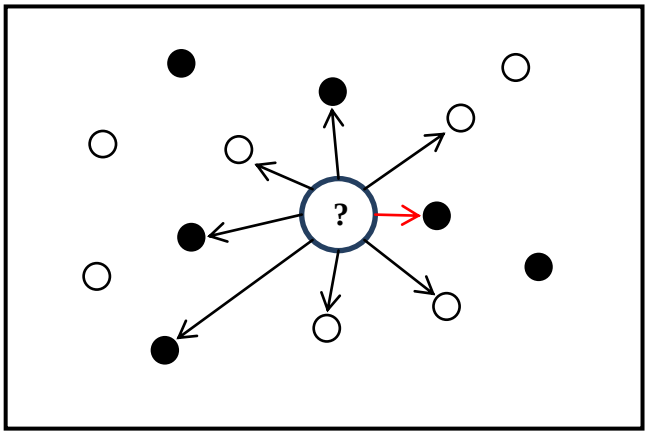


Obr 4.4: Separace metodou SVM

Dnes je již možné metodu SVM využít i pro klasifikaci lineárně neseparovatelných dat. Tato data je nejdříve nutné transformovat do prostoru s vyšší dimenzí a následně provést jejich separaci [10].

4.2.3 Nejbližší soused

Metoda nejbližšího souseda, označovaná jako *k*-NN (Nearest neighbor), je neparametrická klasifikační metoda. Vstupem jsou trénovací data, která se porovnávají s testovaným obrázkem. Klasifikace se provádí na základě nalezení *k* nejbližších sousedů. Výsledná třída je zvolena taková, do které patří většina z nalezených sousedů. Pro určení nejbližších sousedů se nejčastěji využívá výpočet pro Euklidovskou (rov. 4.9) nebo Hammingovu (rov. 4.10) vzdálenost.



Obr 4.5: Hledání nejbližšího souseda (1-NN)

Nejčastější klasifikací je případ 1-NN, tedy klasifikace podle jednoho souseda (viz obr. 4.5). Nejdříve je zjištěna vzdálenost všech prvků od testovaného, následně je vybrán jemu nejbližší, a testovaný prvek přiřazen do stejné třídy [6].

5 Detektor Viola & Jones

Tento objektový detektor pochází od autorů Paula Violy a Michaela Jonese a byl jimi uveřejněn roku 2001 v dokumentu [23]. Detektor se stal úspěšným pro svou rychlost, spolehlivost a značnou nezávislost na osvětlení. Těchto vlastností je dosaženo pomocí třech základních částí:

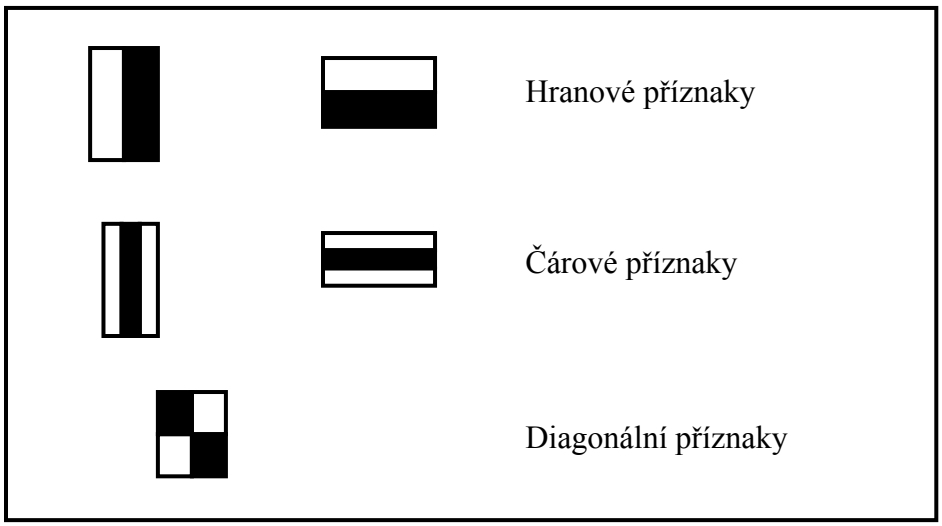
a) AdaBoost klasifikační algoritmus
b) Haarovy příznaky
c) Integrální obraz

5.1 Haarovy příznaky

Haarovy příznaky jsou využity jako vstup do klasifikačního algoritmu. Jedná se o obdélníkové oblasti, které se přikládají na vstupní obraz. Hodnota příznaku je následně získána odečtením sumy intenzit i pixelů obrazu pod černou částí $(x(b))$ oblasti od sumy intenzit j pixelů obrazu pod bílou částí $(x(w))$:

$$f(x) = \sum_{w=1}^{i} x(w) - \sum_{b=1}^{j} x(b) \tag{5.1}$$

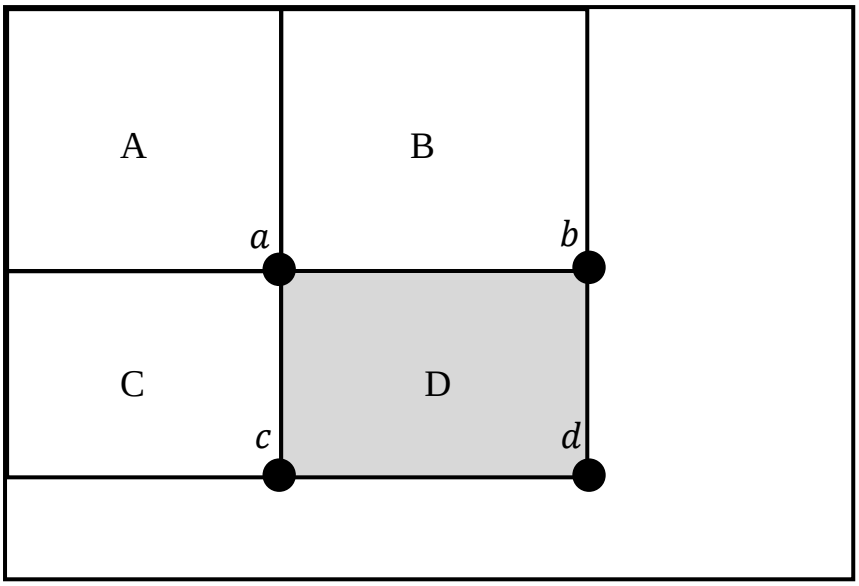
Haarových příznaků je velká řada a dělí se do následujících třech skupin (viz obr. 5.1). Zde jsou uvedeny nejvyužívanější druhy z každé skupiny, ostatní druhy jsou vytvořeny pomocí jejich rotace.



Obr 5.1: Haarovy příznaky [9]

5.2 Integrální obraz

Pro zjednodušení výpočtu příznaků vstupního obrazu se využívá integrálního obrazu. Jedná se o reprezentaci vstupního obrazu, ve které jsou kumulativně sečteny hodnoty intenzit pixelů ve všech řádcích a sloupcích nalevo od pozice aktuálního pixelu. Z toho plyne, že pixel umístěný v pravém dolním rohu nese hodnotu součtu intenzit všech pixelů v obraze.



Obr 5.2: Oblast pro výpočet z rov. 5.2

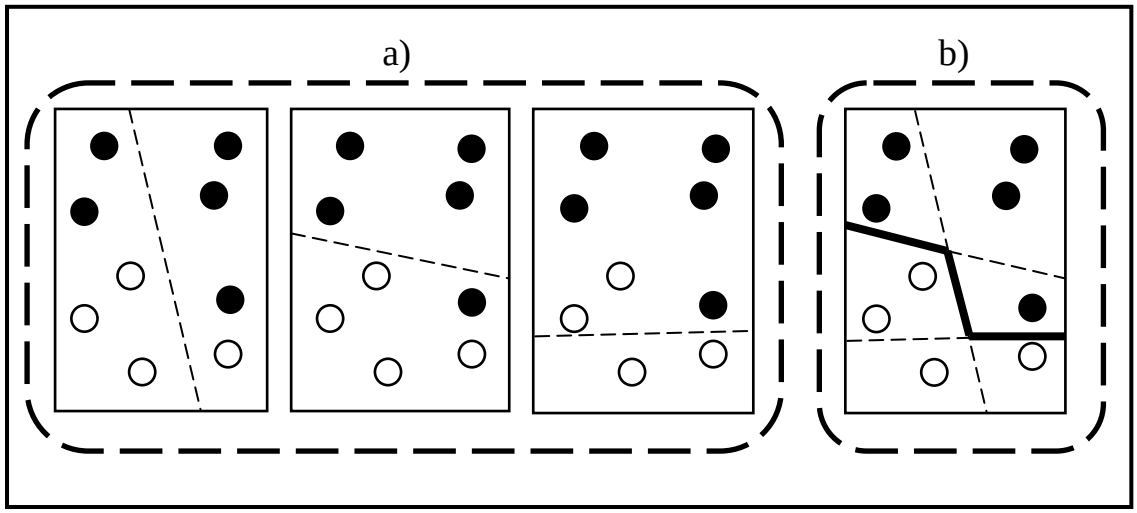
Zjednodušení spočívá ve výpočtu hodnot intenzit pixelů v obdélníku uvnitř obrazu, jak je vidět na obrázku 5.2. Hodnota integrálního obrazu v bodu a odpovídá součtu intenzit pixelů v obdélníku A. Hodnota v bodu b odpovídá pixelům v A+B, v bodu c pixelům v A+C a v bodu d pixelům v A+B+C+D. Pro výpočet hodnoty integrálního obrazu v obdélníku D stačí pouze dvě operace sčítání a jedna operace odčítání [24]:

$$I_D = d + a - (b + c) \tag{5.2}$$

5.3 AdaBoost

Jako klasifikační algoritmus je v detektoru Viola & Jones využit algoritmus AdaBoost (Adaprive Boosting). Účelem algoritmu je co největší zesílení (boosting) klasifikace, tedy výběr nejvhodnějších příznaků pro vytvoření klasifikátoru. Vstupem pro algoritmus jsou jednotlivé příznaky z trénovací

množiny spárované s informací o třídě příznaku. Následně jsou všem příznakům rovnoměrně nastaveny váhy. Tímto vzniknou tzv. slabé klasifikátory $h(x_i)$ (viz obr. 5.3a), jejichž přesnost je jen o něco málo vyšší než přesnost odhadu, což je přes 50%. V každé následující iteraci je vybrán klasifikátor s nejmenší chybou, která ovšem nesmí přesáhnout hodnotu 0,5, a podle něj nastaveny váhy ostatních tak, aby klasifikátor s největší chybou měl největší váhu. Tímto je dosaženo toho, že v dalším kroku bude hledán takový slabý klasifikátor, který bude lépe klasifikovat chybná měření [16]. Požadované efektivity je dosaženo následnou kombinací těchto k slabých klasifikátorů, z čehož vznikne silný klasifikátor s požadovanou přesností (viz obr. 5.3b). Hodnota silného klasifikátoru $H(x)$ je rovna součtu všech slabých klasifikátorů $h(x_i)$, které zahrnuje, vynásobených jejich vahami $w(x_i)$ (viz rov. 5.3) [10].

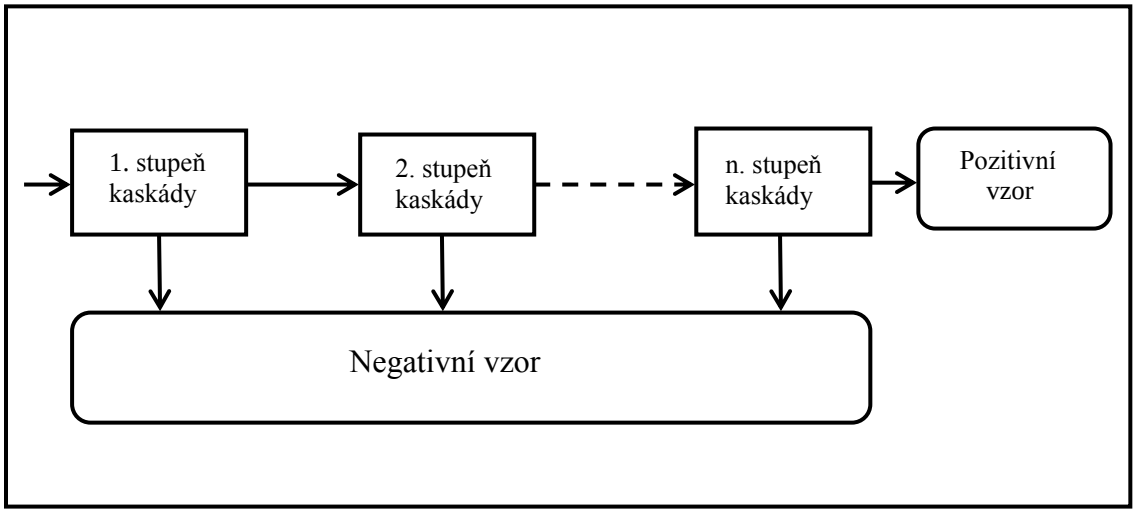


Obr 5.3: a) Slabé klasifikátory b) silný klasifikátor

$$H(x) = \sum_{i=1}^{k} h(x_i).w(x_i) \tag{5.3}$$

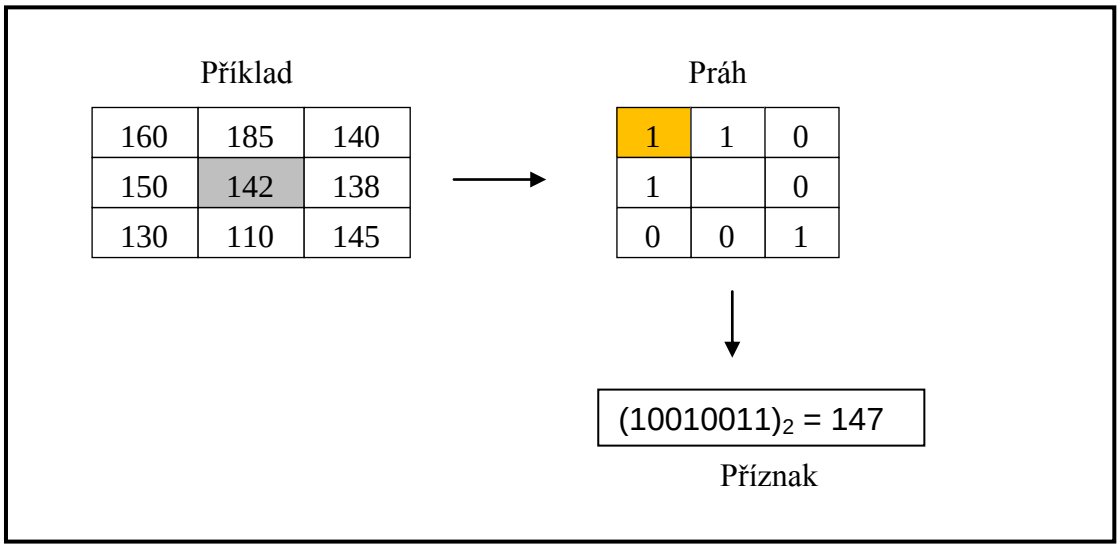
5.4 Kaskádové zapojení klasifikátorů

Podstatného zefektivnění práce detektoru Viola & Jones je dosaženo kaskádovým zapojením jednotlivých klasifikátorů (viz obr. 5.4). Vzhledem k tomu, že se obraz nezpracovává jako celek, ale po dílčích obdélníkových oblastech, bylo by zbytečné dále pracovat s oblastmi, ve kterých hledaný objekt nebyl nalezen.



Obr 5.4: Kaskádové zapojení klasifikátorů [16]

Jednodušší a rychlejší klasifikátory jsou většinou umístěny na začátku kaskády a na konci bývají klasifikátory složitější a pomalejší. Tímto se značně šetří časová náročnost klasifikace, protože se náročnější klasifikátory již nemusí zbytečně zabývat oblastmi bez hledaného objektu. Každý z kaskádních klasifikátorů je trénován samostatně pomocí výše popsaného algoritmu AdaBoost.



Obr 5.5: LBP příznak

5.5 LBP příznaky

Ve snaze zpřístupnit algoritmy detekce a rozpoznávání obličejů i méně výkonným zařízením vznikla alternativa k Haarovým příznakům ve formě příznaků LBP (Local Binary Pattern). LBP příznak je získán pomocí prahování hodnot 8-okolí středovou hodnotou, jak je ukázáno na obrázku 5.5. V prvním

kroku jsou okolní hodnoty vyprahovány podle hodnoty středu a vzniklá posloupnost je přepsána do 8-bitového kódu, který je odezvou příznaku. Přepis posloupnosti začíná horním políčkem vlevo, které následně tvoří LSB ve vzniklém kódu. Následně jsou hodnoty přepisovány po směru hodinových ručiček. Oproti výpočtu Haarových příznaků je tento proces mnohem méně výpočetně náročný [7].

6 Implementace rozpoznávání obličejů

V této kapitole budou popsány vlastní praktické výsledky diplomové práce, tedy aplikace pro rozpoznávání obličejů v obraze a ve videosekvencích. V první části kapitoly budou krátce přiblíženy vlastnosti implementačního prostředí balíčku OpenCV a dále jednotlivé bloky aplikace tak, jak jdou za sebou ve vývojovém diagramu na obr. 6.2. Při psaní aplikace byla snaha o dosažení detekce a rozpoznávání obličejů ve videosekvenci v reálném čase. Metoda PCA, která je v aplikaci využita k rozpoznávání, patří k základním metodám a je podrobně rozebrána v kapitole 4.1.1

6.1 OpenCV

Aplikace je napsána v jazyce C++ za pomoci nejnovější verze balíčku OpenCV (Open Source Computer Vision Library), což je knihovna funkcí určených pro práci s obrazem a videem. Oficiální vývoj této knihovny zahájila v roce 1999 firma Intel, s jejímiž čipy byl také chod celé knihovny v počátcích spjat. Postupem času byla tato závislost odbourána a nyní je knihovna OpenCV zcela volně dostupná každému zájemci, jelikož se jedná o BSD-licencovaný software, stejně jako například operační systém Unix.

Aktuální verze, se kterou bylo pracováno, je verze 2.4.0, konkrétně pak její C++ API. OpenCV se skládá z několika následujících modulů [14]:

a) **core** – modul definující základní datové struktury a funkce, které jsou v OpenCV k dispozici, především vícedimensionální pole *Mat*.
b) **imgproc** – modul pro práci s obrazem obsahující například funkce pro lineární a nelineární filtraci obrazu, geometrické transformace, práci s histogramem, konverze mezi barevnými prostory a další
c) **video** – modul pro analýzu videa podporující funkce pro odhad pohybu, extrakci pozadí a sledování objektů ve videosekvencích.
d) **calib3d** – algoritmy pro kalibraci kamery, odhad pozice objektu, 3D rekonstrukce a další
e) **features2d** – modul obsahující například algoritmy pro detekci hran, extrakci příznaků a další

f) **objdetect** – detekce různých předdefinovaných objektů (např. tváří, očí, lidí, aut)
g) **highgui** – jednoduché rozhraní pro zachytávání videa, kodeky pro obraz a video a další funkce pro uživatelská rozhraní
h) **gpu** – algoritmy využívající GPU akceleraci

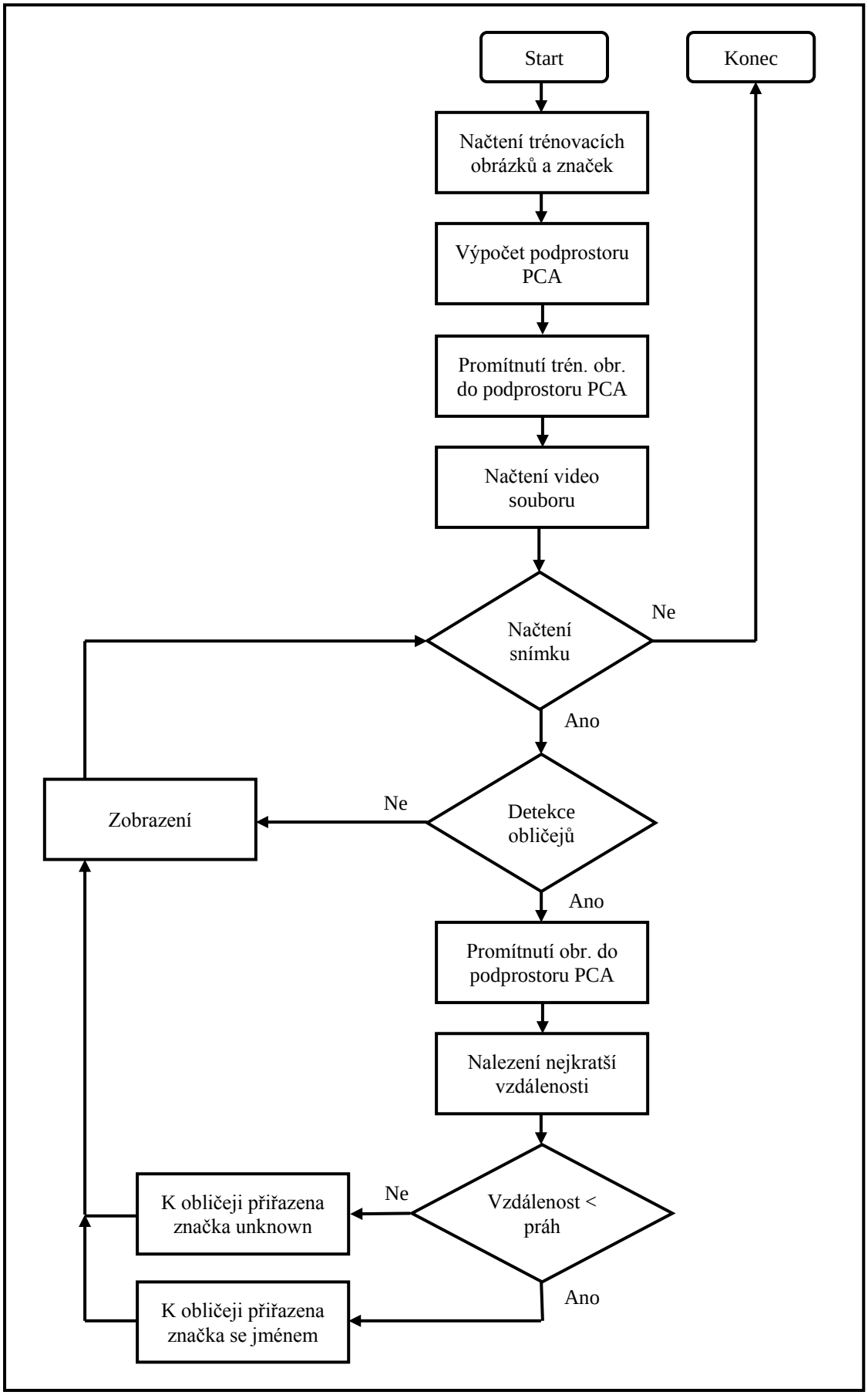
Jak je viděl z krátkého popisu grafické knihovny OpenCV, jedná se o velice praktický nástroj, v současnosti hojně využívaný především k výukovým účelům.

6.2 Vlastní aplikace

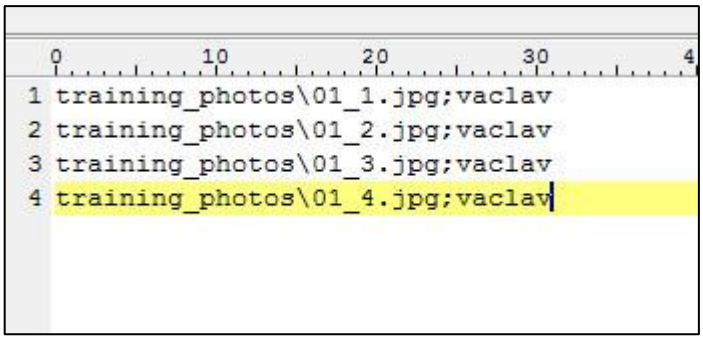
Pro lepší představu o funkčnosti aplikace je uveden vývojový diagram na obr. 6.2. Jsou zde vyobrazeny veškeré bloky, ze kterých se aplikace skládá, a jejich vzájemná provázanost. Další podkapitoly se budou postupně věnovat samostatně každému bloku a na obrázcích budou uvedeny části kódu, které se k danému bloku vztahují. Pro správnou funkčnost programu je zapotřebí mít k dispozici především sadu trénovacích obrázků obsahujících podoby hledaných osob (viz obr. 6.1).

Obr 6.1: Sada trénovacích obrázků pro jednu osobu

Dále pak *.csv soubor obsahující cesty k daným trénovacím obrázkům a k nim přidělené značky (viz obr. 6.3), testovací video a *.xml soubor s natrénovanou kaskádou klasifikátorů pro detektor obličejů. Při seznamování se s prostředím balíčku OpenCV a při psaní aplikace byly čerpány některé důležité poznatky z projektů [27] a [5]. Kvůli přehlednosti uváděných kódů z nich byly umazány podmínky ošetřující výjimečné stavy.



Obr 6.2: Vývojový diagram aplikace



Obr 6.3: Trénovací *.csv soubor

6.2.1 Načtení trénovacích obrázků a značek

Prvním důležitým blokem aplikace je načtení trénovacích obrázků a jejich příslušných značek ze vstupního souboru *.csv (viz kód 6.1). Vstupním údajem funkce je jednak název csv souboru, vektor matic typu `Mat` pro uložení trénovacích obrázků (faceImages) a vektor typu `string` pro uložení značek (faceLabels).

`loadFaceImages()`

```
1  void loadFaceImages (const string& filename, vector<Mat>& faceImages,
2                       vector<string>& faceLabels)
3  {
4    string row, path, label;
5
6    ifstream file (filename, ifstream::in);   //nacteni souboru
7
8    while (getline(file, row))
9    {
10     stringstream Face (row); //nacteni celeho radku
11     getline(Face, path, ';'); //nacteni cesty
12     getline(Face, label); //nacteni znacky (text za strednikem)
13     faceImages.push_back(imread(path, 0)); //nacteni obrazku
14     faceLabels.push_back(label); //nacteni znacek
15   }
16 }
17
```

Kód 6.1: Funkce `loadFaceImages()`

Po načtení celého souboru pomocí funkce `std::ifstream()` probíhá načítání jednotlivých řádků a do proměnných `path` a `label` jsou ukládány informace o cestě k obrázku a značka. Rozlišení značky a cesty je zajištěno jejich oddělením pomocí středníku (viz obr. 6.3), na kterém se zastaví načítání cesty (řádek 11). Řádky 13 a 14 zajišťují uložení obrázků a značek do připravených vektorů.

6.2.2 Výpočet podprostoru PCA

Celý postup výpočtu podprostoru PCA byl podrobně popsán v kapitole 4.1.1, zde je uvedeno, jak je zvolený postup řešen v aplikaci. Na vstupu druhého bloku aplikace je matice trénovacích obrázků, které jsou převedeny z vektoru matic do jediné matice pomocí funkce `createRowMatrix()` (viz kód 6.2). V této výsledné matici reprezentuje každý řádek jeden trénovací obrázek. V první fázi funkce zjistí požadované rozměry výsledné matice a následně ve druhé fázi bere postupně vstupní obrázky a pomocí funkce `cv::reshape()` z nich vytváří jednotlivé řádky výstupní matice.

createRowMatrix()

```
1 Mat createRowMatrix (const vector<Mat>& faceImages, int type)
2 {
3   int x = faceImages.size(); //pocet obrazku (pozadovane radky)
4   int y = faceImages[0].total(); //pocet hodnot obrazku (pozadovane sloupce)
5   Mat matice(x,y,type); //vystupni matice
6
7   //prevod z vector<Mat> do Mat struktury
8   for(int i = 0; i<x; i++)
9   {
10    Mat temp1, temp2 = matice.row(i);
11    faceImages[i].assignTo(temp1, type);
12    temp1.reshape(1,1).copyTo(temp2);
13  }
14  return matice;
15 }
16
```

Kód 6.2: Funkce `createRowMatrix()`

Dále je nutné zjistit, jaký je počet trénovacích obrázků, podle čehož se při výpočtu podprostoru PCA určuje hodnota k. Podle [19] vždy záleží na dané úloze, ale většinou se hodnota k určuje tak, aby kumulativní součet vlastních čísel odpovídajících daným vektorům dával 65-90% ze součtu vlastních čísel. Proto je volena hodnotu o jednu menší, než je počet trénovacích obrázků (viz kód 6.3). Pro samotný výpočet podprostoru PCA je využito již zabudovaných funkcí v OpenCV. Jedná se o funkci `PCA::operator()`, na jejímž výstupu je jednak průměrný obrázek $\mathbf{M}$ (*avgTrainImg*), matice vlastních vektorů $\mathbf{U}$ (`eigenVectors`) a matice vlastních čísel λ (`eigenValues`). Na vstupu této zabudované funkce musí být udáno, zda jsou vstupní data uspořádána do vektorů řádkových (CV_PCA_DATA_AS_ROW), či sloupcových (CV_PCA_DATA_AS_COL). Funkci lze nalézt v modulu core v souboru

`matmul.cpp` od řádku 2813. Pro výpočet kovarianční matice aplikace využívá OpenCV funkci `cv::calcCovarMatrix()` (soubor `matmul.cpp`, řádek 2138) a pro výpočet vlastních čísel a vektorů funkci `cv::eigen()` (soubor `lapack.cpp`, řádek 1429).

`train()`

```
1  vector<Mat> train (const Mat& faceImages)
2  {
3    vector<Mat> faceProjections; //vektor pro projekce obrazku
4    Mat temp;
5
6    int faceNum_train = faceImages.rows; //pocet trenovacich tvari
7
8    //vypocet podprostoru PCA
9    PCA doPCA (faceImages,Mat(),CV_PCA_DATA_AS_ROW, faceNum_train-1);
10
11   avgTrainImg = doPCA.mean; //prumerny obrazek "M"
12   eigenVectors = doPCA.eigenvectors; //vlastni vektory "U"
13   eigenValues = doPCA.eigenvalues; //vlastni čísla "lambda"
14
15   //projekce trenovacich obrazku do podprostoru PCA
16   for (int i = 0; i < faceNum_train; i++)
17   {
18     temp = projectFaces(faceImages.row(i));
19     faceProjections.push_back(temp); //ulozeni vypocitanych projekci do vektoru
20   }
21
22   return faceProjections;
23 }
```

Kód 6.3: Funcke `train()`

6.2.3 Promítnutí obrázků do podprostoru PCA

Jakmile je podprostor PCA vytvořen, mohou se do něj začít promítat vstupní trénovací obrázky, což se děje pomocí funkce `projectFaces()`, nacházející se v kódu 6.4.

`projectFaces()`

```
1  Mat projectFaces (const Mat& faceImage)
2  {
3    Mat P, P_final;
4
5    //odecteni vektoru prumerne tvare (P=faceImage-M)
6    subtract(faceImage, avgTrainImg.reshape(1,1), P);
7
8    //vypocet eigenface (P_final = P*V)
9    gemm(P, eigenVectors.t(), 1, Mat(), 0, P_final);
10
11   return P_final;
12 }
13
```

Kód 6.4: Funkce `projectFaces()`

Promítané obrázky jsou do funkce posílány jeden po druhém a postupně promítány podle vztahu 4.8. Pomocí funkce `cv::subtract()` je realizováno odečtení průměrného obrázku $\mathbf{M}$ od obrázku promítaného a následně proběhne vynásobení transponovanou maticí vlastních vektorů $\mathbf{U}^T$. Násobení matic zajišťuje funkce `cv::gemm()`. Výsledky promítání, což jsou vektory $\mathbf{z}_i$ jsou postupně ukládány do vektoru matic `faceProjections` (viz řádek 19 v kódu 6.3).

Tato funkce bude dále využita i pro promítnutí rozpoznávaných obrázků v pozdější fázi algoritmu.

`faceDetect()`

```
1  vector<Mat> faceDetect (const string& faceCascadeName, const Mat& faceImage)
2  {
3    CascadeClassifier face_cascade;
4
5    Mat faceImageGray;
6    vector<Mat> faceROI;
7
8    face_cascade.load(faceCascadeName); //nacteni natrenovane kaskady
9
10   //prevod obrazku do odstinu sedi a ekvalizace histogramu
11   cvtColor(faceImage, faceImageGray, CV_BGR2GRAY);
12   equalizeHist(faceImageGray, faceImageGray);
13
14   //detekce obliceju
15   face_cascade.detectMultiScale(faceImageGray, faces, 1.1, 2, 0, Size(30, 30));
16
17   //ulozeni nalezenych obliceju do vektoru
18   for (int i = 0; i < faces.size(); i++)
19   {
20     Mat temp = faceImageGray(faces[i]);
21     faceROI.push_back(temp);
22   }
23
24   //zmena rozmeru nalezenych obliceju na 100x120 pixelu
25   for (int a = 0; a < faceROI.size(); a++)
26   {
27     resize(faceROI[a],faceROI[a], Size(100, 120),0, 0, 1);
28   }
29   return faceROI;
30 }
31
```

Kód 6.5: Funkce `faceDetect()`

6.2.4 Detekce obličejů

Dalším velice zásadním blokem je blok detekce obličejů v obraze. Vstupem funkce `faceDetect()` může být buď obrázek nebo jeden snímek

(frame) z videosekvence. Další informací na vstupu je název souboru s natrénovanou kaskádou klasifikátorů.

Po načtení klasifikátorů dochází ke zpracování vstupního snímku (řádky 11 a 12). Nejprve je snímek převeden do odstínů šedi, a následně dochází k ekvalizaci jeho histogramu. Ekvalizace histogramu je operace vyrovnání histogramu na plný rozsah intervalu <0, 255>, což má za následek zvýšení jeho kontrastu. V dalším kroku je vstupní snímek odeslán do detektoru obličejů, který je realizován pomocí funkce `cv::CascadeClassifier::detectMultiscale()`. Funkce na vstupu vyžaduje jednak zpracovaný vstupní snímek (`faceImageGray`), vektor obdélníků pro uložení pozic nalezených obličejů (`vect<Rect> faces`) a hodnotu `scaleFactor`, což je hodnota, o kterou je zvětšován prohledávaný prostor obrázku. V tomto případě je to hodnota 1.1, tedy zvětšení o 10%. Další vstupní proměnnou je hodnota `minNeighbors`, která udává hodnotu, kolikrát musí být daná oblast určena jako obličej, aby mohla být za obličej skutečně považována. Hodnota flags není v novějších klasifikátorech využívána [14] a poslední hodnota udává minimální velikost nalezeného objektu, veškeré menší objekty jsou ignorovány.

Výstupem detektoru jsou souřadnice obdélníků, ve kterých byl detekován obličej. Tyto souřadnice jsou uloženy ve vektoru `faces`, vyříznuty ze vstupního snímku a následně uloženy do vektoru matic `faceROI` (řádky 18 – 22).

Jelikož jedním z hlavních požadavků pro rozpoznávání obličejů pomocí metody PCA, je zachování shodných rozměrů trénovacích i testovacích obrázků, musí být všechny detekované obličeje převedeny na rozměr 100x120 pixelů. Tento převod zajišťuje funkce `cv::resize()`.

6.2.5 Nalezení nejkratší vzdálenosti

V tomto bloku již probíhá samotné rozpoznávání, pro které jsou nutné následující vstupní hodnoty. Jednak obličeje nalezené pomocí detektoru (`faceImages`), dále projekce trénovacích obrázků (`faceProjections`), vůči kterým budou testované obrázky porovnány, a značky trénovacích obrázků (`faceLabels`), které budou přiřazovány k rozpoznaným obličejům.

Na vstupu funkce `recognize()` (viz kód 6.6) jsou obrázky opět převedeny pomocí funkce `createRowMatrix()` (viz kód 6.2) z vektoru matic

do jedné matice, v níž každý řádek reprezentuje jeden obrázek. Vstupní obrázky jsou načítány po jednom a promítány do podprostoru PCA, což zajišťuje funkce `projectFaces()` (viz kód 6.4).

`recognize()`

```
1  vector<string> recognize (const Mat& faceImages, const vector<Mat>&
2                   faceProjections, const vector<string>& faceLabels)
3  {
4    int faceNum_test = faceImages.rows; //pocet testovacich tvari
5    vector<string> recognizeLabels; //vektor pro rozpoznane znacky
6
7    for (int a = 0; a < faceNum_test; a++)
8    {
9      int threshold = 1500; //prah pro urceni identity
10
11     //projekce testovaciho obrazku do podprostoru PCA
12     Mat testFaceProjection = projectFaces(faceImages.row(a));
13
14     double leastDist = DBL_MAX;
15     string iNearest;
16
17     //zjisteni vzdalenosti k trenovacim oblicejum
18     for (int i = 0; i < faceProjections.size(); i++)
19     {
20       //vypocet aktualni vzdalenosti
21       double currentDist = norm(faceProjections[i], testFaceProjection,
22                                 NORM_INF);
23       //vyber nejkratsi vzdalenosti
24       if (currentDist < leastDist)
25       {
26         leastDist = currentDist;
27
28         if (leastDist < threshold) //uplatneni prahu
29         {
30           iNearest = faceLabels[i];
31         }
32         else
33         {
34           iNearest = "neznamy";
35         }
36       }
37     }
38     recognizeLabels.push_back(iNearest); //ulozeni rozpoznanych znacek
39   }
40   return recognizeLabels;
41 }
```

Kód 6.6: Funkce `recognize()`

Nyní je vše připraveno k procesu rozpoznávání, který probíhá na řádcích 18 – 37. V tomto `for` cyklu je počítána prostorová vzdálenost mezi projekcí aktuálně rozpoznávaného obrázku a všemi projekcemi trénovacích obrázků. Výpočet vzdáleností zajišťuje funkce `cv::norm()`, která nabízí tři druhy výpočtu vzdálenosti. `NORM_L1` využívá výpočet Hammingovy vzdálenosti

(rovnice 4.10), `NORM_L2` výpočet Euklidovské vzdálenosti (rovnice 4.9) a v aplikaci využitá `NORM_INF` pro výpočet vzdálenosti na šachovnici (rovnice 4.11).

Po výpočtu každé vzdálenosti je pomocí jednoduché podmínky zjištěno, zda je aktuální vzdálenost menší než předešlé nalezené. Pokud ano, další podmínka porovnává vzdálenost se stanovenou prahovou hodnotou, jejíž určení je popsáno v kapitole 7.2. Pokud je hodnota aktuálně nejmenší vzdálenosti nižší, než hodnota prahu, je k rozpoznávanému obrázku přiřazena značka odpovídající nalezenému trénovacímu obrázku. V opačném případě je rozpoznávanému obrázku přiřazena značka „neznámý“.

Takto jsou nalezeny značky pro všechny rozpoznávané obrázky a postupně ukládány do vektoru `recognizeLabels`.

displayFaces()

```
1  void dispalyFaces (Mat frame, const vector<string>& recognizeLabels,
2                     const string wantedPerson)
3  {
4    for (int a = 0; a < recognizeLabels.size(); a++)
5    {
6      //pokud patri oblicej hledane osobe je oznacen cervenou elipsou a jmenem
7      if (recognizeLabels[a] == wantedPerson)
8      {
9        Point center(faces[a].x + faces[a].width*0.5, faces[a].y +
10                    faces[a].height*0.5);
11       ellipse(frame, center, Size (faces[a].width*0.5, faces[a].height*0.5),
12               0, 0, 360, Scalar(0, 0, 255), 4, 8, 0);
13       putText(frame, recognizeLabels[a], Point(faces[a].x, faces[a].y +
14               faces[a].height*1.5 ), FONT_HERSHEY_COMPLEX_SMALL, 0.8,
15               Scalar(200,200,250), 1, CV_AA);
16     }
17
18     //pokud oblicej nepatri hledane osobe je oznacen modrou elispou
19     else
20     {
21       Point center(faces[a].x + faces[a].width*0.5, faces[a].y +
22                    faces[a].height*0.5);
23       ellipse(frame, center, Size (faces[a].width*0.5, faces[a].height*0.5),
24               0, 0, 360, Scalar(255, 0, 0), 4, 8, 0);
25     }
26   }
27   imshow ("video", frame); //zobrazeni snimku
28 }
```

Kód 6.8: Funkce `displayFaces()`

6.2.6 Zobrazení

Posledním krokem aplikace je zobrazení nalezených obličejů a jejich pojmenování přímo v aktuálním snímku videa. Důležitou vstupní hodnotou

funkce `displayFaces()` je jméno hledané osoby uložené v proměnné `wantedPerson`. Toto jméno je nutné aplikaci zadat před jejím spuštěním. Dalšími vstupními hodnotami jsou informace o rozpoznaných obličejích ve formě vektoru nalezených značek `recognizeLabels` a aktuální snímek `frame`.

V aktuálním snímku se mohou objevit dva druhy zobrazení obličeje, a to červená elipsa se jménem, pokud se přiřazená značka shoduje s hledaným jménem, nebo modrá elipsa bez jména v případě ostatních značek. Elipsy jsou vykreslovány pomocí funkce `cv::ellipse` a jejich poloha je určena pomocí globální proměnné `faces` získané při detekci obličejů funkcí `faceDetect()` (viz kód 6.5).

V prvním kroku (řádky 9,10 a 21,22) je určen střed vykreslované elipsy ve středu daného obdélníku z vektoru `faces`. Dále je vykreslena samotná elipsa (řádky 11,12 a 23,24) s poloměrem `faces[a].width*0.5` na ose x a `faces[a].height*0.5` na ose y. V případě, že se přiřazená značka shoduje s hledaným jménem, je obličej pojmenován funkcí `cv::putText` (řádek 13 – 15). Nakonec je aktuální snímek zobrazen v okně s názvem video (řádek 27).

`multipleOf2()`

```
1  bool multipleOf2 (int numActFrame)
2  {
3    int multiple = numActFrame;
4    while (numActFrame > 0)
5    {
6      multiple = multiple%2;
7      if (multiple == 0)
8        return true;
9      return false;
10   }
11 }
```

Kód 6.9: Funkce `multipleOf2()`

6.2.7 Optimalizace detekce

Jak bude zmíněno dále v kapitole 7.1, nejnáročnější úkon procesu rozpoznávání je detekce. Ve snaze optimalizovat běh celé aplikace je pomocí funkce `multipleOf2` posílán do detektoru pouze každý druhý snímek (viz kód

6.9). Filtrování snímků probíhá pomocí matematické operace modulo 2, tedy celočíselný zbytek po dělení dvěma. Vstupem funkce je pořadí aktuálního snímku a výstupem hodnota true nebo false, a to podle toho, zda se jedná o lichý nebo sudý snímek. Při testování aplikace však tato funkce využita nebyla, aby bylo k dispozici co nejvíce výsledků.

7 Testování aplikace

7.1 Detektor obličejů

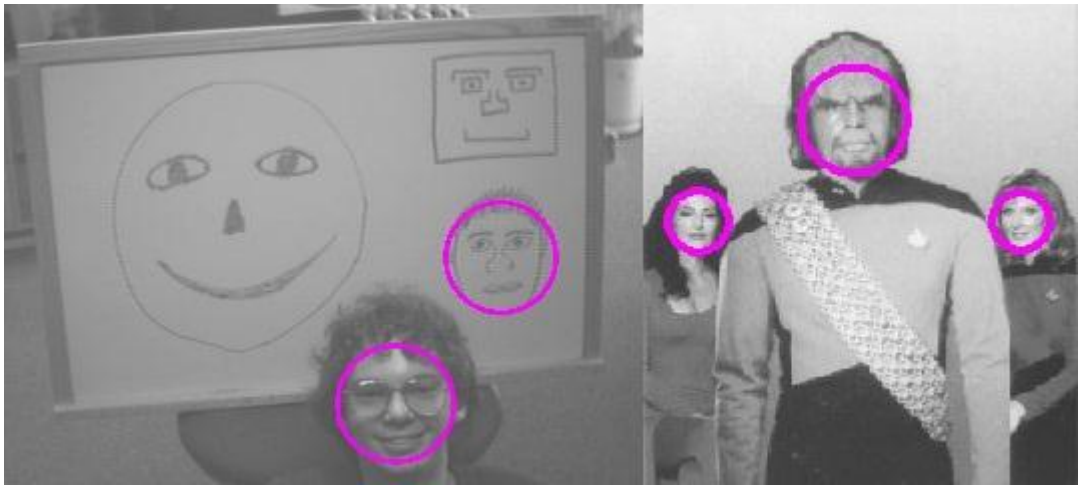
K testování detektoru byla využita testovací databáze fotografií ze stránky institutu The Vision and Autonomous Systems Center [2]. OpenCV nabízí pro detekci obličejů kaskády klasifikátorů natrénované dvěma druhy příznaků, a to jednak haarovými příznaky, nebo LBP příznaky.

Pro detekci byly využity klasifikátory natrénované pomocí LBP příznaků, i přes jejich nižší úspěšnost detekce obličejů v porovnání s klasifikátory natrénovanými pomocí Haarových příznaků. Rozhodnutí využít tyto klasifikátory ovlivnila především značně vyšší rychlost detekce (viz tabulka 7.1), což byl vzhledem ke snaze dosáhnout rozpoznávání obličejů v reálném čase rozhodující faktor. Srovnání kaskád z tabulky 7.1 probíhalo na videosekvenci dlouhé 15 sekund, podle výsledků je tedy zřejmé, že aplikace neběží zcela v reálném čase, ale při porovnání dosahovaných časů je LBP jasným vítězem.

Tab 7.1: Srovnání kaskád

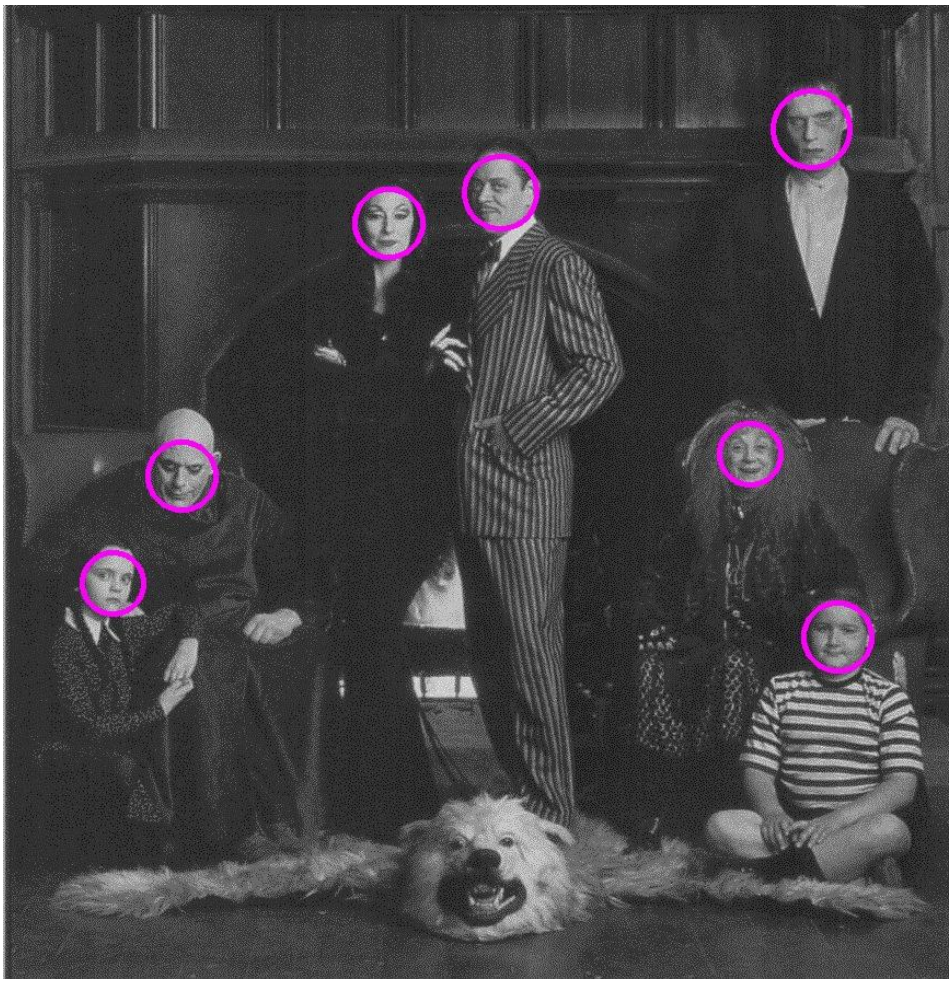
xml soubor	čas [min:s.ms]	ms/snímek	FPS	true positive	true negative
haarcascade_frontalface_alt.xml	02:07.14	282,5	3,5	368	82
haarcascade_frontalface_alt2.xml	01:50.36	245,2	4,1	374	76
lbpcascade_frontalface.xml	00:26.73	59,4	16,8	350	100

Využitý soubor s natrénovanou kaskádou klasifikátorů je určen pro detekci obličeje z přímého pohledu, s čímž nemá detektor problém ani při několikanásobné detekci (viz obr. 7.2). Úspěšná detekce obličeje proběhne například i při využití filmové masky nebo u nakresleného obličeje (viz obr. 7.1).

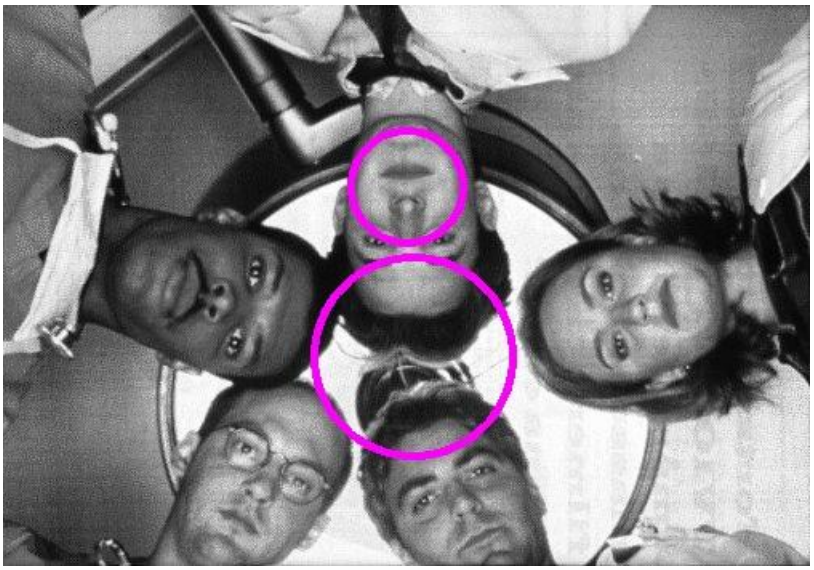


Obr 7.1: Detekce upravených obličejů

Problém však nastává při natočení obličeje. V takových případech detektor zcela selhává i při přímém pohledu na daný obličej (viz obr. 7.3).

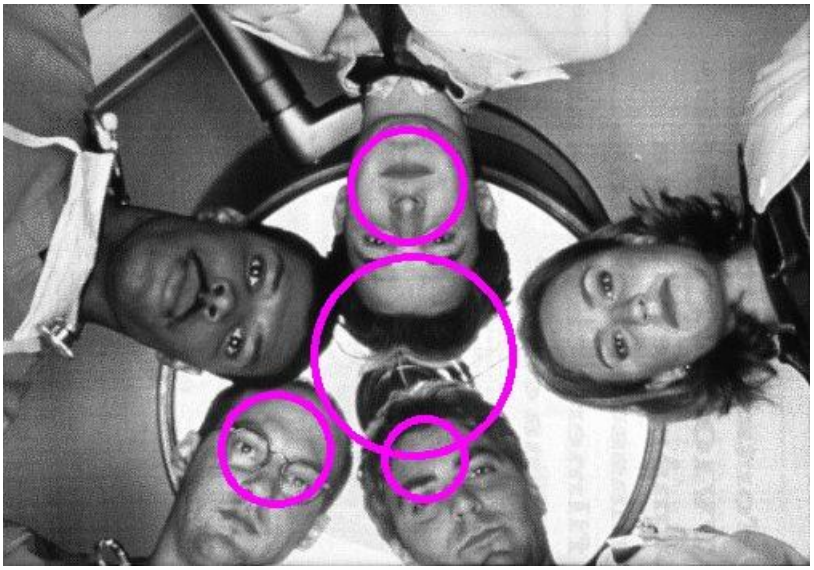


Obr 7.2: Několikanásobná detekce obličeje



Obr 7.3: Neúspěšná detekce

Veškeré znázorněné testy probíhaly se shodným nastavením detektoru, jak je uvedeno v kódu 6.5. Při snaze o lepší detekci u obr. 7.3 bylo testováno snížení hodnoty `minNeighbors` na 1 (viz obr 7.4). Detektor v tomto nastavení zdetekuje o 2 oblasti více, ale výsledek stále není nijak uspokojivý, tudíž je možné dojít k závěru, že lze úspěšně detekovat pouze obličeje bez natočení.



Obr 7.4: Neúspěšná detekce

Následně byl proveden test na první videosekvenci (viz tabulka 7.2). Tato videosekvence má celkem 450 snímků a v každém snímku je jeden obličej. Celkem proběhlo 15 měření, při kterých bylo zjišťováno, kolik nastane úspěšných detekcí (true positive), kolik chybných (false positive), jako vícenásobné detekce nebo detekce mimoobličejového prostoru, kolikrát detekce neproběhne (true negative) a jak dlouho detekce trvá. Při jednotlivých měřeních byly měněny hodnoty parametrů `scaleFactor` a `minNeighbors`.

Podle dosažených výsledků tohoto měření bylo vybráno výchozí nastavení pro proces rozpoznávání. Vzhledem k dosaženým výsledkům v počtu true positive detekcí, nulové hodnotě false positive detekcí a relativně dobrému dosaženému času bylo zvoleno měření číslo 7.

Tab 7.2: Měření na videosekvenci

#	scaleFactor	minNeigh	čas	true positive	true positive [%]	false positive	false positive [%]	true negative	true negative [%]
1	1,05	1	00:41.95	474	105,3	72	15,2	48	10,7
2	1,1	1	00:27.05	369	82,0	13	3,5	94	20,9
3	1,15	1	00:21.64	368	81,8	13	3,5	95	21,1
4	1,2	1	00:20.15	358	79,6	4	1,1	96	21,3
5	1,25	1	00:18.84	308	68,4	3	1,0	145	32,2
6	1,05	2	00:41.30	384	85,3	14	3,6	83	18,4
7	1,1	2	00:26.73	350	77,8	0	0,0	100	22,2
8	1,15	2	00:22.0	338	75,1	1	0,3	113	25,1
9	1,2	2	00:19.61	330	73,3	2	0,6	122	27,1
10	1,25	2	00:18.11	276	61,3	0	0,0	174	38,7
11	1,05	3	00:41.37	359	79,8	2	0,6	93	20,7
12	1,1	3	00:26.49	338	75,1	0	0,0	112	24,9
13	1,15	3	00:21.54	328	72,9	0	0,0	122	27,1
14	1,2	3	00:19.48	213	47,3	1	0,5	139	30,9
15	1,25	3	00:18.14	256	56,9	0	0,0	194	43,1

Obr 7.5: Pozitivní detekce

Obr 7.6: Chybná detekce

7.2 Volba prahové hodnoty při rozpoznávání obličejů

Hodnota prahu je při rozpoznávání obličejů zásadní při snaze snížit počet chybně rozpoznaných obličejů. Uplatnění tohoto prahu probíhá po zjištění nejmenší vzdálenosti promítnutého rozpoznávaného obrázku vůči promítnutým trénovacím obrázkům (viz kap. 6.2.5).

S rostoucí hodnotou nalezené minimální vzdálenosti roste také pravděpodobnost chybného rozpoznání. U metody PCA může být tento problém způsoben například rozdílnými světelnými podmínkami mezi rozpoznávaným a trénovacím obrázkem. Proto je vhodnější obrázek označit jako neznámý a předejít tak chybnému označení.

Ke zjišťování co nejlepší prahové hodnoty byla využita testovací databáze obličejů společnosti AT&T určená právě k rozpoznávání [1]. Databáze obsahuje série 10-ti fotografií od celkem 40-ti osob, tedy dohromady 400 testovacích fotografií.

Vlastní test probíhal ve třech měřeních, při kterých bylo využito prvních 20 testovacích osob. Jednotlivá měření se od sebe liší počtem trénovacích obrázků od každé osoby. V prvním měření byl pro každou osobu využit pouze jeden trénovací obrázek, ve druhém měření dva a ve třetím tři. S rostoucím počtem trénovacích obrázků se snižoval počet nově úspěšně rozpoznaných až k takové hodnotě, která se téměř rovnala počtu nově přidaných trénovacích obrázků. Proto již další měření s více trénovacími obrázky nebyly provendeny.

Výsledkem každého měření byl výpis obsahující informace o rozpoznaných osobách a nalezených vzdálenostech. V tabulce 7.3 je výpis rozdělen do skupin podle minimálních vzdáleností odstupňovaných po násobcích 100 (1. sloupec tabulky 7.3). Každé měření obsahuje v tabulce dva sloupce. V prvním sloupci je počet true positive a false positive rozpoznání a ve druhém procentuálně vyčíslená hodnota úspěšnosti true positive rozpoznání. Při volbě prahu bylo v každém měření vycházeno z hodnot procentuální úspěšnosti, konkrétně byly hledány dvě po sobě následující skupiny kdy počet false positive převýší hodnotu true positive, tedy kdy hodnota úspěšnosti rozpoznávání klesne pod 50%.

V prvním měření této podmínce odpovídají skupiny 1400, 1500, v druhém měření skupiny 1500, 1600 a v třetím měření podmínka splněna nebyla. Z těchto dvou skupin byla jako prahová hodnota vybrána ta, která

odpovídá nižší hodnotě minimální vzdálenosti. Vzhledem k tomu, že většinou je k trénování využit více než jeden obrázek, byla zvolena hodnota z druhého měření, tedy 1500.

Tab 7.3: Určování hodnoty prahu

hodnota vzdálenosti	1.měření		2.měření		3.měření	
	TP/FP	úspěšnost [%]	TP/FP	úspěšnost [%]	TP/FP	úspěšnost [%]
200	19/0	100,0	38/0	100,0	57/0	100,0
300	1/0	100,0	2/0	100,0	2/0	100,0
400	4/0	100,0	4/0	100,0	5/0	100,0
500	4/0	100,0	10/0	100,0	8/0	100,0
600	9/0	100,0	8/0	100,0	18/0	100,0
700	11/0	100,0	12/0	100,0	15/0	100,0
800	10/0	100,0	15/0	100,0	18/1	94,7
900	8/0	100,0	22/1	95,7	19/2	90,5
1000	11/0	100,0	8/2	80,0	8/3	72,7
1100	10/0	100,0	7/7	50,0	3/6	33,3
1200	8/6	57,1	10/4	71,4	5/3	62,5
1300	9/9	50,0	6/9	40,0	4/3	57,0
1400	8/14	36,4	7/6	53,8	3/1	75,0
1500	6/8	42,9	1/4	20,0	0/3	0,0
1600	2/12	14,3	2/4	33,3	1/1	50,0
1700	4/10	28,6			0/1	0,0
1800	1/0	100,0	0/1	0,0		
1900	1/3	25,0				
2000	1/1	50,0				
celkem	127/63	66,8	152/38	80,0	166/24	87,4

7.3 Testovací videosekvence

7.3.1 První videosekvence

K testování aplikace byly k dispozici celkem dvě videosekvence. První obsahuje pouze jedinou osobu a sloužila k otestování funkčnosti detektoru a k jednoduchému rozpoznávání. Při pořizování videosekvence byly shodné světelné podmínky jako při získávání trénovací sekvence obrázků. Videosekvence byla přínosná v tom, že byl znám přesný počet tváří, které měl detektor nalézt k následné identifikaci, jelikož tento počet odpovídá počtu snímků videosekvence.

Vstupní trénovací sekvence obsahuje 4 snímky z různých úhlů pohledu (viz obr. 7.7). Na obrázku 7.8 je průměrný obrázek a vlastní vektory, tzv. eigenfaces, získané ze vstupní trénovací sekvence při výpočtu podprostoru PCA.

Obr 7.7: Vstupní trénovací série videosekvence 1

Obr 7.8: Průměrný obrázek a vlastní vektory

Na obrázku 7.9 je vidět výstup z detektoru obličejů a následné pozitivní rozpoznání obličeje ve videosekvenci. Kdežto na obrázku 7.10 je obličej identifikovaný jako neznámý, jelikož vypočítaná minimální vzdálenost byla vyšší než zvolená prahová hodnota.

Obr 7.9: Detekce s úspěšným rozpoznáním

Je možné zde pozorovat, že i v případě zahrnutí bočního pohledu na obličej v trénovací sekvenci jsou hodnoty nalezených vzdáleností tak vysoké, že je obličej označen jako neznámý. Jak lze vyčíst z tabulky 7.2, toto trénovací video obsahuje celkem 450 obličejů, z nichž 350 je úspěšně zdetekovaných. Z těchto úspěšně zdetekovaných obličejů je při využití zvolené prahové hodnoty 1500 úspěšně rozpoznáno 192 obličejů a 158 označených jako neznámých.

Naprostá většina rozpoznaných obličejů je z přímého pohledu. Průměrné hodnoty vypočítaných vzdáleností a uplatněných trénovacích obrázků jsou vidět v tabulce 7.4.

Obr 7.10: Detekce s neúspěšným rozpoznáním

Tab 7.4: Vzdálenosti a trénovací obrázky

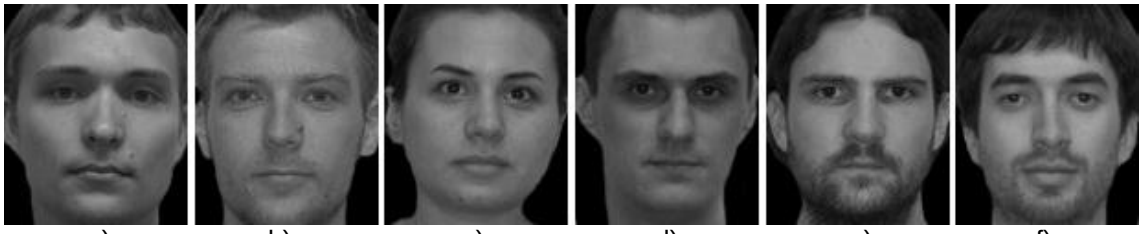
pozice ve videu	průměrná vzdálenost	většinově uplatněný trénovací obrázek
přímý pohled	800-950	přímý pohled
levý profil	2400-2800	levý profil
pravý profil	2000-2400	přímý pohled
nadhled	1200-1500	přímý pohled

Podle informací z tabulky lze vyvodit, že hlavním problémem bude sekvence trénovacích obrázků. Trénovací obrázek z přímého pohledu a z levého profilu jsou uplatněny správně, ale trénovací obrázky z pravého profilu a z nadhledu nejsou uplatněny vůbec. I přes správné uplatnění trénovacího obrázku z levého profilu, je vypočítaná hodnota minimální vzdálenosti tak vysoká, že je uplatněn práh a obličej označen jako neznámý. Řešením by tedy mohla být bohatší trénovací sekvence a případné zvýšení prahové hodnoty.

7.3.2 Druhá videosekvence

V druhé videosekvenci se vyskytuje celkem 6 osob a úkolem aplikace bylo vždy správně identifikovat jednu z nich. V první části videosekvence jednotlivé osoby přecházejí po scéně a v druhé části se všechny osoby

shluknou před kamerou k hromadnému záběru. Na obrázku 7.11 jsou vyobrazeny jednotlivé osoby a současně stejné obrázky slouží jako trénovací sekvence pro rozpoznávání. Pro identifikaci osob slouží tabulka 7.5, kde jsou k jednotlivým osobám přiřazeny odpovídající značky v aplikaci. Na obrázku 7.12 je průměrný obrázek a vlastní vektory ze vstupní trénovací sady z obr. 7.11.



a) b) c) d) e) f)

Obr 7.11: Vstupní trénovací série videosekvence 2

Tab 7.5: Jména a značky

#	značka v aplikaci
a)	osoba1
b)	osoba2
c)	osoba3
d)	osoba4
e)	osoba5
f)	osoba6

Obr 7.12: Průměrný obrázek a vlastní vektory

Druhá videosekvence je již plnohodnotnou zatěžkávací zkouškou vzniklé aplikace. Testování a pozdější statistické zpracování výsledků bylo rozděleno do dvou kroků. Nejdříve bylo provedeno měření úspěšnosti aplikace v první polovině videosekvence, kde jsou jednotlivé osoby v záběrech samotné. V druhém kroku byla analyzována druhá část videosekvence, kde jsou všechny osoby pohromadě.

První měření

Výsledky prvního měření jsou v tabulkách 7.6, 7.7 a 7.8. Dosažené výsledky jsou velice nestejnoměrné, prakticky jsou zastoupeny veškeré případy. Naprosto nulová úspěšnost v případě osoby2, kterou si aplikace nejvíce pletla s osobou4 (14x) a osobou1 (5x), kontrastovala s velice solidní úspěšností osoby1, kterou si aplikace spletla pouze dvakrát, a to s osobou4. Stoprocentní chybovost rozpoznání u osoby2 je také dána velice nízkým počtem detekcí. Naopak osoba3 zaznamenala nejvyšší počet detekcí, ale současně také nejvyšší počet špatných rozpoznání (osoba1 29x, osoba4 14x, osoba5 24x a osoba6 1x).

Tab 7.6: Úspěšnost rozpoznávání (1. část)

značka	práh	detekovaných obličejů	správně rozpoznané	neznámé	špatně rozpoznané	úspěšnost [%]	chybovost [%]
osoba1	1000	49	13	36	0	**26,5**	**0,0**
osoba2	1000	19	0	12	7	**0,0**	**36,8**
osoba3	1000	82	1	73	8	**1,2**	**9,8**
osoba4	1000	38	2	36	0	**5,3**	**0,0**
osoba5	1000	79	0	76	3	**0,0**	**3,8**
osoba6	1000	19	0	19	0	**0,0**	**0,0**
osoba1	1500	49	44	5	0	**89,8**	**0,0**
osoba2	1500	19	0	0	19	**0,0**	**100,0**
osoba3	1500	82	11	20	51	**13,4**	**62,2**
osoba4	1500	38	20	11	7	**52,6**	**18,4**
osoba5	1500	79	0	72	7	**0,0**	**8,9**
osoba6	1500	19	4	15	0	**21,1**	**0,0**
osoba1	1800	49	47	0	2	**95,9**	**4,1**
osoba2	1800	19	0	0	19	**0,0**	**100,0**
osoba3	1800	82	14	0	68	**17,1**	**82,9**
osoba4	1800	38	23	0	15	**60,5**	**39,5**
osoba5	1800	79	11	51	17	**13,9**	**21,5**
osoba6	1800	19	11	8	0	**57,9**	**0,0**
osoba1	2500	49	47	0	2	**95,9**	**4,1**
osoba2	2500	19	0	0	19	**0,0**	**100,0**
osoba3	2500	82	14	0	68	**17,1**	**82,9**
osoba4	2500	38	23	0	15	**60,5**	**39,5**
osoba5	2500	79	50	0	29	**63,3**	**36,7**
osoba6	2500	19	19	0	0	**100,0**	**0,0**

Tab 7.7: Úspěšnost prahových hodnot (1. část)

práh	úspěšnost	chybovost
1000	5,5	8,4
1500	29,5	31,6
1800	40,9	41,3
2500	56,1	43,9

Nejčastěji si aplikace spletla některou z osob s osobou1, a to celkem 55x, což může svědčit o vysoké podobnosti rysů, především pak tváře bez vousů. Dobrého výsledku aplikace dosáhla u osoby6, kterou poznala téměř v polovině případů a ani jednou ji neidentifikovala jako jinou osobu, což by mohlo být připisováno vousům. O to překvapující je ale výsledek rozpoznávání osoby5, kterou i přes nejvýraznější vousy, začala aplikace správně rozpoznávat až při prahové hodnotě 1800 a byla chybně identifikována celkem 29x (osoba1 6x, osoba4 15x a osoba6 8x).

Tab 7.8: Úspěšnost osob (1. část)

značka	úspěšnost	chybovost
osoba1	77,0	2,0
osoba2	0,0	84,2
osoba3	12,2	59,5
osoba4	44,7	24,3
osoba5	19,3	17,7
osoha6	44,7	0,0

Jako největší problém se jeví proměnlivé světelné podmínky v prostředí natáčení videosekvence a zároveň odlišné podmínky, ve kterých byla pořízena trénovací sekvence obrázků, což však bylo učiněno záměrně kvůli testování. Jak je vidět, v celé videosekvenci zprava dopadá na osoby světlo, které jim osvětluje jednu část obličeje a na druhou vrhá stín. Těmto stínům je možné připisovat značně vysoké číslo chybných rozpoznání především u osoby3, kterou celkem 24x aplikace označila za osobu5. Při úsměvu se jí vytváří v dolní části obličeje výrazný stín, který může být aplikací vyhodnocen jako vousy.

Druhé měření

Výsledky druhého měření jsou v tabulkách 7.9, 7.10 a 7.11. Měření probíhalo obdobným způsobem jako u první části, oproti které se ale výsledky poněkud liší. Jednak aplikaci působila problémy přítomnost několika osob pro identifikaci, čímž se značně zpomalil její chod, a za druhé se potvrdilo tvrzení

ohledně problematiky různorodého osvětlení scény. Oproti první části videosekvence zde byly osoby rozmístěny rovnoměrně v celém záběru. Toto rozmístění se pozitivně projevilo například u osoby2, která oproti předchozí nejhorší úspěšnosti tentokrát dosáhla úspěšnosti nejvyšší. Toto může být způsobeno skutečností, že na její pozici jsou světelné podmínky značně podobné podmínkám při pořizování trénovací sekvence, která byla pořízena v jiné části místnosti než videosekvence. Osoba3 se v druhé části nachází ve stejném prostoru jako v první části, čemuž také odpovídají přibližně stejné výsledky úspěšnosti i chybovosti.

Tab 7.9: Úspěšnost rozpoznávání (2. část)

značka	práh	detekovaných obličejů	správně rozpoznané	neznámé	špatně rozpoznané	úspěšnost [%]	chybovost [%]
osoba1	1000	168	0	168	0	**0,0**	**0,0**
osoba2	1000	91	50	40	1	**54,9**	**1,1**
osoba3	1000	138	0	125	13	**0,0**	**9,4**
osoba4	1000	199	18	133	48	**9,0**	**24,1**
osoba5	1000	187	5	181	1	**2,7**	**0,5**
osoba6	1000	143	0	142	1	**0,0**	**0,7**
osoba1	1500	168	3	165	0	**1,8**	**0,0**
osoba2	1500	91	83	1	7	**91,2**	**7,7**
osoba3	1500	138	12	80	46	**8,7**	**33,3**
osoba4	1500	199	42	0	157	**21,1**	**78,9**
osoba5	1500	187	54	46	87	**28,9**	**46,5**
osoba6	1500	143	5	81	57	**3,5**	**39,9**
osoba1	1800	168	13	155	0	**7,7**	**0,0**
osoba2	1800	91	84	0	7	**92,3**	**7,7**
osoba3	1800	138	14	12	112	**10,1**	**81,2**
osoba4	1800	199	42	0	157	**21,1**	**78,9**
osoba5	1800	187	60	12	115	**32,1**	**61,5**
osoba6	1800	143	29	19	95	**20,3**	**66,4**
osoba1	2500	168	74	1	93	**44,0**	**55,4**
osoba2	2500	91	84	0	7	**92,3**	**7,7**
osoba3	2500	138	14	0	124	**10,1**	**89,9**
osoba4	2500	199	42	0	157	**21,1**	**78,9**
osoba5	2500	187	60	0	127	**32,1**	**67,9**
osoba6	2500	143	40	0	103	**28,0**	**72,0**

Tab 7.10: Úspěšnost prahových hodnot (2. část)

práh	úspěšnost	chybovost
1000	11,1	6,0
1500	25,9	34,4
1800	30,6	49,3
2500	37,9	62,0

V případě osoby4 jsou výsledky z počátku druhé části dosti slušné, ale jakmile se ocitne v pozadí všech ostatních, rozpoznávání naprosto selhává a hodnota chybovosti začíná narůstat.

V průběhu videosekvence dochází k lehkému rozostření videa, když se fotoaparát snaží přeostřit scénu. Tato situace má také na výsledek neblahý vliv, ale současně může sloužit jako další z prvků reálné situace.

Tab 7.11: Úspěšnost osob (2. část)

značka	úspěšnost	chybovost
osoba1	13,4	13,8
osoba2	82,7	6,0
osoba3	7,2	53,4
osoba4	18,1	65,2
osoba5	23,9	44,1
osoba6	12,9	44,8

Tab 7.12: Úspěšnost rozpoznávání 2 (1. část)

značka	práh	detekovaných obličejů	správně rozpoznané	neznámé	špatně rozpoznané	úspěšnost [%]	chybovost [%]
osoba1	1000	49	27	19	3	**55,1**	**6,1**
osoba2	1000	19	14	5	0	**73,7**	**0,0**
osoba3	1000	82	36	46	0	**43,9**	**0,0**
osoba4	1000	38	31	7	0	**81,6**	**0,0**
osoba5	1000	79	76	2	1	**96,2**	**1,3**
osoba6	1000	19	12	7	0	**63,2**	**0,0**
osoba1	1500	49	31	11	7	**63,3**	**14,3**
osoba2	1500	19	15	4	0	**78,9**	**0,0**
osoba3	1500	82	52	24	6	**63,4**	**7,3**
osoba4	1500	38	34	0	4	**89,5**	**10,5**
osoba5	1500	79	76	2	1	**96,2**	**1,3**
osoba6	1500	19	13	4	2	**68,4**	**10,5**
osoba1	1800	49	32	0	17	**65,3**	**34,7**
osoba2	1800	19	15	4	0	**78,9**	**0,0**
osoba3	1800	82	62	11	9	**75,6**	**11,0**
osoba4	1800	38	34	0	4	**89,5**	**10,5**
osoba5	1800	79	78	0	1	**98,7**	**1,3**
osoba6	1800	19	16	0	3	**84,2**	**15,8**
osoba1	2500	49	32	0	17	**65,3**	**34,7**
osoba2	2500	19	15	1	3	**78,9**	**15,8**
osoba3	2500	82	67	3	12	**81,7**	**14,6**
osoba4	2500	38	34	0	4	**89,5**	**10,5**
osoba5	2500	79	78	0	1	**98,7**	**1,3**
osoba6	2500	19	16	0	3	**84,2**	**15,8**

Třetí měření

Poslední test, který byl s touto videosekvencí proveden, slouží k potvrzení domněnky o vlivu rozdílných světelných podmínek při pořizování trénovací sekvence obrázků a testovací videosekvence. Pro každou osobu bylo přímo z videa získáno 6 trénovacích obrázků, které byly následně využity při analýze první části videa. Výsledky posledního testu jsou v tabulkách 7.12, 7.13 a 7.14.

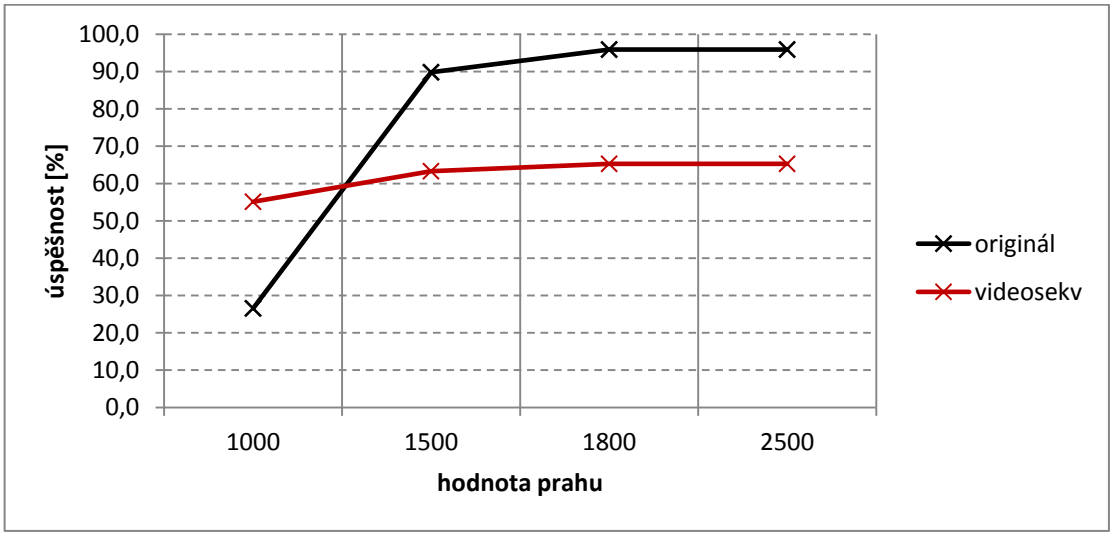
Tab 7.13: Úspěšnost prahových hodnot 2 (1. část)

práh	úspěšnost	chybovost
1000	68,9	1,2
1500	76,6	7,3
1800	82,0	12,2
2500	83,1	15,4

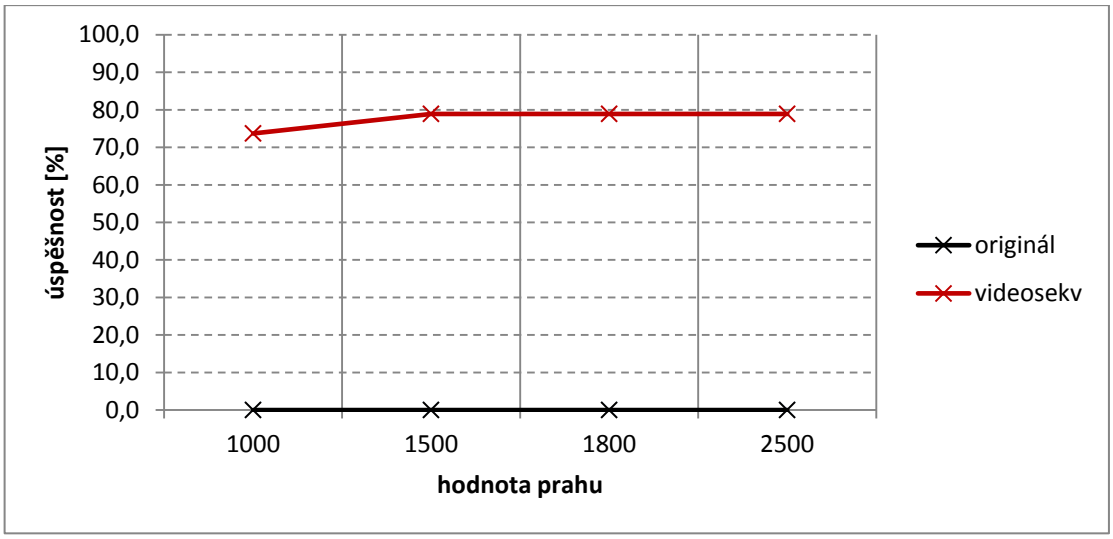
Tab 7.14: Úspěšnost osob 2 (1. část)

značka	úspěšnost	chybovost
osoba1	62,2	22,4
osoba2	77,6	3,9
osoba3	66,2	8,2
osoba4	87,5	7,9
osoba5	97,5	1,3
osoba6	75,0	10,5

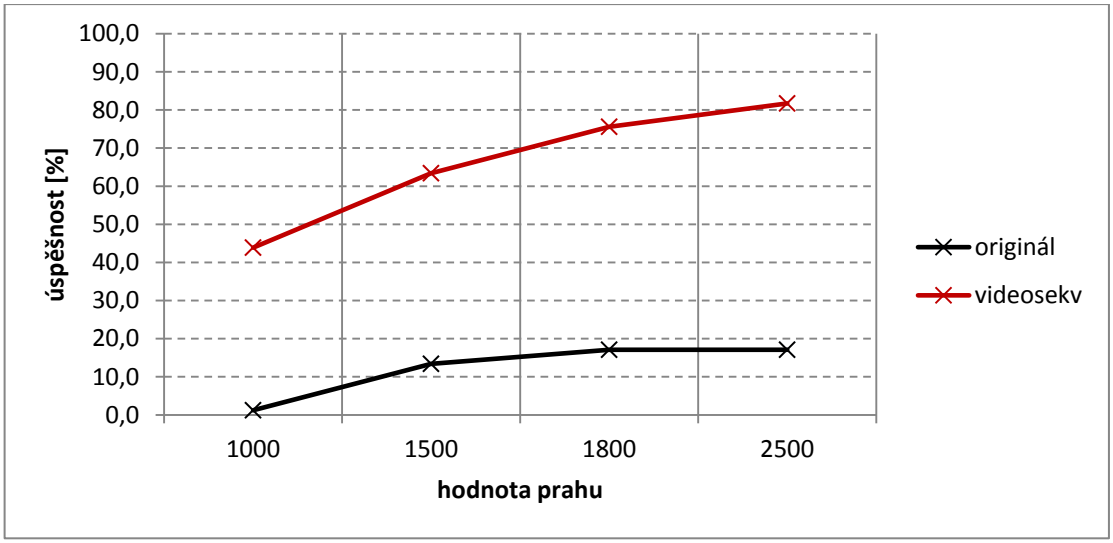
Pro lepší porovnání s výsledky prvního testu slouží následující grafy úspěšnosti rozpoznávání jednotlivých osob v závislosti na využité prahové hodnotě.



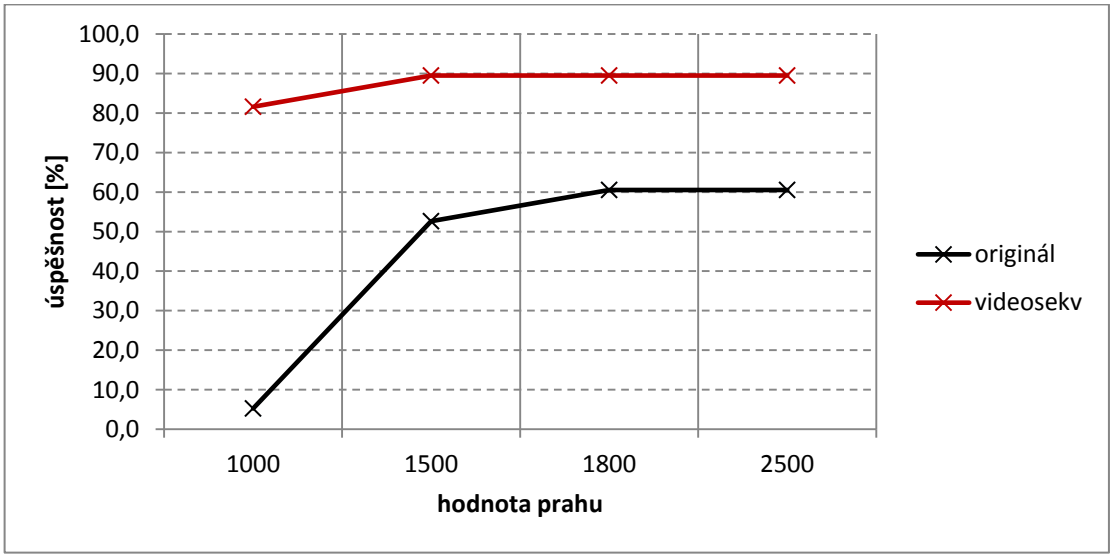
Graf 7.1: Úspěšnost rozpoznávání osoby1



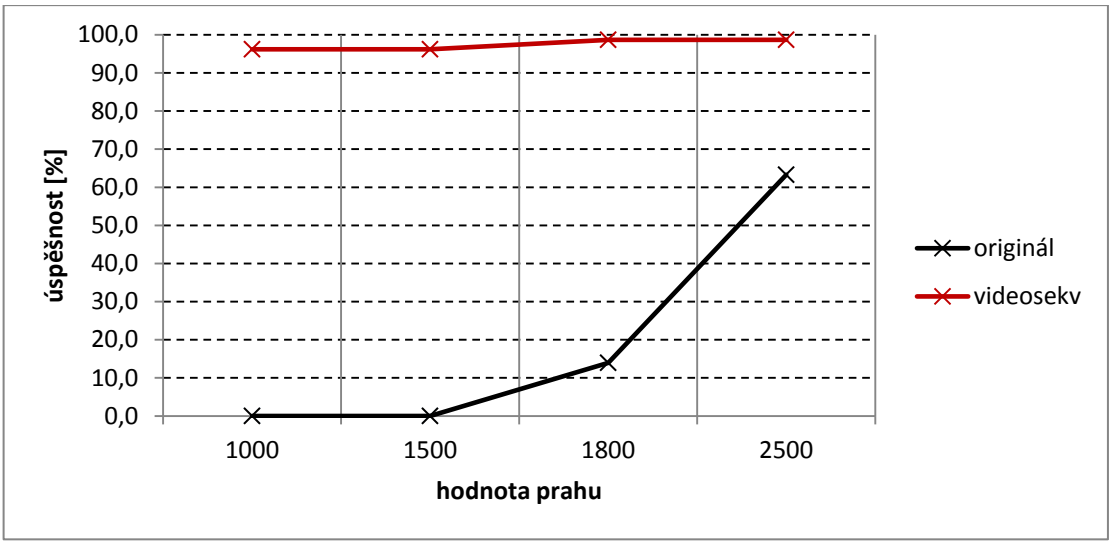
Graf 7.2: Úspěšnost rozpoznávání osoby2



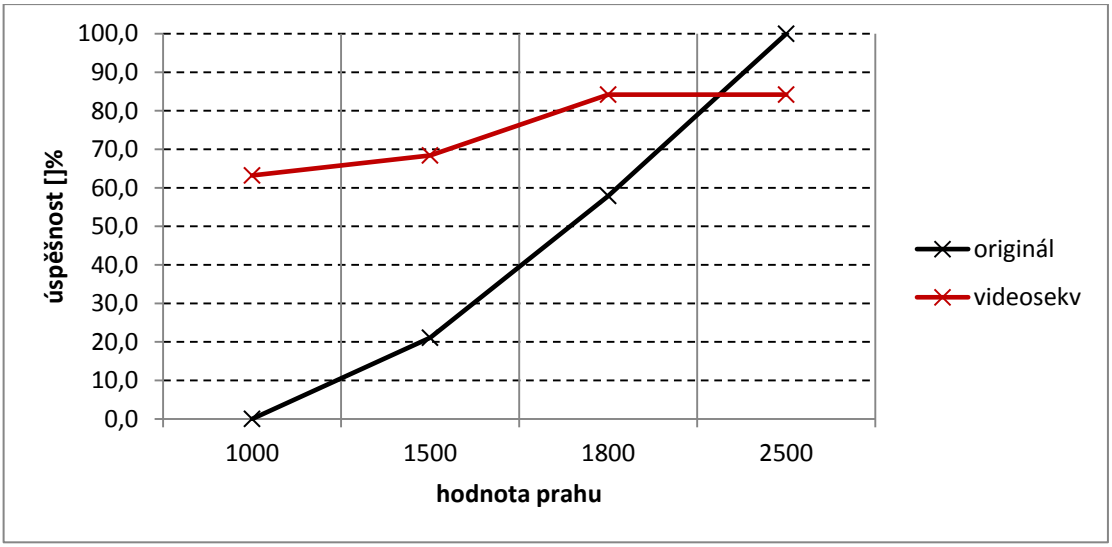
Graf 7.3: Úspěšnost rozpoznávání osoby3



Graf 7.4: Úspěšnost rozpoznávání osoby4



Graf 7.5: Úspěšnost rozpoznávání osoby5



Graf 7.6: Úspěšnost rozpoznávání osoby6

Jak je vidět z grafů, téměř u všech osob bylo rozpoznávání úspěšnější při využití trénovací série pořízené přímo z videosekvence. Různorodost průběhů je způsobena různorodostí trénovacích snímků jednotlivých osob. Snaha byla sice pořídit snímky přibližně ve stejných místech videosekvence, ale pokaždé se to nepodařilo. Jedinou výjimkou je osoba1, jejíž úspěšnost při využití původní trénovací sekvence je lepší než se sérií pořízenou přímo z videosekvence. Aplikace si při využití trénovacích obrázků z videosekvence, spletla osobu1 s osobou2 a osobou3, jejichž úspěšnosti ve třetím testu značně stoupla. Tyto tři osoby si aplikace pletla i v předchozích testech, což je způsobeno jejich vzájemnou podobností obličeji.

8 Rozpoznávání obličejů v OpenCV 2.4.0

Od verze 2.4.0 jsou v balíčku OpenCV implementovány celkem 3 metody pro rozpoznávání obličejů. Jedná se o metody PCA (viz kap. 4.1.1), LDA (viz kap. 4.1.2) a LBPH (Local Binary Patterns Histograms). V této závěrečné kapitole bude srovnána úspěšnost rozpoznávání jednotlivých metod na databázi obličejů AT&T [1].

Metody jsou spustitelné přes soubor `facerec_demo.cpp`[1] (viz kód 8.1), který je umístěný na následující cestě: `…\opencv\samples\cpp\`. Na řádku 3 je nutné zadat název *.csv souboru obsahujícímu cesty k trénovacím obrázkům

facerec_demo.cpp

```
1  int main(int argc, const char *argv[]) {
2      // path to your CSV
3      string fn_csv = string(argv[1]);
4      // images and corresponding labels
5      vector<Mat> images;
6      vector<int> labels;
7      // read in the data
8      read_csv(fn_csv, images, labels);
9      // get width and height
10     //int width = images[0].cols;
11     int height = images[0].rows;
12     // get test instances
13     Mat testSample = images[images.size() - 1];
14     int testLabel = labels[labels.size() - 1];
15     // ... and delete last element
16     images.pop_back();
17     labels.pop_back();
18     // build the PCA/LDA/LBPH model
19     Ptr<FaceRecognizer> model = createEigenFaceRecognizer();
20   //Ptr<FaceRecognizer> model = createFisherFaceRecognizer();
21   //Ptr<FaceRecognizer> model = createLBPHFaceRecognizer();
22     model->train(images, labels);
23     // test model
24     int predicted = model->predict(testSample);
25     cout << "predicted class = " << predicted << endl;
26     cout << "actual class = " << testLabel << endl;
27     }
28     waitKey(0);
29     return 0;
30 }
```

Kód 8.1: Soubor `facerec_demo.cpp`

a značky (viz obr. 6.3). Tyto načtené obrázky jsou využité jako trénovací sekvence kromě posledního, který je na řádcích 13 a 14 nastaven jako

[1] Ve verzi platné ke dni 20. 5. 2012 je v souboru chyba. Funkční soubor je dostupný na odkazu: http://bytefish.de/dev/cpp/facerec_demo.cpp

testovací a následně na řádcích 16 a 17 je odebrán z trénovací sekvence. Na 19. řádku probíhá volba využité metody rozpoznávání. `createEigenFaceRecognizer()` pro metodu PCA, `createFisherFaceRecognizer()` metodě LDA a `createLBPHFaceRecognizer()` metodě LBPH. Po výběru metody následuje proces trénování (řádek 22) a rozpoznávání (řádek 24). Na závěr je do konzole vypsána značka, kterou byl označen testovací obrázek při procesu rozpoznávání (řádek 25), a značka přidělená k obrázku v *.csv souboru (řádek 26).

Tab 8.1: Úspěšnost metody PCA

PCA						
měření	1. měření		2. měření		3. měření	
osoba	true positive [%]	false positive [%]	true positive [%]	false positive [%]	true positive [%]	false positive [%]
1	50,0	50,0	80,0	20,0	100,0	0,0
2	90,0	10,0	100,0	0,0	100,0	0,0
3	70,0	30,0	100,0	0,0	100,0	0,0
4	100,0	0,0	100,0	0,0	100,0	0,0
5	100,0	0,0	100,0	0,0	100,0	0,0
6	100,0	0,0	100,0	0,0	100,0	0,0
7	100,0	0,0	100,0	0,0	100,0	0,0
8	90,0	10,0	100,0	0,0	100,0	0,0
9	70,0	30,0	90,0	10,0	90,0	10,0
10	80,0	20,0	90,0	10,0	90,0	10,0
11	100,0	0,0	100,0	0,0	100,0	0,0
12	100,0	0,0	90,0	10,0	100,0	0,0
13	90,0	10,0	90,0	10,0	90,0	10,0
14	50,0	50,0	50,0	50,0	100,0	0,0
15	40,0	60,0	40,0	60,0	40,0	60,0
16	30,0	70,0	100,0	0,0	100,0	0,0
17	60,0	40,0	70,0	30,0	100,0	0,0
18	80,0	20,0	90,0	10,0	90,0	10,0
19	80,0	20,0	60,0	40,0	100,0	0,0
20	0,0	100,0	100,0	0,0	100,0	0,0
celkem	74,0	26,0	87,5	12,5	95,0	5,0

Při testování bylo využito celkem 20 osob z databáze, přičemž u každé testované osoby se jednalo o 10 obrázků. Každá metoda prošla celkem třemi měřeními a stejně jako při testování v kapitole 7.2 se jednotlivá měření lišila v počtu trénovacích obrázků k jednotlivým osobám. V prvním měření byl ke

každé osobě pouze jeden trénovací obrázek, ve druhém měření dva obrázky a ve třetím tři.

Tab 8.2: Úspěšnost metody LDA

LDA						
měření	1. měření		2. měření		3. měření	
osoba	true positive [%]	false positive [%]	true positive [%]	false positive [%]	true positive [%]	false positive [%]
1	50,0	50,0	70,0	30,0	100,0	0,0
2	90,0	10,0	100,0	0,0	100,0	0,0
3	80,0	20,0	100,0	0,0	90,0	10,0
4	100,0	0,0	90,0	10,0	100,0	0,0
5	100,0	0,0	100,0	0,0	100,0	0,0
6	100,0	0,0	100,0	0,0	100,0	0,0
7	90,0	10,0	100,0	0,0	100,0	0,0
8	80,0	20,0	100,0	0,0	90,0	10,0
9	50,0	50,0	80,0	20,0	100,0	0,0
10	90,0	10,0	80,0	20,0	80,0	20,0
11	100,0	0,0	100,0	0,0	100,0	0,0
12	80,0	20,0	80,0	20,0	100,0	0,0
13	100,0	0,0	100,0	0,0	100,0	0,0
14	50,0	50,0	50,0	50,0	100,0	0,0
15	30,0	70,0	50,0	50,0	70,0	30,0
16	70,0	30,0	70,0	30,0	90,0	10,0
17	50,0	50,0	70,0	30,0	100,0	0,0
18	60,0	40,0	90,0	10,0	90,0	10,0
19	70,0	30,0	60,0	40,0	100,0	0,0
20	0,0	100,0	100,0	0,0	100,0	0,0
celkem	72,0	28,0	84,5	15,5	95,5	4,5

Výsledky jednotlivých metod jsou dobré, žádná z nich výrazně nepřevyšuje ostatní. Procentuálně nejlepšího výsledku dosažených rozpoznání true positive dosáhla metoda PCA (viz tab. 8.4). Stejné měření bylo také provedeno na aplikaci vytvořené spolu s touto prací. Výsledek je uveden v nejnižším řádku tabulky 8.4. Výsledek je sice nejnižší ze všech měřených metod, ale rozdíl není nijak zásadní.

Tab 8.3: Úspěšnost metody LBPH

LBPH						
měření	1. měření		2. měření		3. měření	
osoba	true positive [%]	false positive [%]	true positive [%]	false positive [%]	true positive [%]	false positive [%]
1	40,0	60,0	60,0	40,0	100,0	0,0
2	90,0	10,0	100,0	0,0	100,0	0,0
3	30,0	70,0	90,0	10,0	90,0	10,0
4	60,0	60,0	60,0	40,0	60,0	40,0
5	70,0	30,0	100,0	0,0	100,0	0,0
6	100,0	0,0	100,0	0,0	100,0	0,0
7	100,0	0,0	100,0	0,0	100,0	0,0
8	70,0	30,0	80,0	20,0	80,0	20,0
9	40,0	60,0	80,0	20,0	90,0	10,0
10	80,0	20,0	100,0	0,0	100,0	0,0
11	100,0	0,0	100,0	0,0	100,0	0,0
12	60,0	40,0	80,0	20,0	100,0	0,0
13	100,0	0,0	100,0	0,0	100,0	0,0
14	50,0	50,0	50,0	50,0	100,0	0,0
15	70,0	30,0	60,0	40,0	50,0	50,0
16	30,0	70,0	80,0	20,0	80,0	20,0
17	40,0	60,0	30,0	70,0	80,0	20,0
18	100,0	0,0	100,0	0,0	100,0	0,0
19	70,0	30,0	60,0	40,0	100,0	0,0
20	0,0	100,0	90,0	10,0	90,0	10,0
celkem	65,0	35,0	81,0	19,0	91,0	9,0

Tab 8.4: Průměrné úspěšnosti

metoda	průměrná úspěšnost [%]
PCA	85,5
LDA	84
LBPH	79
PCA	78

Závěr

Tématem diplomové práce bylo Rozpoznávání obličejů v obraze. Hlavním úkolem bylo podrobné seznámení s biometrickým odvětvím detekce a identifikace člověka na základě obličeje. V tomto odvětví bylo vytvořeno mnoho různých metod, které byly v práci shrnuty, některé stručně, jiné podrobněji. Největší prostor byl věnován metodě PCA (Principal Component Analysis) (viz kap. 4.1.1), jejíž funkčnost byla následně vyzkoušena v realizované aplikaci pro rozpoznávání obličejů.

Pro úspěšnou realizaci rozpoznávání je důležitým předpokladem kvalitní databáze trénovacích obrázků, detektor obličejů a vybraná metoda pro samotné rozpoznávání. Pro detekci byl využit detektor Viola & Jones (viz kap. 5), který je zabudovaný v balíčku OpenCV. Detektor byl testován v kapitole 7.1. I přes nižší úspěšnost při detekci bylo rozhodnuto detektor realizovat s využitím klasifikátorů natrénovaných pomocí LBP příznaků, kvůli vyšší rychlosti detekce (viz tab. 7.1).

K rozpoznávání byla využita metoda analýzy hlavních komponent, tedy PCA. Metoda byla testována na videosekvencích v kapitole 7.3 a její úspěšnost vyšla průměrná. Úspěšnost rozpoznávání jednotlivých testovaných osob byla individuální, často docházelo k chybnému rozpoznání nebo k uplatnění prahové hodnoty. Jak bylo testem potvrzeno, metoda je velice citlivá na světelné podmínky. Největší problém způsobovaly odlišné světelné podmínky při pořizování trénovací sekvence obrázků a videosekvence.

Jako časově nejnáročnější úkon aplikace se ukázal proces detekce, proto byla snaha tento proces co nejvíce optimalizovat pro práci v reálném čase. Dobré plynulosti videosekvence bylo docíleno jednak využitím LBP příznaků a také detekcí v každém druhém snímku. Veškeré testy v kapitole 7 však probíhaly na všech snímcích videosekvence. Ve výsledku se bohužel rozpoznávání v reálném čase dosáhnout nepodařilo.

V závěrečné kapitole byly porovnány výsledky vzniklé aplikace s výsledky rozpoznávacích algoritmů zabudovaných v nejaktuálnější verzi OpenCV. Dosažené výsledky v tomto srovnávacím testu byly uspokojivé.

Problematika detekce a rozpoznávání obličejů v obraze byla prostudována teoreticky i prakticky s využitím jedné ze základních metod. Využití této metody v praxi je možné pouze u systémů, které kladou menší důraz na přesnost. Výhodou metody PCA může být pak její rychlost.

Použitá literatura:

[1] AT&T Laboratories Cambridge. CAMBRIDGE UNIVERSITY COMPUTER LABORATORY. *Face Database* [online]. [cit. 2012-05-21]. Dostupné z: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[2] CMU/VASC Image Database. WANG, Chieh-Chih. CARNEGIE MELLON UNIVERSITY. [online]. [cit. 2012-05-21]. Dostupné z: http://vasc.ri.cmu.edu/idb/html/face/frontal_images/

[3] DUBSKÝ, Ing. Milan. *Simulace biometrických zabezpečovacích systémů pracující na základě rozpoznávání tváře* [online]. VUT v Brně, 2008 [cit. 2012-05-21]. Diplomová práce. VUT v Brně. Dostupné z: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=7441.

[4] FRITSCH, Lukáš. *Metoda PCA a její implementace v jazyce C++* [online]. Praha, 7 s. Oborová práce. ČVUT v Praze. Dostupné z WWW: <http://dsp.vscht.cz/konference_matlab/MATLAB07/prispevky/fritsch_l/fritsch_l.pdf>.

[5] HEWITT, Robin. Seeing with OpenCV. [online]. 2007 [cit. 2012-05-21]. Dostupné z: http://www.cognotics.com/opencv/servo_2007_series/part_1/index.html

[6] HOUDEK, Michal. *Klasifikace podle nejbližších sousedů* [online]. Praha, 2001. 12 s. Referát. ČVUT v Praze. Dostupné z WWW: <http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_01/4/rpz4.pdf>.

[7] CHANG-YEON, Jo. Face Detection using LBP features. *Final Project Report* [online]. 2008, s. 4 [cit. 2012-05-22]. Dostupné z: http://cs229.stanford.edu/proj2008/Jo-FaceDetectionUsingLBPfeatures.pdf

[8] JONES, M. J., REHG, J.M. *Statistical Color Models with Application to Skin Detection*, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1999, Vol. 1.

[9] KIČINA, Pavol. *Automatická identifikace tváří v reálných podmínkách* [online]. [s.l.], 2011. 52 s. Diplomová práce. VUT v Brně. Dostupné z WWW: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=38755>.

[10] KRHUT, Miloš. *Rozpoznávání obličejů v obraze* [online]. [s.l.], 2009. 60 s. Diplomová práce. VUT v Brně. Dostupné z WWW: <http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=15722>.

[11] LATHA, P.; GANESAN, L.; ANNADURAI, S. *Face Recognition using Neural Networks*. Signal Processing: An International Journal [online]. 2010, 5, [cit. 2011-12-12]. Dostupný z WWW: <http://www.cscjournals.org/csc/manuscript/Journals/SPIJ/volume3/Issue5/SPIJ-37.pdf>.

[12] LI S. Z., JAIN A. K.: *Handbook of face recognition*, Springer, New York, 2005.

[13] MACHÁČ, Bc. Martin. *Toolbox pro neuronové sítě pro prostředí Mathematica* [online]. Zlín, 2009 [cit. 2012-05-21]. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Dostupné z: http://dspace.k.utb.cz/bitstream/handle/10563/10979/mach%C3%A1%C4%8D_2009_dp.pdf?sequence=1.

[14] OPENCV DEVELOPMENT TEAM. *OpenCV v2.4.0 documentation* [online]. 2012, 28. 4. 2012 [cit. 2012-05-21]. Dostupné z: <http://docs.opencv.org/index.html>

[15] Princeton University [online]. [cit. 2011-12-12]. *Face Recognition using Eigenfaces*. Dostupné z WWW: <http://www.cs.princeton.edu/~cdecoro/eigenfaces/>.

[16] PŘINOSIL, Jiří. *Biometrická identifikace*. [s.l.], 2010. 39 s. Studijní text. VUT v Brně.

[17] ROSYPAL, Stanislav. Úvod do molekulární biologie. První díl, Vstup do molekulární biologie. Molekulární biologie prokaryotické buňky. 4. inov. vyd. Brno : Rosypal, 2006. 289 s. ISBN 80-902562-5-2. [cit. 2011-12-12].

[18] ŠČUREK, Radomír. *Biometrické metody identifikace osob v bezpečnostní praxi* [online]. 2008. 58 s. Studijní text. VŠB TU Ostrava. Dostupné z WWW: <http://www.vsb.cz/export/sites-root/fbi/040/cs/sys/resource/PDF/biometricke_metody.pdf>.

[19] ŠOCHMAN Jan, Principal Component Analysis [online], 29. 12. 2006, ČVUT. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/recognition/Lab_archive/RPZ_06-07w/pca/index.html

[20] TSALAKANIDOU, Filareti; MALASSIOTIS, Sotiris; STRINTZIS, Michael G. Online Encyclopedia [online]. [cit. 2011-12-12]. *Face Recognition - Face Detection, Global Approaches for, Feature Based Techniques, Problems and Considerations, Conclusions and Future Developments Read more: Face Recognition - Face Detection, Global Approaches for, Feature Based Techniques, Problems and Considerations, Conclusions and Future Developments*. Dostupné z WWW: <http://encyclopedia.jrank.org/articles/pages/6741/Face-Recognition.html>

[21] TURK, M.; PENTLAND, A. *Eigenfaces for Face Detection/Recognition*. In Journal of Cognitive Neuroscience [online]. 1991. Dostupné z WWW: <http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf>.

[22] *Understanding biometrics* [online]. [s.l.] : Griaule Biometrics, 2008 [cit. 2011-12-12]. Dostupné z WWW: <http://www.griaulebiometrics.com/en-us/book/understanding-biometrics>.

[23] VIOLA, Paul; JONES, Michael: *Rapid Object Detection using a Boosted Cascade of Simple Features*. cvpr, ročník 1, 2001: str. 511, ISSN 1063-6919.

[24] VIOLA, Paul, JONES, Michael, *Robust Real-Time Face Detection*, International Journal of Computer Vision 57, str. 137-154, Netherlands, 2004.

[25] VLACH, Jan; PŘINOSIL, Jiří. Lokalizace obličeje v obraze s komplexním pozadím. Elektrorevue [online]. 11. 4. 2007, 2007/12, Dostupný z WWW: <http://www.elektrorevue.cz/cz/download/presne-sledovani-pohybu-tvari-v-realnemcase/>.

[26] VLACH, Jan. *Metody a aplikace detekce mrkání očí s využitím číslicového zpracování obrazu* [online]. [s.l.], 2008. 98 s. Doktorská práce. VUT v Brně. Dostupné z WWW:

<http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=10286>.

[27] WAGNER, Philipp. Face Recognition with OpenCV2. [online]. 2012, s. 26 [cit. 2012-05-21]. Dostupné z: http://www.bytefish.de/pdf/facerec_octave.pdf

[28] YANG, Ming-Hsuan; KRIEGMAN, David J.; AHUJA, Narendra *Detecting Faces in Images: A Survey*. In IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. 2002. s. 25.

Seznam použitých zkratek:

API	Application Programming Interface
BSD	Berkeley Software Distribution
FP	False Positive
FN	False Negative
GPU	Graphics Processing Unit
HLS	Barevný model (Hue, Lightness, Saturation)
HSV	Barevný model (Hue, Saturation, Value)
LBP	Local Binary Patterns
LBPH	Local Binary Patterns Histograms
LDA	Linear Discriminant Analysis
LSB	Least Significant Bit
NN	Nearest neighbor
OpenCV	Open Source Computer Vision Library
PCA	Principal Component Analysis
RGB	Barevný model (Red, Green, Blue)
SVM	Support Vector Machine
TP	True Positive
TN	True Negative
YCbCr	Barevný model
YIQ	Barevný model pro NTSC

Příloha: Obsah přiloženého CD

Zdrojové soubory aplikace pro rozpoznávání obličejů ve videosekvenci:

- trenovaci_sekvence (umístění trénovačích sekvencí)
- lbpcascade_frontalface.xml
- main.cpp
- OpenCV_Debug.props[2]
- OpenCV_Release.props
- train_original.csv (původní trénovací sekvence)
- train_videosekvence.csv (trénovací sekvence z videosekvence)
- vhauser_projekt.vcxproj
- vhauser_projekt.vcxproj.filters
- vhauser_projekt.vcxproj.user
- video.avi

Diplomová práce:

- VHauser-DP.pdf

[2] Cesty k OpenCV jsou nastaveny pro umístění C:\OpenCV\opencv a C:\OpenCV\build