VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

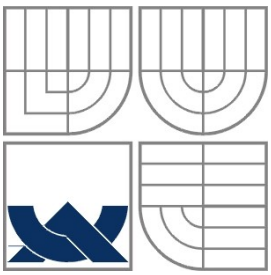
VYHLEDÁVÁNÍ OSOB VE FOTOGRAFII

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

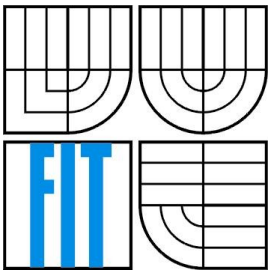
AUTOR PRÁCE
AUTHOR

Bc. PAVEL SVOBODA

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ OSOB VE FOTOGRAFII

RECOGNIZING FACES WITHIN IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Pavel Svoboda

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Miroslav Švub

BRNO 2009

Abstrakt

Základem rozpoznávání lidí ve fotografii je obecně počítačové vidění, které poskytuje metody a algoritmy pro samotnou implementaci. Některé z nich popisuje právě tato práce. Celý proces rozpoznávání lidí je zpracován do tří fází. Těmi jsou detekce, zarovnání detekovaného obličeje a konečně jeho rozpoznání. Ke každé fázi jsou zmíněny algoritmy, které se v dané problematice používají a jenž jsou ze současného pohledu stále vyvíjeny.
V implementaci tvoří páteř systému 3 základní algoritmy, mezi které patří AdaBoost pro získání klasifikátoru k detekci, metoda zarovnání obličeje na základě markantních rysů a metoda Eigenfaces k samotnému rozpoznávání. Teoreticky jsou rozebrány mimo výše uvedené i neuronové sítě pro detekci, ASM - Active Shape Models pro zarovnání a AAM - Active Appearance Model pro rozpoznávání. Závěrem nechybí tabulky dat vyhodnocující implementaci.

Abstract

The essence of face recognition within the image is generally computer vision, which provides methods and algorithms for the implementation. Some of them are described just in this work. Whole process is split in to three main phases. These are detection, aligning of detected faces and finally its recognition. Algorithms which are used to applied in given issue and which are still in progress from todays view are mentioned in every phase.
Implementation is build up on three main algorithms, AdaBoost to obtain the classifier for detection, method of aligning face by principal features and method of Eigenfaces for recognizing. There are theoretically described except already mentioned algorithms neural networks for detection, ASM – Active Shape Models algorithm for aligning and AAM – Active Appearance Model for recognition. In the end there are tables of data retrieved by implemented system, which evaluated the main implementation.

Klíčová slova

Rozpoznávání obličeje, Eigenfaces, PCA – Principal Component Analysis, AdaBoost, SVD – Singular Value Decomposition, Neuronové sítě, ASM, AAM, Gaussovo rozložení, OpenCv

Keywords

Face recognition, Eigenfaces, PCA – Principal Component Analysis, AdaBoost, SVD – Singular Value Decomposition, Neural networks, ASM, AAM, Gauss distribution, OpenCv

Citace

Pavel Svoboda: Vyhledávání osob ve fotografii, diplomová práce, Brno, FIT VUT v Brně, 2009

Vyhledávání osob ve fotografii

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Miroslava Švuba.
Další informace mi poskytli Ing. Michal Hradiš.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Svoboda
25.5. 2009

Poděkování

Za cenné rady a připomínky při vypracování této diplomové práce děkuji Ing. Miroslavu Švubovi.

Obsah

Úvod

Vyhledávání osob ve fotografii a obecně rozpoznávání objektů v jakémkoliv obrazovém médiu je z hlediska počítačového vidění komplexní systém algoritmů. Vzhledem k aktuálním přístupům lze úlohu rozdělit na tři základní části. Těmi jsou detekce, zpracování pro rozpoznání, a samotné rozpoznání objektu. Tyto části tvoří podmnožinu oboru počítačového vidění.

Cílem počítačového vidění je zautomatizovat zpracování obrazu a získat jeho interpretaci v kontextu dané úlohy. Počítačové vidění lze dělit podle využití dané znalosti na dva celky, první zpracovává data na bázi zpracování signálů, zatímco druhý celek se za pomocí metod umělé inteligence snaží data interpretovat.

Pro rozpoznání konkrétní osoby na snímku byl zvolen přirozeně nejvýraznější rys a tím je tvář. Informace v podobě postavy, postoje a všech jiných markantních rysů, podle kterých reálně osoby rozpoznáváme, nejsou využity. Od začátku byly stanoveny striktní podmínky, jenž jsou premisy pro realizovatelnost celé práce. Namátkou mezi ně patří ideální poloha obličeje, pokud možno stejné světelné podmínky, konstantní výraz tváře atd.

Detekce tváří je v současnosti algoritmicky řešitelná hned několika způsoby. V práci jsou popsány pouze Appearance based metody, které skenují obraz klasifikátorem. Nejznámější jsou různé typy Boosting metod a Neuronové sítě. Jejich popisem se zabývá kapitola Detekce obličeje, kde je důraz kladen na metodu AdaBoost, která byla v projektu vybrána pro realizaci.

Zarovnání tváří je důležité pro následné rozpoznání. Řeší se buď kombinací jednodušších metod, mezi které patří zarovnání na základě polohy markantních rysů – očí, nosu a úst, popřípadě pomocí komplexních algoritmů ASM, AAM. Implementován je způsob na základě polohy markantních rysů v kombinaci s detekcí obličeje.

Posledním krokem je samotné rozpoznání, jenž lze realizovat pomocí několika již ověřených algoritmů typu PCA, ICA, LDA, nebo naopak progresivním algoritmem AAM. Implementace zahrnuje algoritmus PCA.

Kapitola Návrh implementace rozpoznávání řeší kombinaci vybraných algoritmů a popisuje podmínky a premisy, ze kterých se v implementaci vychází. Zmíněny jsou nástroje, které jsou v implementaci použity a jsou definovány atributy, které by implementace měla umožnit uživatelsky konfigurovat.

Kapitola Implementace rozpoznávání klade důraz na kostru programu a shrnuje aplikační a technické prostředky. Zaměřuje se na zajímavé oblasti implementace a shrnuje jejich aplikaci. Součástí jsou digramy znázorňující běh aplikace a datové propojení.

V závěru jsou prezentovány výsledky na testujících datech, které tvoří tři databáze obličejů. Nechybí shrnutí vytyčených cílů, kritický závěr celé problematiky a možný směr budoucího vývoje.

1 Detekce obličeje

Obecně je detekce objektů rapidně se vyvíjející část počítačového vidění. Teorie přináší různé druhy přístupů, které lze z celkového hlediska dělit na několik základních typů.

- **Top – down**: Vyhledávání z vysoké úrovně k detailům objektu. Při zkoumání lidského vnímaní se ukázalo, že kontext pozorované scény usnadňuje určení oblastí, kterým přirozeně věnujeme pozornost z důvodů možného výskytu hledaného objektu. [1]
- **Bottom – up**: Nejdříve jsou detekovány prvky ze kterých se skládá objekt zájmu, po vyhodnocení jejich pozice je detektor schopen rozhodnout, zda se jedná či nejedná o hledaný objekt.
- **Template matching, model fitting** : Porovnání vzoru v podobě konvoluce s obrazem, místa s nejsilnější odezvou budou pravděpodobně detekované objekty podle vzoru. Popřípadě porovnáním šablony, která může být reprezentována jak tvarem – konturou, tak texturou. Mezi tyto algoritmy patří například ASM, AAM. [2][3]
- **Appearance based** : Skenování obrazu klasifikátorem.

Pro detekci byly zvoleny algoritmy typu Appearance based. Do dané kategorie spadají například algoritmy AdaBoost a Neuronové sítě, jenž ve výsledku poskytují klasifikátor, kterým je obraz skenován. Získání klasifikátoru předchází fáze trénování a validace na dostatečném množství předem připravených dat.

1.1 Klasifikace

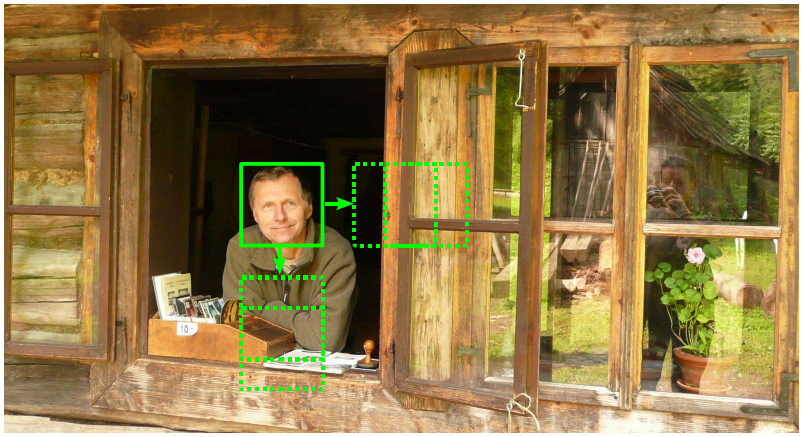
Klasifikovat objekty znamená přiřadit každý objekt jeho třídě. V případě klasifikace nad obrazovými daty (fotografie, obrázek, snímek videa) se rozlišuje příslušnost k určité třídě na základě extrahovaných příznaků. Stroj, který klasifikaci provádí se nazývá klasifikátor. Jeho vstupem je vektor příznaků X. Výstup se pak odvíjí podle rozhodovací funkce $w = h(X)$, která rozdělí obrazový prostor $IS = \{X_1, \ldots, X_n\}$ na R disjunktních podmnožin (tříd) $K_r;\ r = 1, \ldots, R$ tak, že K_r obsahuje všechny vektory příznaků X, pro které platí $w_r = h(X)$. Vektor příznaků X je obyčejně získán v rámci klasifikačního okénka, kterým je obraz skenován.

V případě detekce obličeje stačí binární klasifikátor. Ten využívá pouze dvě třídy klasifikace a to obličej (true) a pozadí (false). Takovýto klasifikátor reprezentuje například natrénovaná neuronová síť, natrénovaná kaskáda slabých klasifikátorů algoritmem AdaBoost, popřípadě Support Vector Machine. Příznaky jsou extrahovány pomocí odezvy filtrů, odezvy waveletů, intenzity pixelů, histogramu apod.

Ilustrace 1.1.1 znázorňuje klasifikátor jako okénko skenující obraz. Vektor příznaků je získán v rámci okénka a je dán na vstup natrénovanému klasifikátoru, který rozhodne, zda výřez v okénku patří do třídy obličej, nebo pozadí. Čárkovaná okénka naznačují procházení vstupního obrazu

klasifikátorem. Z toho plyne veliká variabilita pozic klasifikačního okénka. Při detekci klasifikátorem se vychází z několika základních předpokladů:

- Invariance vůči posunu – Klasifikátor musí projít celý vstupní obraz.
- Invariance vůči změně měřítka – Klasifikátor dynamicky mění svou velikost, popřípadě dochází k převzorkování vstupního obrazu.
- Invariance vůči rotaci – Vstupní obraz je postupně rotován.
- Invariance vůči úhlu pohledu, zkreslení optikou atd.



Ilustrace 1.1.1: Skenující klasifikátor

1.1.1 Trénování klasifikátoru

K natrénování binárního klasifikátoru pro detekci obličejů se obvykle volí metoda učení se s učitelem (supervised learning). K tomu je třeba mít k dispozici trénovací anotovanou sadu, kde jedna třída jsou obličeje a druhá pouze pozadí a testující sadu, na které je zjištěna úspěšnost klasifikátoru. Trénování, neboli učení, pak znamená zjištění rozhodovací funkce. Ta se podle anotovaných dat snaží generalizovat rozhodnutí, do které třídy vstupní data patří. Celý proces učení je iterativní a je zastaven až když se dosáhne požadované hodnoty prahu, který určuje maximální povolenou chybu. Využívá se takzvaně cross validation, jenž zabraňuje přetrénování. To spočívá v iterativním rozdělení trénovacích dat na S skupin. Po té se trénuje na $S-1$ částech a validuje na zbylých, nicméně cross validation je výpočetně náročnou metodou.

Vlastnosti klasifikátoru z pohledu úspěšnosti shrnuje ROC (Receiver Operating Characteristic) [4]. Klasifikátor predikuje náležitost vstupních dat k třídám $\{obličej - positive, pozadí - negative\}$ k rozlišení mezi predikovanou a skutečnou třídou se označí $\{Y, N\}$, kde Y značí shodu a N neshodu. Lze rozlišit čtyři možné typy klasifikace:

- True Positive – Správné přijetí
- True Negative – Správné odmítnutí
- False Positive – Nesprávné přijetí

– False Negative – Nesprávné odmítnutí

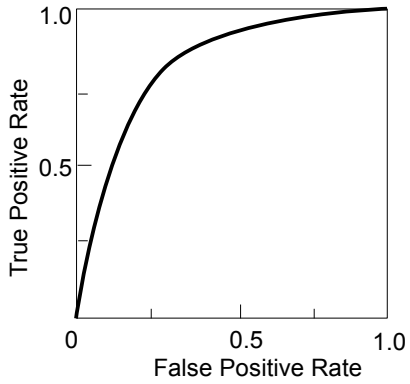
		Skutečná třída	
		positive	negative
Predikovaná třída	*Y*	True Positives	False Positives
	N	False Negatives	True Negatives
Součet hodnot ve sloupci		p	n

Tabulka 1.1: Kontingenční tabulka predikované a skutečné třídy.

Z kontingenční tabulky lze odhadnout „Míru pozitivní shody“ (TP rate - True Positive Rate) a „Míru negativní shody“ (FP rate - False Positive Rate), ze kterých se odvodí ROC křivka.

$$TP\ rate = \frac{TP}{p}$$
$$FP\ rate = \frac{FP}{n} \tag{1.1.1}$$

Ilustrace 1.1.2 znázorňuje ROC křivku, závislost TP rate na FP rate. Cílem trénování klasifikátoru je získat jeho úspěšnost danou prahem na ROC křivce.



Ilustrace 1.1.2: ROC křivka.

Chyba klasifikátoru je získána ze vztahu (1.1.2).

$$\frac{FP + FN}{TP + FP + FN + TN} \tag{1.1.2}$$

1.2 Neuronové sítě

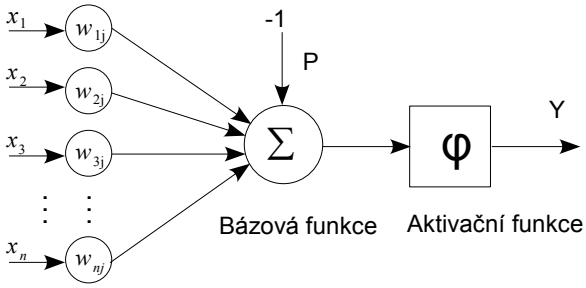
Koncept neuronových sítí pochází již z 50. let 20. století. Vycházejí z modelu biologických neuronových sítí, kde jsou neurony propojeny skrze synapse, což je spoj mezi dendritem jednoho a axonem druhého neuronu, kde dendrit je vstup a axon výstup. Synapsemi se šíří signály, které jsou ovlivněny synaptickou vazbou, ta může, ale nemusí, buď zesilovat nebo zeslabovat signál. Neuron má pak obecně *n* vstupů a 1 výstup, na který se může navázat několik dendritů. V případě, že součet

vstupních signálů dosáhne, nebo překročí určitý práh, vyšle neuron na výstup impuls – signál. K učení neuronu pak dochází ve smyslu nastavování synaptických vah, které ovlivňují signály.

Neuronová síť se v oblasti detekce obličeje používá jako klasifikátor, jehož vstupem jsou obrazová data a výstupem rozhodnutí – {obličej, pozadí}. Pro klasifikaci obrazu o velikosti 32 x 32 pixelů je třeba mít na vstupní vrstvě 32 x 32 neuronů.

1.2.1 Formální neuron

Umělá neuronová sít je paralelní systém se schopností učení se, který transformuje vstupní data na výstupní. Skládá se z neuronů a jejich propojení definuje typ topologie umělé neuronové sítě. Základním prvkem umělé neuronové sítě je formální neuron, který se skládá ze dvou funkcí, bázové - nelineární a aktivační – převádí vnitřní potenciál neuronu do definovaného oboru výstupních hodnot.



Ilustrace 1.2.1: Formální neuron, definicí se shoduje se perceptronem.

Bázovou funkci definuje 1.2.1, aktivační funkci (1.2.2)

$$g = \sum x_i w_{ij} + P \tag{1.2.1}$$

Kde P je zvolený práh, w_{ij} je váha (j – index vrstvy, i – index váhy) a x_i je vstup.

$$y = \varphi(g) \tag{1.2.2}$$

Aktivační funkce definuje výstup neuronu :

- Silně omezená aktivační funkce (Hard Limit Transfer Function)

$$y = \begin{cases} 0 & \text{když platí } \sum x_i w_{ij} + P < 0 \\ 1 & \text{když platí } \sum x_i w_{ij} + P \geq 0 \end{cases}$$

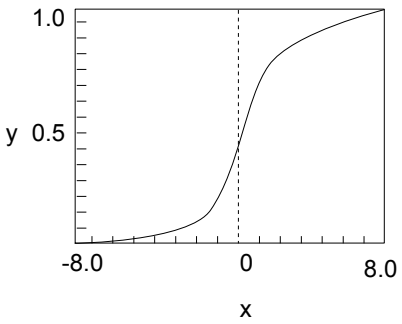
- Lineární aktivační funkce (Linear Transfer Function) $y = \left\{ \alpha * \sum x_i w_{ij} + P \right\}$

Bližší detaily viz [5][6].

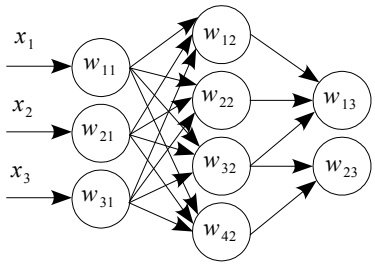
Neuronové sítě lze klasifikovat podle architektury. Jednou z hojně používaných architektur je dopředná architektura (feed forward). Jedná se o vícevrstvý perceptron, kde se základní prvek nazývá perceptron. Ten je shodný s formálním neuronem Ilustrace 1.2.1. Vícevrstvý perceptron obsahuje jako aktivační funkci svých neuronů nelineární funkci, často jde o sigmoidní funkci (1.2.3).

$$S(x) = \frac{1}{1 + e^{-\gamma x}} \tag{1.2.3}$$

kde γ udává strmost křivky v počátku soustavy.



Ilustrace 1.2.2: Sigmoida



Ilustrace 1.2.3: 3-vrstvý perceptron, dopředná neuronová síť.

Vícevrstvý perceptron se skládá z 3 a více vrstev.

- Vstupní vrstva (Input layer).
- Skrytá vrstva (Hidden layer) 1 nebo více.
- Výstupní vrstva (Output layer).

Jednovrstvá síť – jedna vrstva perceptronů je omezená pouze na lineárně separovatelná data. Typickým problémem je pro jednovrstvový perceptron logická funkce XOR, kterou nelze lineárně separovat. Vícevrstvý perceptron již je schopný separovat i lineárně neseparovatelná data.

1.2.2 Učení neuronové sítě

Neuronová síť pracuje ve dvou fázích, adaptivní (neuronová síť se učí) a aktivní (neuronová síť vykonává naučenou činnost).

Trénování neuronové sítě lze obecně rozdělit na:

- učení s učitelem – viz níže.
- učení bez učitele.

Vícevrstvý perceptron se typicky trénuje s učitelem. Trénování (učení) se provádí minimalizací výstupní chyby definovanou (1.2.4).

$$Err = T - O \tag{1.2.4}$$

Kde T je očekávaný správný výstup, zatímco O je reálný výstup neuronové sítě. Velikost chyby ovlivňuje hodnotu vah w_j vztahem (1.2.5)

$$w_j <- w_j + \alpha * x_j * Err \tag{1.2.5}$$

Konstanta α bývá malá hodnota (0,1) a nazývá se learning rate. Minimalizace se provádí například algoritmem zpětné propagace chyby (Back propagation) [6]. Při učení může dojít k

přetrénování, tomu se lze vyhnout použitím sady pro trénování a sady pro validaci dat. Trénování se zastaví ve chvíli, kdy na sadě pro validaci je dosažena minimální chyba.

Neuronové sítě se dokáží vyrovnat se ztrátou jednotlivých neuronů, jejich výhodou je schopnost vypořádat se s šumem v datech. Úspěšně se používají jako klasifikátor pro detekci obličejů [7][8], obecně je lze natrénovat na cokoliv. Další z variant neuronové sítě pro klasifikaci je síť typu neocognitron [9], jenž byla použita pro rozpoznávání rukou psaných znaků, je schopna se vyrovnat s transformacemi typu rotace a změna měřítka.

1.3 AdaBoost

Boosting je obecná metoda pro dosažení lepší přesnosti jakéhokoliv učícího se algoritmu. Jeho stěžejním předpokladem je, že nalezení lineární kombinace slabých klasifikátorů, které jako celek klasifikují jako jeden silný klasifikátor, je jednodušší, než nalezení jednoho silného klasifikátoru. Boosting se skládá ze dvou částí, první je nalezení slabých klasifikátorů, což probíhá iterativně a druhým je vytvoření silného klasifikátoru z nalezených. Silný klasifikátor by pak měl být mnohem přesnější, než kterýkoliv ze slabých klasifikátorů.

AdaBoost, představen Freund a Schapire [10] v roce 1995, je jedním z nejmarkantnějších algoritmů typu boost. Byl jednou z prvních metod pro získání silného klasifikátoru kombinací slabých. Výsledný silný klasifikátor je lineární kombinací slabých klasifikátorů (1.3.1), který v základní verzi provádí binární klasifikaci (funkce *sign*). Jeho významnou předností je, že nemá sklony se přetrénovat a exponenciálně snižuje chybu špatné klasifikace.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{1.3.1}$$

$H(x)$ - silný klasifikátor, α_t - koeficient slabého klasifikátoru, h_t - slabý klasifikátor

1.3.1 Algoritmus AdaBoost

Následující dva odstavce jsou převzaty a mírně upraveny z [11]. V každém kroku učení je do lineární kombinace přidán jeden slabý klasifikátor z množiny klasifikátorů H . Výběr v každém kroku učení je realizován hladovým způsobem tak, aby byl minimalizován horní odhad chyby klasifikátoru. Trénovací množina se skládá z dvojic (x_i, y_i) , kde x_i je vstup klasifikátoru a y_i správný výsledek klasifikace x_i do jedné ze dvou tříd $\{-1,1\}$. AdaBoost využívá při učení vážení trénovací množiny váhami D_t . Ty jsou na začátku inicializovány rovnoměrně $D_t(i)=1/m$. Vlastní algoritmus učení probíhá ve smyčce. V každém cyklu smyčky je třeba provést následující kroky:

1. nalézt nejlepší slabý klasifikátor při dané distribuci vah D_t trénovacích dat,
2. ověřit, že chyba tohoto klasifikátoru nepřekročila 0,5,
3. spočítat koeficient slabého klasifikátoru v lineární kombinaci $H(x)$,
4. aktualizovat váhy D_t .

Algoritmus AdaBoost

Vstup: $(x_1, y_1), \ldots, (x_m, y_m); x \in X, y_i \in \{-1, 1\}$
Inicializace vah : $D_1(i) = 1/m$
for $t = 1, \ldots, T$:

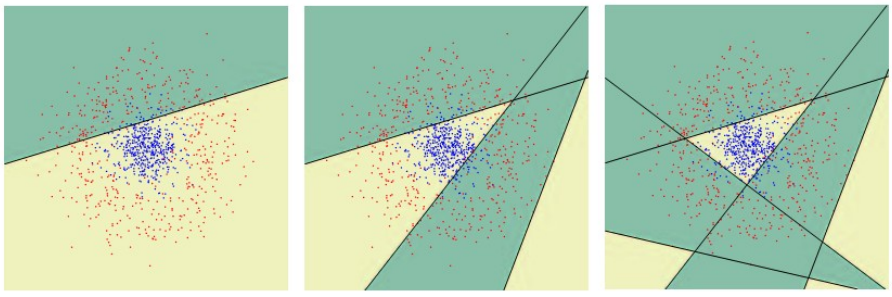
1. Find $h_t = \arg\min_{h_j \in H} \epsilon_j$; $\epsilon_j = \sum_{i=1}^{m} D_t(i) I[y_i \neq h_j(x_i)]$
2. If $\epsilon_t > \frac{1}{2}$ then stop
3. $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
4. Aktualizuj $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$; $Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$

Výsledný klasifikátor : $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

Vybraný slabý klasifikátor (krok 1) je takový, jehož vážená chyba na trénovací množině je nejmenší. Výraz $I[výrok]$ v kroku 1. vrací 1 pokud je výrok pravdivý a 0 pokud je nepravdivý. Podmínka v kroku 2. zajišťuje, aby byl nalezený slabý klasifikátor lepší než klasifikátor vracející náhodnou třídu. To je potřeba k zajištění konvergence algoritmu. Výběr příznaku podle minima vážené chyby společně s výpočtem koeficientu pro lineární kombinaci α_t (krok 3.) jsou navrženy tak, aby byl v každém kroku učení minimalizován horní odhad chyby (1.3.2) výsledného klasifikátoru [12]. Aktualizace vah v kroku 4. způsobí to, že váha špatně klasifikovávaných měření se zvětší a váha dobře klasifikovaných se zmenší. V následujícím kroku bude tedy hledán slabý klasifikátor, který bude muset lépe klasifikovat doposud špatně klasifikovaná data.

$$\prod_t \left[\sqrt{(\epsilon_t(1-\epsilon_t))} \right] = \prod_t \sqrt{(1-\gamma_t^2)} \leq \exp(-2\sum_t \gamma_t^2) \qquad (1.3.2)$$

Teoretickou vlastností AdaBoost je možnost redukovat chybu výsledného silného klasifikátoru. To dokázali Freund a Schapire [10]. γ_t určuje o kolik je lepší míra predikce klasifikátoru h_t, než náhodná klasifikace. Chybu klasifikátoru h_t tedy lze zapsat jako $\epsilon_t = 1/2 - \gamma_t$.



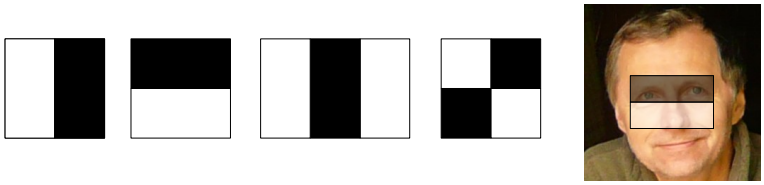
Ilustrace 1.3.1: Iterace AdaBoost, zleva t = 1, t = 3 a t = 5, v t = 5 je již vybráno 5 slabých klasifikátor. Zdroj [13].

1.3.2 Viola-Jones detektor obličeje

Devět let po prezentaci algoritmu AdaBoost představili Paul Viola a Michael J. Jones v roce 2004 real-time detektor obličejů [14], který používá modifikovaný AdaBoost k výběrů malého počtu markantních příznaků z celkově veliké množiny všech příznaků a následně také k sestavení kaskády klasifikátorů, která urychluje vyhodnocení obrazu.

1.3.2.1 Příznaky

Viola a Jones nevyhodnocují přímo hodnoty pixelů, nýbrž používají příznaky připomínající Haarovy vlnky, které jsou v základě velmi jednoduché. Důvod jejich použití je fakt, že mohou kódovat specifickou doménu[1] informací, jež lze těžko natrénovat na konečném množství trénovacích dat. Druhým důvodem použití těchto příznaků je, že jejich vyhodnocení je rychlejší, než vyhodnocení pomocí hodnot pixelů. V základní verzi Viola-Jones detektoru jsou použity tři typy těchto příznaků - Ilustrace 1.3.2.



Ilustrace 1.3.2: Haarovy příznaky, postupně zleva a, a, b, c, d – ilustrace Haarova příznaku.

1. Rozdíl mezi sumou pixelů uvnitř dvou obdélníků (a).
2. Rozdíl mezi sumou pixelů ve dvou obdélnících a jedním obdélníkem (b).
3. Rozdíl mezi sumou pixelů mezi čtyřmi obdélníky (c).

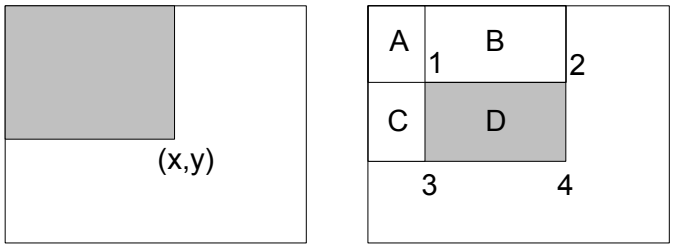
Počet takovýchto příznaků v okně 24 x 24 pixelů je velmi vysoký, přibližně 160000. Počet příznaků lze vypočítat pomocí vzorce v [15].

Pro rychlý výpočet Haarových příznaků je z obrazu spočten integrální obraz. Ten je definován jako suma hodnot pixelů obrazu v odstínech šedi tak, že na souřadnici $[x, y]$ integrálního obrazu je suma všech pixelů vlevo nad ním. Definice integrálního obrazu (1.3.3).

$$Integrální\ obraz(x, y) = \sum_{x \le x_{img},\, y \le y_{img}} obraz(x_{img}, y_{img}) \qquad (1.3.3)$$

Výpočet nad integrálním obrazem lze provést v případě Ilustrace 1.3.3 se čtyřmi přístupy do integrálního obrazu pro výpočet obdélníku D $(x_4, y_4) - (x_2, y_2) - (x_3, y_3) + (x_1, y_1)$. Integrální obraz se vypočítá jedním průchodem obrazu, v kombinaci s Haarovými příznaky pak umožňuje vypočítání příznaku pro jakoukoliv pozici a míru měřítka pomocí několika málo operací.

1 Například rozdíl sumy intenzity pixelů v oblasti očí a nosu Ilustrace 1.3.2 d.



Ilustrace 1.3.3: Integrální obraz, výpočet nad integrálním obrazem.

V pozdějších implementacích se objevují různé druhy Haarových příznaků, většina ale špatně vystihuje natočené obličeje. Nad integrálním obrazem lze efektivně počítat i Haarovy příznaky pootočené o 45°, to ale neřeší invarianci vůči rotaci.

Markantní příznaky

Adaboost je vhodná metoda pro hledání malého počtu markantních příznaků. V každém kroku vybírání příznaků přiřadí AdaBoost větší váhy markantním příznakům a naopak malé váhy špatně popisujícím příznakům. Tímto způsobem je vybráno pouze malé množství obdélníkových příznaků. Každý klasifikátor je připsán jednomu příznaku. Ten je definován (1.3.4),

$$h(x, f, p, \theta) = \begin{cases} 1 & if\ pf(x) < p\theta \\ 0 & jindy \end{cases} \qquad (1.3.4)$$

Kde je x - klasifikovaný výřez obrazu, f - příznak, θ - práh rozhodnutí (threshold), p - polarita (typ neshody).

Modifikovaný algoritmus AdaBoost pro výběr příznaků.

Vstup: $(x_1, y_1), \ldots, (x_m, y_m); x \in X, y_i \in \{0,1\}$

Inicializace vah : $w_1(i) = 1/(2\mathrm{m}), 1/(2\mathrm{l})$, kde m je počet negativních a l pozitivních vzorků.

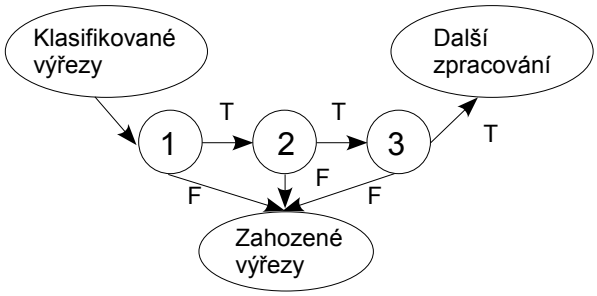
for $t = 1, \ldots, T$:

1. Normalizace vah $\dfrac{w_t(i)}{\sum_{j=1}^{n} w_t(j)} \rightarrow w_t(i)$
2. Pro každý příznak *j* trénuj klasifikátor h_j a vyhodnoť jeho chybu ϵ_j vzhledem k w_i
3. Vyber klasifikátor h_j s nejmenší chybou
4. Aktualizuj váhy vzhledem k: $w_{t+1}(i) = w_t(i)\beta_t^{1-\epsilon_i}$, kde $\epsilon_i = 0$ když x_i je klasifikováno správně, jinak je rovno 1. $\beta_t = \dfrac{\epsilon_t}{1-\epsilon_t}$

Výsledný klasifikátor : $H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & jinak \end{cases}$, kde $\alpha_t = \log(\frac{1}{\beta_t})$

První příznaky vybrané AdaBoost metodou jsou významné a dobře popisují detekovaný objekt. Podle [14] lze vytvořit silný klasifikátor pracující s pouze dvěma příznaky.

1.3.2.2 Kaskáda klasifikátorů



Ilustrace 1.3.4: Kaskáda klasifikátorů.

Po vybrání markantních příznaků, je opět použit algoritmus AdaBoost, tentokrát pro vytvoření rozhodovacího stromu, který se nazývá kaskáda. Uzly kaskády tvoří klasifikátory. Postupně jsou stupně složitější (klasifikují podle více příznaků atd.) $|F_1| < |F_2| < .. < |F_n|$. Nejprve klasifikuje výřez uzel F_1 , rozhodne-li, že ve výřezu není hledaný objekt, výřez zahodí a klasifikace výřezu je ukončena. V případě, že klasifikuje výřez jako detekovaný objekt, je předán výřez uzlu F_2 , ten, klasifikuje-li vstup jako pozadí, zahodí jej, jinak pošle kaskádou dál. Díky tomu je většina výřezů s pozadím rychle přeskočena a jen výřezy s možným výskytem detekovaného objektu jsou klasifikovány i složitějšími uzly kaskády.
Pro dosažení efektivní kaskády pro zvolenou false positive rate F ,

$$F = \prod_{i=1}^{K} f_i \tag{1.3.5}$$

Kde f_i je false positive rate i-tého klasifikátoru kaskády, detection rate[1] D ,

$$D = \prod_{i=0}^{K} d_i \tag{1.3.6}$$

kde d_i je detection rate i-tého klasifikátoru, je žádoucí minimalizovat očekávaný počet počítaných příznaků N :

$$N = n_0 + \sum_{i=1}^{K} \left(n_i \prod_{j<i} p_j \right) \tag{1.3.7}$$

Kde K je počet klasifikátorů, p_i je positive rate i-tého klasifikátoru a n_i je počet příznaků i-tého klasifikátoru.

Trénování pak hledá optimum mezi třemi kritérii, kterými jsou počet příznaků N , cílová false positive rate F a detection rate D .

1 Detection rate je ekvivalentní k True Positive Rate

Algoritmus pro sestavení kaskády

Uživatelem zadané hodnoty pro f, maximální akceptovaná false positive rate dané vrstvy kaskády a d, minimální detection rate dané vrstvy kaskády.
Uživatelem zadaná cílová false positive rat celé kaskády F_{target}.
P = množina positivních vzorků.
N = množina negativních vzorků.
$F_0 = 1.0$; $D_0 = 1.0$
$i = 0$
while $F_i > F_{target}$

- $i = i + 1$
- $n_i = 0$; $F_i = F_{i-1}$
- while $F_i > f \times F_{i-1}$
  1. $n_i = n_i + 1$
  2. Použij P a N k natrénování klasifikátoru s n_i příznaky pomocí AdaBoost.
  3. Otestuj aktuální kaskádu klasifikátorů na validační sadě pro zjištění F_i a D_i
  4. Zmenší práh i-tého klasifikátoru dokud aktuální kaskáda má detection rate minimálně $d \times D_{i-1}$
  5. $N = \emptyset$
  6. Jestliže $F_i > F_{target}$ pak otestuj aktuální kaskádu na množině obrázků s pozadím a vlož každou false detection (chybnou detekci) do množiny N.

Finální detektor byl 38 úrovňová kaskáda klasifikátorů a pracoval celkem s 6060 příznaky. První uzel kaskády klasifikoval na základě dvou příznaků a byl schopen zamítnout 50% výřezů s pozadím, zatímco s přesností blízkou 100% byl schopen klasifikovat výřez jako obličej. Testovaní bylo provedeno na MIT + CMU obličejové sadě, která obsahovala 130 obrázků s 507 označenými obličeji (přímý pohled). Při False Detection 50 obličejů měl detektor úspěšnost 91.4%.

Viola-Jones detektor není invariantní vůči rotaci, což je dáno použitím Haarových vlnek jako příznaků. Haarovy vlnky – odezvy těchto filtrů lze efektivně počítat nad integrálním obrazem i v případě, že jsou natočené o 45°.

1.4 Shrnutí kapitoly Detekce obličeje

Kapitola detekce obličeje popsala tzv. appearance based metody, které využívají klasifikátor pro skenování obrazu. Dominantními přístupy klasifikace jsou neuronové sítě a klasifikace pomocí klasifikátoru získaného AdaBoost algoritmem. Klasifikátor je třeba natrénovat na detekované objekty, to probíhá ve fázi učení, jenž v případě detekce obličeje probíhá tzv. s učitelem. Úspěšnost klasifikátoru se udává v podobě ROC křivky, která udává počet správně detekovaných objektů vůči počtu chybně detekovaných objektů.

Oblast počítačového vidění – detekce se v současné době vyvíjí velmi rychle. Výsledky jsou využívány v širokém spektru průmyslu, biometrických systémech, ale i zpracování medicínských dat. Zřejmě nelze přímo vystihnout současný trend, ale výzkum v oblasti boosting algoritmů a následně hardware akcelerace přináší kvalitní výsledky těchto metod. AdaBoost se stal základním algoritmem

trénování klasifikátorů, existuje několik modifikací, které jeho vlastnosti posouvají dál, například WaldBoost, LogitBoost, MadaBoost.

Neuronové sítě díky svým přirozeným možnostem paralelního zpracování nabízejí vzhledem k současnému trendu paralelismu slibný výzkum a uplatnění v praxi. Jejich kombinací s metodami typu boost vznikají kvalitní detektory používané v průmyslu i bezpečnostních složkách.

Vzhledem k současnému trendu boosting metod a dostupné knihovně OpenCv, byla jako detekční metoda zvolena Viola-Jones [14], která je v OpenCv implementována. Detekce při použití Viola-Jones klasifikátoru nabízí přijatelnou detection rate vůči false positive rate.

2 Zarovnání obličeje

Ve chvíli, kdy je v obraze detekován obličej, je třeba jej vůči dále popsaným metodám rozpoznávání zarovnat. Citlivými metodami pro zarovnání obličeje jsou obecně statistické přístupy ve formě PCA, ICA, nebo LDA. Pro algoritmus AAM je kritická fáze zarovnávání modelu obličeje, který má být identifikován. Výsledky statistických metod tak přímo úměrně závisí na zarovnání vstupních dat porovnávaných se známými modely.

Intuitivním způsobem zarovnání obličeje je zarovnání podle markantních rysů v obličeji jako jsou oči, nos a ústa. To znamená detekovat tyto markantní rysy a pomocí jejich pozice zarovnat obličej pro rozpoznávání. Zarovnání obličeje v obraze budou reprezentovat dvě transformace, rotace a změna měřítka (scale). Pro detekci markantních rysů je třeba použít detektory trénované na oči a nos. Ten lze získat pomocí algoritmu AdaBoost. V případě, že v detekovaném obličeji dojde k chybné detekci ve formě nekorektního počtu markantních rysů, nelogické pozici vůči ostatním rysům atd., lze použít metody pro odhad možné pozice nedetekovaného objektu, popřípadě zvolení správné detekce z několika špatných.

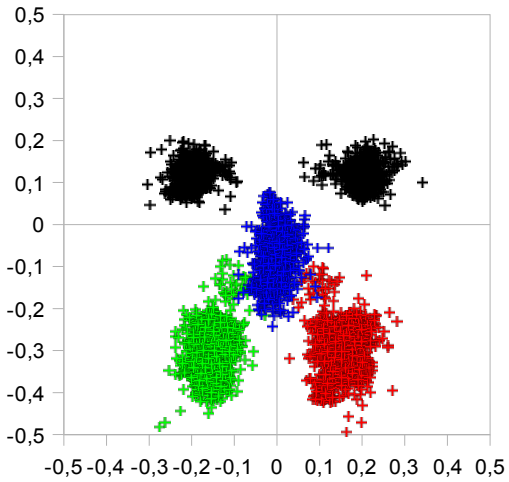
Zarovnání pomocí ASM [3] je iterativní metoda, která se snaží průměrný model obrysu obličeje pomocí regresní analýzy zarovnat na vstupní obličej. Algoritmus hledá transformaci zarovnání. Další možností je zakomponovat algoritmus AAM [2], který spojuje ASM a texturu obličeje. Ten lze přímo použít pro rozpoznávání podle obrysu tváře a samotné tváře – celkového vzhledu obličeje.

2.1 Zarovnání podle markantních rysů v obličeji

Markantní rysy v této kapitole znamenají oko, nos a koutky úst. Nos pro vyhodnocení značí střed na ose x, tj souřadnice $[0, y]$. Vzhledem k premise, že se zpracovává obličej v přímé poloze s pohledem kolmo na rovinu obrazu, lze předpokládat, že oči budou ležet na kladné ose Y. Dále je třeba definovat stavy, které mohou při detekci těchto rysů nastat :

1. Detekovány dvě oči, oba koutky úst a nos.
2. Detekován nesprávný počet očí, nebo nesprávný počet koutků úst a nos.
3. Jiná detekce.

Jiná detekce znamená jakýkoliv jiný výsledek detekce rysů, v takovém případě zarovnání podle zvolených markantních rysů nebude provedeno a je otázkou implementace, zda předpokládat, že se jedná o chybnou detekci obličeje, nebo se jen opravdu nepodařilo detekovat potřebné rysy. Ostatní dva stavy dovolují podle jednoduché heuristiky odhadnout, popřípadě vypočítat zarovnání obličeje podle potřeby.



Ilustrace 2.1.1: Graf rozložení očí, nosu, levého a pravého koutku úst vzhledem k OpenCv detektoru na datech z [16].

Pro odhad správné detekce lze použít model rozložení daného rysu vzhledem ke středu detekčního rámce obličeje Ilustrace 2.1.1. Polohu lze modelovat pomocí dvojrozměrného Gaussova rozložení (2.1.1).

$$p(x) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] = N(x;\mu,\Sigma) \qquad (2.1.1)$$

N - počet dimenzí (2), μ - vektorová střední hodnota (sloupcový vektor), Σ -kovarianční matice

Problémem přetrvává v získání parametrů těchto Gaussových rozložení. Lze definovat empirické hodnoty, které jsou normalizovány rozměrem obličeje, nebo detekčním oknem. Popřípadě se pokoušet spočítat parametry Gaussova rozložení z trénovací množiny, zde by ale zřejmě musely být rysy anotované. Před fázi rozpoznávání by pak detekované rysy měly být vyhodnoceny v rámci daného Gaussova rozložení a jako validní by byly přijaty detekce s nejvyšší mírou pravděpodobnosti. Podle validních rysů se obličej zarovná tak, aby oči ležely v horizontální rovině a pokud možno ve ve všech detekovaných obličejích ve stejné výšce.

2.2 Zarovnání pomocí ASM

ASM – Active Shape Model zahrnuje statistický model tvarů[1] objektů. Model je definován body[2], které leží na významných místech tvaru z biometrického, či geometrického pohledu a jejich blízkým okolím. Biometricky významná místa jsou koutky úst, body vymezující bradu, koutky očí atd. Model se iterativně deformuje, aby co nejlépe odpovídal cílovému obrázku. Deformace jsou omezené pouze v rozsahu statistického modelu. Je nutné vytvořit anotovaná data, ze kterých je průměrný model tvaru

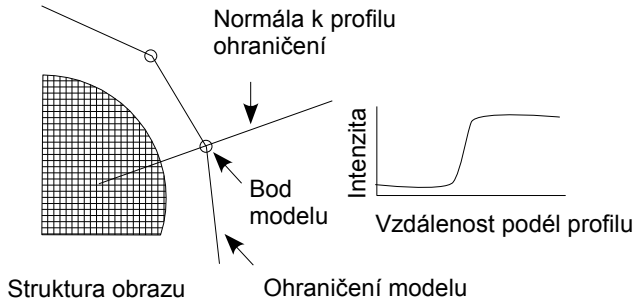
1 V anglické literatuře shape.

2 V anglické literatuře landmarks.

získán. To znamená definované významné body kolem obrysu obličeje, očí a úst. Algoritmus adaptace modelu na obličej se není schopný adaptovat na výrazné změny tvaru, což v případě obličeje se nezdá být fatálním problémem v případě, kdy obraz obličeje vyhovuje určitým podmínkám – obličej je zpříma. Na tvarově nestálé objekty existuje algoritmus aktivní kontury, který se pokouší minimalizovat součet vnitřní a vnější energie působící na tvar modelu.

2.2.1 Vytvoření průměrného modelu tvarů

Model se vytvoří vzorkováním *k* pixelů od bodů modelu Ilustrace 2.2.1 ve směru normály. Získá se tak vzorek $2k+1$ pixelů tvořících vektor g_i. Ukládají se derivace přechodu mezi intenzitami pixelů místo absolutních hodnot intenzit (potlačení globální intenzity, uchovává se pouze míra změny). Ilustrace 2.2.1 znázorňuje skokovou změnu intenzity při okraji hrany - tváře. Vzorek se normalizuje sumou absolutních intenzit použitých pixelů. Tento postup se opakuje pro každý anotovaný obrázek z trénovací množiny. Předpokladem je, že body modelu mají napříč všech anotovaných modelů Gaussovo rozložení. Toho se využívá v případě užití při rozpoznávání i podle obrysu tváře.



Ilustrace 2.2.1: Průběh intenzit ve směru normály k profilu. Zdroj [3].

2.2.2 Transformace průměrného modelu na obličej

Na začátku je dán hrubý odhad polohy obličeje, ten se může získat například díky Viola-Jones detektoru [14], případně je pomocí pozice očí a úst nastavena globální[1] transformace modelu. Kromě globální transformace se nastavuje poloha jednotlivých bodů modelu skrze parametry *b*. Celý algoritmus probíhá iterativně.

Ilustrace 2.2.2: Zarovnání průměrného modelu na obličej. Zdroj [3].

1 Globální transformací se rozumí posunutí, popřípadě i rotace modelu v obraze vzhledem k obličeji.

Algoritmus ASM – transformace modelu na obličej

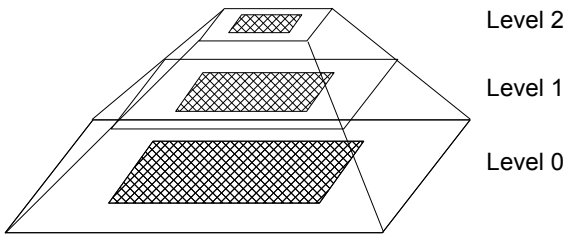
Přiloží se model na vstupní tvar - obličej.

1. Prozkoumá se okolí jednotlivých bodů modelu tvaru.
2. Aktualizují se parametry modelu (X_t, Y_t, s, θ, b) , aby model nejlépe odpovídal tvaru pod ním.
3. Výpočet se opakuje dokud model tvaru konverguje k tvaru obličeje.

X_t, Y_t - posunutí ve směru osy X, Y, s - změna měřítka (scale), θ - velikost rotace, b - parametry bodů modelu.

K nalezení transformace bodů modelu je podle [3] možno přistupovat dvěma základními způsoby. První postup využívá hodnot v okolí bodu modelu k nalezení pozice, která by mohla být hranicí mezi obličejem a pozadím – skoková změna intenzity. Druhý přístup je založený na klasifikaci do dvou tříd, kdy se vytvoří příklady příznaků hranice obličeje s pozadím. Skenování klasifikátorem se potom snaží nalézt místo, které nejvíce odpovídá korektní pozici. Finální pozice jednotlivých bodů modelu pomocí statistické analýzy lokálního okolí se hledá iterativně. Pro každý bod modelu se ve směru normály získá průběh intenzit jednotlivých pixelů Ilustrace 2.2.1. Nejvýraznější skoková změna intenzity je prohlášena za hranici obličeje.

Pro urychlení konvergence je vhodné použít prohledávání ve více úrovních rozlišení Ilustrace 2.2.3. Tímto se dá i potlačit chybná konvergence způsobená lokálními změnami intenzity. Pro každé rozlišení se vytvoří statistický model. Obraz je mírně vyhlazen a následně převzorkován do vyšší úrovně o polovičním množství pixelů. Takto vytvořenou pyramidou se pak prochází od shora dolů. Na nižší úroveň se přejde v okamžiku, kdy více než 90% nejlépe vyhovujících pixelů leží v prostředním pásmu modelu (Ilustrace 2.2.1 - pixely musí ležet v určité maximální vzdálenosti od Model boundary). Iterace probíhá dokud se nesestoupí na nejnižší úroveň, výsledkem je soubor transformačních parametrů.



Ilustrace 2.2.3: Víceúrovňový model. Zdroj [3].

Pokud je počáteční pozice modelu příliš vzdálená od hledaného tvaru, může algoritmus konvergovat špatně nebo dokonce divergovat.;

2.3 Shrnutí kapitoly Zarovnání

Metody zarovnání obecně patří do kategorii předzpracování obrazu, tomu odpovídá například zarovnání na základě markantních rysů v obličeji. Protipólem je ale metoda ASM, která je sama o

sobě komplexním algoritmem. ASM bývá aplikována na segmentaci biomedicínských dat jako crt snímků. V počátcích byla začleněna i do rozpoznávání více či méně rigidních objektů jako například obličej. V této oblasti se ale významněji uplatnila AAM metoda, která byla generalizovaným vyústěním ASM.

Komplexní algoritmy vycházející z AAM již zarovnání zahrnují jako součást svého popisu. Zarovnání a přesná lokalizace obličeje je velmi důležitá pro statistické metody z důvodu odstranění pozadí – okolí obličeje, které je z pohledu rozpoznávání obličeje šumem, nastavení pokud možno co nejpodobnější počátečních podmínek pro měření atp.

3 Rozpoznání obličeje

Lidské vnímání obličeje se na různých úrovních komplexnosti snaží implementovat algoritmy rozpoznávání obličeje. Prakticky většina současných algoritmů vychází z porovnávání statistického modelu reprezentace obličeje v rámci trénovací množiny dat. V roce 1991 byla představena jedna z prvních funkčních implementací rozpoznávání obličeje pomocí statistické analýzy trénující množiny obličejů a následného porovnávání s „neznámými“ obličeji – rozpoznávání pomocí Eigenfaces [17]. Obecně platí, že pozice a nastavení obličeje, světelné podmínky a čas jsou jedny z nejdůležitějších faktorů správného rozpoznání v2D obraze.

Základní metodou pro statistické rozpoznávání je rozpoznávání pomocí Eigenfaces. Jde o nalezení vysokých variancí (rozptylů) v rámci trénovací množiny. K tomu se používá technika PCA – Principal Component Analysis, která identifikuje informaci, kterou je obličej v rámci trénovací množiny dobře identifikovatelný. Tento postup byl v implementaci [17] aplikován na celý obraz obličeje s velkou mírou vlivu pozadí. Empirické výsledky ukázaly, že tento způsob je pro rozpoznávání efektivní. Posléze byly vyvíjeny algoritmy vycházející z metody Eigenfaces s obecně přesnějšími, popřípadě podobnými výsledky. Užití Fishers` faces namísto PCA používá LDA – Linear Discriminant Analysis, jenž dobře separuje objekty mezi příslušné třídy.

V roce 1998 byl popsán a implementován algoritmus AAM pro rozpoznávání obličejů s využitím LDA [18]. Tři roky později v roce 2001 byl implementován algoritmus AAM [2] využívající PCA k rozpoznávání biomedicínských dat. Algoritmus AAM pracuje s tvarem obličeje podobně jako ASM [3], množina tvarů obličejů se následně zpracuje pomocí PCA, čímž se získá markantní rys[1]. AAM ale zároveň přidává informaci o textuře obličeje, kterou mapuje do průměrného modelu tvaru obličeje a následně také zpracovává pomocí PCA.

Pro potlačení variace nastavení a pozice obličeje, ale také světelných podmínek, je vhodné pracovat s obličejem definovaným ve 3D prostoru, v němž je popsán tvar i textura. Vyvíjí se algoritmy, které zpětně z 2D obrazu extrahují texturu i tvar obličeje, jenž mapují na 3-D model. A dále jej zpracovávají

V kapitole rozpoznávání je popsán algoritmus s využitím Eigenfaces a AAM – Active Appearance Model.

3.1 Rozpoznávání pomocí Eigenfaces

Rozpoznávání pomocí Eigenfaces je proces pracující s informací v 2D prostoru, proto je důležité, aby obličej byl v obraze ve vzpřímené poloze. Redukce z 3D prostoru do 2D, ve kterém se rozpoznává, deklaruje menší výpočetní náročnost. Samozřejmě na úkor přesnosti.

Před uvedením metody Eigenfaces, bylo rozpoznávání aplikováno pomocí lokálních a intuitivně volených rysů jako vzdálenost očí, vzdálenost úst od špičky nosu, vzdálenost uší atp. Metoda Eigenfaces používá způsoby Teorie informace[2], která vychází z předpokladu, že významný

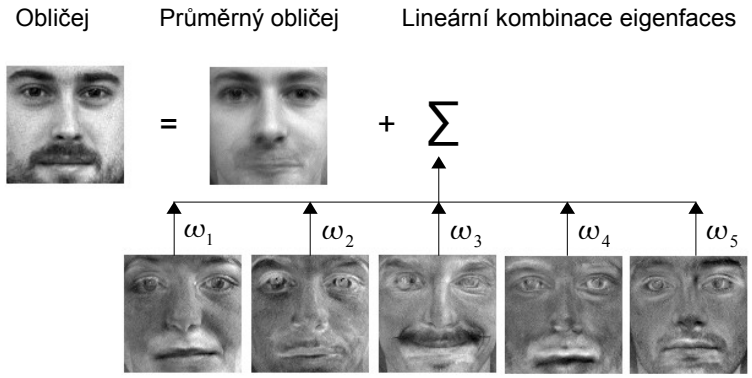
1 Nejde o viditelný rys, nýbrž o rys, který je získán statistickým zpracováním napříč trénovací množinou dat.

2 Teorie informace se zabývá kódováním, přenosem a měřením informace.

rys popisující obličej je zakódována v množině „základních obličejů“ - eigenfaces, jenž nejlépe rozlišují jednotlivé obličeje. Obličej je transformován do báze definované eigenfaces, které jsou hlavními komponentami množiny obličejů v trénovací sadě. Tím se získá vektor parametrů, na základě něhož se rozpoznává. Obličej Γ_i pak místo intenzit pixelů celého obrazu definuje lineární kombinace eigenfaces (3.1.1). Kde Φ je rozdíl obličeje a průměrného obličeje Ψ , w_j je souřadnice získaná promítnutím obličeje do báze eigenfaces dané osy – vlastní vektor v_j a K je počet vlastních vektorů.

$$\Gamma_i = \Psi + \Phi_i; \quad \Phi_i = \sum_{j=1}^{K} \omega_j v_j \tag{3.1.1}$$

Ilustrace 3.1.1 Znázorňuje rekonstrukci obličeje pomocí průměrného obličeje a lineární kombinace eigenfaces. Právě koeficienty ω_j tvoří vektor parametrů, pomocí kterého probíhá rozpoznávání.



Ilustrace 3.1.1: Obličej lze použitím PCA popsat jako průměrný obličej + lineární kombinace eigenfaces.

Obličej je reprezentován vektorem obrazových bodů - jejich intenzit (v případě odstínů šedi). Vektor je $x \times y$ dimenzionální – každá jeho položka je intenzita v daném bodě (Snímek obličeje o rozlišení 640×480 tvoří vektor, jehož dimensionalita je 307200). V trénovací množině dat jsou veškeré tyto obličeje reprezentovány svým vektorem. Z těchto vektorů se získá průměrný vektor, který se nazývá průměrný obličej (avarage face). Od vektoru v trénovací množině (obličeje) se odečte průměrný obličej. Tím je získán pro každý vektor jeho rozdíl od průměrného obličeje, z těchto rozdílů se vytvoří matice, která se vynásobí svou transponovanou podobou - výsledkem je kovarianční matice. Výpočtem vlastních vektorů kovarianční matice se získají eigenfaces[1] – vlastní vektory. Vlastní čísla vlastních vektorů popisují velikost rozptylu – čím větší, tím markantnější rozdíl. Typicky se vyberou jen markantní vlastní vektory, které poté definují prostor obličejů (facespace), $x \times y$ dimenze se redukuje na K dimenzionální prostor. Vlastní vektory jsou na sebe ortogonální, tvoří tedy novou bázi prostoru obličejů. Obličej, promítnutý do prostoru obličejů, je chápán jako bod v m dimenzionálním prostoru. Obličeje z trénovací množiny se nyní zobrazí do prostoru obličejů a získané souřadnice se použijí jako parametry pro rozpoznávání.

1 Dále je v textu použit termín vlastní vektor namísto eigenface.

V danou chvíli se rozpoznávání provádí tak, že se neznámý obličej promítne do prostoru obličejů a určí se, od kterého stejně promítnutého obličeje z trénovací množiny má minimální vzdálenost. Nastane-li situace že je objekt „příliš daleko“ od samotného prostoru obličejů, lze předpokládat, že na vstupu není obličej. Je-li relativně blízko v prostoru obličejů, ale zároveň se neblíží k žádnému z obličejů v trénovací množině, je prohlášen za neznámý.

3.1.1 PCA – Principal Component Analysis

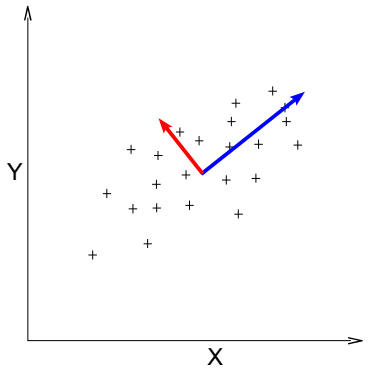
Pomocí PCA se z trénovací množiny obličejů spočítají vlastní vektory, jenž definují báze prostoru obličejů, v tom jsou vstupní data dekorelována. Do prostoru obličejů se promítne obličej a spočítá se vzdálenost, na základě které se rozhodne o jeho identitě.

Vlastní vektory se získají z kovarianční matice trénovací množiny. Vlastní vektory kovarianční matice jsou ortogonální a nejsou lineárně závislé, proto smí tvořit bázi. Ke každému vlastnímu vektoru se váže vlastní číslo (eigenvalue), které v případě PCA definuje, jak dobře vlastní vektor rozlišuje obličeje.

Je dána čtvercová matice A , téměř všechny vektory vynásobené maticí A změní svůj směr. Právě vlastní vektor $\bar{x}$ má stejný směr jako $A\bar{x}$ a zároveň platí (3.1.2), podrobnosti v [19].

$$A\bar{x} = \lambda\bar{x} \tag{3.1.2}$$

λ je skalár - vlastní číslo matice A .



Ilustrace 3.1.2: Hlavní komponenty - vlastní vektory udávají směr nejvyšší variance, hodnotu nejvyšší variance definuje vlastní číslo.

Ilustrace 3.1.3: Zleva : Obličeje trénovací sady, průměrný obličej a získané eigenfaces (vizuálně upraveny).

Následující kapitoly popisují výpočet vlastních vektorů a získání parametrů obličeje pro rozpoznávání.

3.1.1.1 Výpočet vlastních vektorů

Výpočet vlastních vektorů lze rozdělit na dvě částí, první je získání kovarianční matice ze vstupních dat a druhou výpočet vlastních vektorů získané kovarianční matice.[20]

Kovarianční matice

Kovarianční matice je definována kovariancemi mezi dimenzemi vstupních dat. Kovariance je střední hodnota součinu odchylek dvou náhodných veličin X, Y od jejich středních hodnot. Hodnota kovariance určuje míru vztahu mezi náhodnou veličinou X a Y. V případě, že kovariance je rovna nule, jsou na sobě náhodné veličiny nezávislé. Diagonálu kovarianční matice tvoří variance (rozptyl). Kovarianční matice pro 2-dimenzionální data je znázorněna v (3.1.3).

$$C = \begin{pmatrix} cov(X,X) & cov(X,Y) \\ cov(Y,X) & cov(Y,Y) \end{pmatrix}; \quad cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - EX)(y_i - EY) \tag{3.1.3}$$

Trénovací množina obličejů (obrazy s obličejem), ze které budou vlastní vektory počítány obsahuje M obličejů $I_1, \ldots, I_M$ o rozlišení $N \times N$.
Obraz obličeje je transformován do sloupcového vektoru Γ^{N^2} (3.1.4).

$$I(x,y) \rightarrow \Gamma(i) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N^2} \end{bmatrix}; \; x, y \in [1,\ldots,N], \; i \in [1,\ldots,N^2] \tag{3.1.4}$$

Z vektorů Γ_i se získá průměrný vektor Ψ Ilustrace 3.1.3.

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{3.1.5}$$

Před samotným výpočtem kovarianční matice nyní zbývá získat vektory rozdílů (3.1.6).

$$\Phi_i = \Gamma_i - \Psi \tag{3.1.6}$$

Kovarianční matice je pak rovna (3.1.7).

$$C = \frac{1}{M-1}\sum_{i=1}^{M}\Phi\Phi^T = AA^T \tag{3.1.7}$$

Kde A je matice vektorů Φ_i $A = [\Phi_1\Phi_2\ldots\Phi_M]$ o velikosti $N^2 \times M$. Výsledná kovarianční matice C má velikost $N^2 \times N^2$.

Předešlým postupem byla získána kovarianční matice trénovací množiny obličejů. Za předpokladu, že obrázky s obličeji jsou v rozlišení 256×256 jsou její rozměry 65536×65536, což je pro zpracování ohromná velikost. Při faktu, že z dané kovarianční matice C je třeba získat maximálně $M - 1$ vlastních vektorů, je zbytečné počítat s takto velkou maticí. Tuto skutečnost řeší algebraický „trik“ (3.1.11), který se standardně použije v případě, kdy dimenze vektoru z trénovací množiny mnohonásobně převyšuje počet vektorů samotných, tedy platí-li : $N^2 > M$. Místo výpočtu vlastních vektorů z kovarianční matice C se vlastní vektory získají z kovarianční matice C_s (3.1.8) a následně se dopočítají pro C (3.1.11).

$$C_s = A^T A \tag{3.1.8}$$

Matice vektorů rozdílů (Φ), A má rozměr $N^2 \times M$. Z násobení matic plyne, že rozměr výsledné matice je : $(m \times n) \times (n \times p) = m \times p$, Kovarianční matice C_s má tedy rozměr $M \times M$, neboť $A^T_{(M \times N^2)} \times A_{(N^2 \times M)} = C_{s(M \times M)}$.

Výpočet vlastních vektorů z kovarianční matice

Vlastní vektory kovarianční matice se získají pomocí SVD – Singular Value Decomposition [21], která rozloží matici $A_{m\times n}$ na součin matic $U\,\Sigma\,V^T$.

$$A = U\,\Sigma\,V^T,\ \Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \tag{3.1.9}$$

Pro unitární matice $U_{m\times m}$, $V_{n\times n}$ platí $U^T U = I$ a $V^T V = I$ (inverzní matice je totožná s transponovanou). Matice Σ_r je diagonální a pro prvky na její diagonále platí $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$, σ_i jsou nenulová singulární čísla matice A. Unitární matice jsou po sloupcích $U = [u_1, \ldots, u_m]$, $V = [v_1, \ldots, v_n]$ tak, že u_i a v_i jsou levé a pravé singulární vektory odpovídající singulárním číslům σ_i.

Pro kovarianční matice C a C_s, které jsou definovány jako AA^T a $A^T A$ platí (3.1.10).

$$\begin{aligned} C &= AA^T = (U\,\Sigma\,V^T)^T(U\,\Sigma\,V^T) = V\,\Sigma^2 V^T \\ C_S &= A^T A = (U\,\Sigma\,V^T)(U\,\Sigma\,V^T)^T = U\,\Sigma^2 U^T \end{aligned} \tag{3.1.10}$$

Σ^2 je matice vlastních čísel matice $C = AA^T$ i $C_s = A^T A$. Vlastní čísla leží na její diagonále, kde poslední hodnota na diagonále je 0. Vektory v_i jsou vlastní vektory matice $C = AA^T$, zatímco vektory u_i jsou vlastními vektory matice $C_s = A^T A$.

Se znalostí SVD a výpočtu kovarianční matice, lze postup získání vlastních vektorů popsat v několika bodech. Mějme tedy kovarianční matici $C_s = A^T A$ a $C = AA^T$. Z definice vlastních vektorů a vlastních čísel (3.1.2) lze spočítat.

$$\begin{aligned} &SVD(C_s) = U\,\Sigma^2 U^T \\ &A^T A u_i = \sigma u_i \\ &\textit{Násobeno zleva } A \\ &AA^T A u_i = A\sigma u_i \\ &C A u_i = \sigma(A u_i);\ \ C = AA^T \\ &C v_i = \sigma v_i;\ \ v_i = A u_i \\ &A u_i = v_i \end{aligned} \tag{3.1.11}$$

Stačí zvolit pouze K významných vlastních vektorů, ty lze zvolit podle odpovídajících vlastních čísel. Platí, že vlastní číslo blížící se nule značí malou varianci a tudíž lze příslušný vlastní vektor zanedbat. Na závěr výpočtu vlastních vektorů je třeba je normalizovat $\|v_i\| = 1$.

3.1.1.2 Výpočet parametrů pro rozpoznávání

Nyní lze každý obličej z trénovací množiny, odečtený od průměrného obličeje, Φ reprezentovat lineární kombinací vlastních vektorů Ilustrace 3.1.1 a vzorec (3.1.1). Parametry se získají promítnutím Φ do báze vlastních vektorů v_i.

$$\omega_j = v_j^T \Phi_i \tag{3.1.12}$$

Takto se získá K parametrů ω_i, kde K je počet použitých vlastních vektorů.

3.1.1.3 Výsledná aplikace PCA

Ve chvíli, kdy je zřejmé, jak a z čeho získat vlastní vektory, jenž definují bázi prostoru, ve kterém se provádí rozpoznávání, lze PCA metodu algoritmicky shrnout.

PCA – promítnutí obličeje do báze vlastních vektorů

1. Transformace obrazu $I_i^{(N,N)}$ do sloupcového vektoru $\Gamma_i^{N^2}$.
2. Výpočet průměrného obličeje $\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$.
3. Získání vektorů $\Phi_i = \Gamma_i - \Psi$, dále normalizovaný obličej.
4. Vytvoření kovarianční matice $C_s = A^T A = [\Phi_1 ... \Phi_n]^T \times [\Phi_1 ... \Phi_n]$.
5. Výpočet vlastních vektorů v_i : $SVD(C_s) = U \Sigma U^T$ kde $U = [u_i ... u_{M-1}]$; $v_i = A u_i$. Při použití SVD jsou výsledné vlastní vektory již uspořádány podle hodnoty vlastního čísla.
6. Normalizace vlastních vektorů v_i : $v_i = \frac{v_i}{|v_i|}$.
7. Promítnutí normalizovaných tváří do báze tvořené vlastními vektory – získání parametrů Ω_i : $V^T \Phi_i = \Omega_i$, kde $V^T = [v_1 ... v_{M-1}]^T$ a $\Omega_i = [\omega_1 ... \omega_{M-1}]$.

Při získání vlastních vektorů lze provést komprimaci na základě potlačení určité variance v datech, většinou se potlačí variance, které jsou nevýrazné v kontextu celé množiny. Vlastní číslo udává míru variance, stačí z transformační matice V vyjmout vlastní vektor související s malým vlastním číslem.

PCA – rekonstrukce původního obličeje z báze vlastních vektorů

1. Získání normalizovaného obličeje Φ : $\Phi_i = V \Omega_i$
2. Přičtení k průměrnému obličeji Ψ : $\Gamma_i = \Psi + \Phi_i$
3. Transformace zpět do 2D obrazu $I_i^{N,N}$ z $\Gamma_i^{N^2}$

Krok 1. v rekonstrukci původního obličeje plyne z faktu, že matice vlastních vektorů V je ortogonální a také unitární, tedy platí $V^{-1} = V^T$.

3.1.2 Proces rozpoznávání

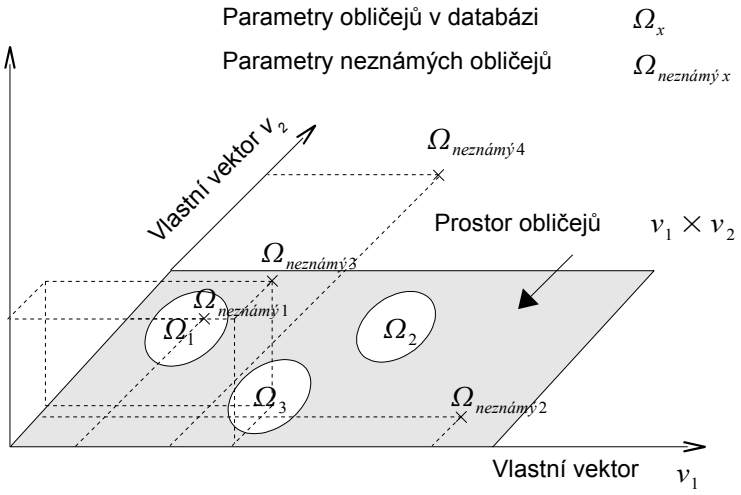
Pomocí výše popsané metody PCA byl získán vektor parametrů Ω_i pro každý obličej I_i v trénovací množině. Neznámý obličej, který se rozpoznává se zpracuje stejně jako obličeje v trénovací množině, a promítne se do prostoru vlastních vektorů trénovacích obličejů. Získá se tak jeho vektor parametrů $\Omega_{neznámý}$. Rozpoznání spočívá v nalezení nejmenší vzdálenosti mezi $\Omega_{neznámý}$ a Ω_i .

$$min\|\Omega_{neznámý} - \Omega_i\| \tag{3.1.13}$$

Algoritmus rozpoznání

1. Transformace obrazu $I^{(N,N)}_{neznámý}$ do sloupcového vektoru $\Gamma^{N^2}_{neznámý}$.
2. Získání vektorů $\Phi_{neznámý} = \Gamma_{neznámý} - \Psi$.
3. Promítnutí normalizovaných tváří do báze tvořené vlastními vektory – získání parametrů Ω_i : $V^T \Phi_{neznámý} = \Omega_{neznámý}$, kde $V^T = [v_1 ... v_{M-1}]^T$ a $\Omega_{neznámý} = [\omega_1 ... \omega_{M-1}]$.
4. Zjištění minimální vzdálenosti od Ω_i : $min \|\Omega_{neznámý} - \Omega_i\|$

Je třeba zvolit práh vzdálenosti, pod který ještě bude platit, že pravděpodobnost, že neznámý obličej byl rozpoznán je dostatečně vysoká. Jestliže vzdálenost překročí zvolená práh, je obličej prohlášen za neznámý. Lze zvolit ještě druhý práh, který definuje, zda leží vstupní obraz v prostorů obličejů. V případě, že by vzdálenost překročila i druhý práh, lze s určitou pravděpodobností konstatovat, že vstupní obraz zřejmě neobsahuje obličej. Mohou nastat 4 případy rozpoznání.



Ilustrace 3.1.4: Prostor obličejů a možné případy rozpoznání.

1. $\Omega_{neznámý\,1}$ leží v prostoru obličejů a zároveň blízko známému obličeji – obličej je identifikován.
2. $\Omega_{neznámý\,2}$ leží v prostoru obličejů, ale není blízko žádného ze známých obličejů – identifikován jako neznámý obličej.
3. $\Omega_{neznámý\,3}$ neleží v prostoru obličejů, ale leží blízko známému obličeji – $\Omega_{neznámý\,3}$ je podobný známému obličeji, ale zřejmě se vůbec nejedná o obličej.
4. $\Omega_{neznámý\,4}$ Neleží v prostou obličejů a není blízko žádnému známému obličeji - $\Omega_{neznámý\,4}$ není obličej.

Pro samotné měření vzdálenosti mezi vektory parametrů dvou obličejů se typicky užívají dvě metody : Euklidovská vzdálenosti a Mahalanobisova vzdálenost.

Euklidovská vzdálenost

Jde o typickou vzdálenost definovanou vzorcem (3.1.14). Její slabou stránkou oproti Mahalanobisově vzdálenosti je závislost na měřítku obou porovnávaných vektorů.

$$\|X - Y\| = \sqrt{(x_i - y_i)^2} \tag{3.1.14}$$

Mahalanobisova vzdálenost

Mahalanobisova vzdálenost určuje, jak moc si jsou srovnávána data podobná. Je taktéž nazývána kvadratickou vzdáleností. Měří rozdílnost dvou tříd objektů.

$$d(X, Y)_M = \sqrt{(X - Y)^T C^{-1} (X - Y)} \tag{3.1.15}$$

C je kovarianční matice (3.1.3) měřených dat. Vstupní vektory musí mít stejnou dimenzi, což v případě parametrů získaných pomocí PCA platí.

Jestliže je kovarianční matice diagonální, pak se Mahalanobisova vzdálenost nazývá normalizovaná euklidovská vzdálenost a je definována vzorcem (3.1.16).

$$d(X, Y)_{normED} = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2}} \tag{3.1.16}$$

Kde σ je singulární číslo matice odchylek A, jeho druhá mocnina je rovna vlastnímu číslu kovarianční matice $C = AA^T$. Normalizovaná euklidovská vzdálenost lze použít pouze v případě, že se pracuje se všemi vlastními vektory a vlastními čísly.

3.1.3 Shrnutí rozpoznávání pomocí Eigenfaces

Již v úvodu bylo zmíněno, že se jedná o jednu z prvních v praxi použitelných metod. Princip rozpoznávání na základě eigenfaces se dále uplatňuje jako součást komplexnějších způsobů rozpoznávání, jedním z nich je dále zmíněný AAM.

Při implementace rozpoznávání osob podle obličejů byla implementována právě tato metoda, její teoretické pochopení je stěžejní pro implementaci.

3.2 Rozpoznávání pomocí AAM

Algoritmus AAM pracuje s kombinovaným modelem, který se skládá z tvaru – kontury a textury, jenž je uvnitř tvaru.

Princip AAM spočívá v porovnávání statistického modelu objektu s objektem v obraze. Uplatnění je podle zdrojů vcelku úspěšně používáno při interpretaci lékařských snímků, detekcí a rozpoznávání obličeje a stopování objektů ve videu. Skládá se ze dvou částí

1. Vytváření statistického modelu vzhledu.
2. Nalezení transformace modelu na vstupní obraz.

Tvorba statistického modelu se skládá ze sběru informací ve dvou doménách, v doméně obrysu – tvaru a v doméně textury. Tyto dvě jsou nakonec spojeny do jednoho modelu, statistického modelu vzhledu.

Transformace modelu na vstupní obraz je řešena pomocí regresní analýzy, která minimalizuje rozdíl mezi zkoumaným obrazem a syntetickým obrazem vytvořeným ze statistického modelu.

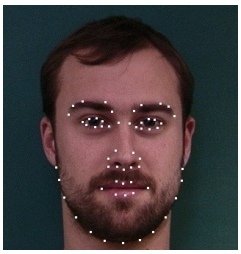
3.2.1 Vytvoření statistického modelu

Model se skládá ze dvou částí, jedna popisuje tvar obličeje a druhá jeho podobu, neboli texturu. Princip spočívá ve vytvoření průměrného modelu každé domény a zjištění jeho odchylek od trénovacích obličejů v dané doméně. Ty jsou zpracovány pomocí PCA, výsledkem je tedy promítnutí do prostoru dané domény – tvarů a textury a získání parametrů – souřadnic.

3.2.1.1 Statistický model tvaru

Trénovaná data musí být anotovaná pomocí tzv. Landmarks, dále použitý pojem jsou definující body, což jsou body, definující anatomii obličeje. Dělí se do tří základních skupin.

1. Anatomicky důležité body.
2. Matematicky významné body - extrémy.
3. Body upřesňující tvar – aproximační.



Ilustrace 3.2.1: Landmarks ze sady [22].

Celý tvar je definován více dimenzionálním vektorem, který vznikne konkatenací jednotlivých souřadnic.

$$X = (x_1, x_2, \ldots, x_n; y_1, y_2, \ldots, y_n)^T \tag{3.2.1}$$

Tvary obličejů, ze kterých se tvoří statistický model, musí být před získáním průměrného modelu co nejlépe zarovnány. Toho se dosáhne pomocí Procrustovy analýzy.

Algoritmus zobecněné Procrustovy analýzy

1. Vyber první tvar jako počáteční odhad průměrného tvaru.
2. Zarovnej všechny ostatní tvary podle průměrného tvaru.
3. Vypočítej nový průměrný tvar ze zarovnaných tvarů. Tím, že těžiště jednotlivých tvarů je zarovnáno do počátku, můžeme průměrný tvar vypočítat jako průměr.
4. Nekonverguje-li průměrný tvar, vrať se na bod 2.

Konvergence v kroku 4. je definována jako míra změny oproti předešlému průměrnému tvaru. Je-li konvergence dostatečně malá, může se iteraci zastavit. Zarovnávání se podle [2] dá algoritmicky shrnou do několika kroků.

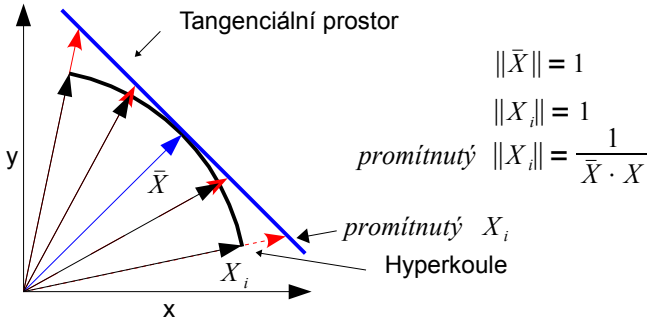
Algoritmus zarovnání

1. Vypočítej těžiště každého tvaru
2. Nastav velikost tvaru ekvivalentní s porovnávaným. Popř. Normalizuj velikost tvarů $\|X\| = 1$
3. Zarovnej tvary podle těžišť. $(\bar{x}, \bar{y}) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i \right)$
4. Zarovnej tvary podle rotace.

K zarovnání podle rotace může posloužit metoda dekompozice matice SVD uvedena v kapitole PCA. Transformační matici rotace pak lze získat z (3.2.2) zdroj [2].

$$SVD(A) = U\Sigma V^T \quad R = UV^T \quad R = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} \tag{3.2.2}$$

Ve chvíli, kdy se tvary před zarovnáním normalizují, tak bod v prostoru, definován právě vektorem tvaru, leží na povrchu hyperkoule o poloměru 1 – normalizovaný vektor má velikost právě 1. Nepříjemnou vlastností tohoto hyperprostoru je jeho nelinearita, která způsobí, že euklidovská vzdálenost neodpovídá vzdálenosti v zakřiveném prostoru. Jde o typický příklad měření vzdálenosti na varietě (manifold), vezme-li se v potaz měření vzdálenosti mezi dvěma relativně blízkými body na zeměkouli, smí se aproximovat na velikost úsečky mezi nimi. V případě, že by se ale měřila vzdálenost mezi Prahou a Sydney (trajektorie tvoří téměř polovinu kružnice), pak by euklidovská vzdálenost byla značně rozdílná od vzdálenosti po povrchu. Tato nelinearita se řeší promítnutím tvaru do Kentova prostoru (tangenciální prostor), který je definován jako veškeré kolmé vektory na vektor průměrného tvaru Ilustrace 3.2.2. Tangenciální prostor je lineární.



Ilustrace 3.2.2: Tangenciální prostor průměrného tvaru a promítnuté tvary do něj.

Po promítnutí zarovnaných tvarů se aplikuje metoda PCA, každý tvar poté lze definovat jako vektor parametrů, který vznikne zobrazením do prostoru vlastních vektorů kovarianční matice. Aplikace PCA je totožná s uvedením v kapitole 3.1.1 PCA – Principal Component Analysis.

3.2.1.2 Statistický model vzhledu - textury

Podobně jako statistický model tvarů, model textur reprezentuje variace, tentokrát ale uvnitř dané množiny textur. K vymezení textury, se kterou se pracuje slouží definovaný tvar obličeje. Typicky v případě obličeje půjde o pixely definované uvnitř ohraničujícího tvaru. Textura konkrétního obličeje

se namapuje do průměrného tvaru, který se získá Procrustovou analýzou. Namapování se provádí pomocí warping funkce. Tento postup se iteruje přes všechna data v trénovací množině. Po té, co jsou textury konkrétních obličejů namapovány do průměrného tvaru, následuje výpočet průměrné textury. Opět je použita metoda PCA a výsledkem je statistický model.

Texturu reprezentuje vektor (3.2.3), jehož hodnoty jsou intenzity pixelů uvnitř tvaru obličeje.

$$T = [t_1, t_2, \ldots, t_m]^T \tag{3.2.3}$$

Algoritmus získání modelu textury

1 Namapovat texturu ze vstupního tvaru do průměrného tvaru
  1.1 Potřeba definice warp funkce.
  1.2 Vytvořit trojúhelníkovitou síť pro warp funkci.
  1.3 Pomocí barycentrických souřadnic transformovat obsah ze vstupního trojúhelníku do cílového.
  1.4 Za použití algoritmů interpolace dodefinovat sporné pixely.
  1.5 Normalizovat model texturu vzhledem k různým světelným podmínkám.
2 Vypočítat průměrnou texturu.
3 Pomocí PCA získat vektory parametrů, popřípadě redukovat prostor.

Triangulace vytvoří trojúhelníkovitou síť ze vstupních bodů definujících tvar. Triangulace by měla produkovat síť, jejíž trojúhelníky se co nejvíce budou blížit rovnostranným trojúhelníkům. Tuto vlastnost má Delaunayova triangulace, která je použita pro vytvoření sítě, jejíž trojúhelníky slouží jako konvexní obálka pro popis množinu pixelů, jenž leží uvnitř. Delaunayova triangulace staví na podmínce, že opsaná kružnice svého trojúhelníku neobsahuje žádné další body jiných trojúhelníků.

Warpovací funkce provádí transformaci z jednoho obrazu do jiného. Vstupní obraz je anotován body, které definují tvar. Tyto body po triangulaci definují trojúhelníky, skrze které popíšeme po částech afinní transformaci. Transformaci pomocí warp funkce poté lze provést skrze barycentrické souřadnice. Po částech afinní je transformace proto, že se provede vždy z jednoho trojúhelníku tvaru na vstupu do ekvivalentního trojúhelníku v průměrné formě.

Barycentrické souřadnice se využívají k popisu bodu uvnitř trojúhelníku T pomocí vrcholů $v_1 = [x_1; y_1], v_2 = [x_2; y_2], v_3 = [x_3; y_3]$ a jejich vah α, β, γ. Bod B lze vyjádřit jako váženou sumu vrcholů.

$$B = \alpha v_1 + \beta v_2 + \gamma v_3 \tag{3.2.4}$$

Barycentrickými souřadnicemi se nazývají právě váhy α, β, γ, pro které platí

$$\alpha + \beta + \gamma = 1 \tag{3.2.5}$$

Převod bodu $B = [x, y]$ z euklidovského souřadného systému do barycentrických souřadnic je popsán rovnicemi (3.2.6).

$$\alpha = 1 - (\beta + \gamma)$$
$$\beta = \frac{x_3 y - x_1 y - x_3 y_1 - x y_3 + x_1 y_3 + x y_1}{- x_2 y_3 + x_2 y_1 + x_1 y_3 + x_3 y_2 - x_3 y_1 - x_1 y_2} \qquad (3.2.6)$$
$$\gamma = \frac{x y_2 - x y_1 - x_1 y_2 - x_2 y + x_2 y_1 + x_1 y}{- x_2 y_3 + x_2 y_1 + x_1 y_3 + x_3 y_2 - x_3 y_1 - x_1 y_2}$$

Rovnice (3.2.4) je třeba vypočítat pro každý bod vstupního trojúhelníku. Pomocí barycentrických souřadnic se určí, zda určitý bod leží vně, na nebo uvnitř trojúhelníku.

- Pro bod B uvnitř trojúhelníku T platí $0 < \alpha, \beta, \gamma < 1$
- B leží na trojúhelníku $0 \leq \alpha, \beta, \gamma \leq 1$
- V ostatních případech leží bod mimo trojúhelník T.

Popsáním statistického modelu textur byla získána druhá část statistického modelu pro AAM. Výchozím bodem se stal průměrný tvar získaný ve statistickém modelu tvarů. Do tohoto tvaru se transformují pomocí warp funkce textury trénovací množiny dat. Přitom se normalizují s ohledem na světelné podmínky. Interpolace se ke slovu dostává ve chvíli, kdy transformací textury vstupního tvaru do průměrného vznikají nedefinované hodnoty. Průměrný model textury, který se získal je zpracováváme stejně jako průměrný tvar pomocí PCA.

3.2.1.3 Statistický model vzhledu

Statistický model vzhledu je kombinace modelu tvarů a modelu textur.

$$b_{aam} = \begin{pmatrix} W_s \cdot b_s \\ b_t \end{pmatrix} = \begin{pmatrix} W_s \cdot P_s^T (X - \bar{X}) \\ P_t^T (T - \bar{T}) \end{pmatrix} \qquad (3.2.7)$$

Kde b_s je vektor parametrů tvaru, b_t vektor parametrů textury, oba jsou výsledkem PCA. W_s je diagonální matice vah mezi intenzitou pixelu a vzdáleností.

3.2.2 Algoritmus AAM

Podobně jako u ASM je vhodné aplikovat algoritmus na několika úrovních rozlišení pro urychlení výpočtu a vyhnutí se nevhodné konvergenci.

Algoritmus AAM

- Model se umístí přibližně do středu hledaného objektu s počátečními parametry pozice modelu.
- Zjistí se rozdíl modelu od obrázku pod ním a lineární regresí se odvodí změnu parametrů, tak aby byla chyba mezi syntetickým obrázkem generovaným modelem a skutečným obrazem co nejmenší.
- Pokud se chyba od minulé iterace nezměnila, může se výpočet ukončit a za výsledek považovat parametry v poslední iteraci. Jinak se vygeneruje nový model a pokračuje se krokem

Ilustrace 3.2.3: Algoritmus AAM, zleva přiložení průměrného modelu, hledání transformace modelu, dokončená transformace, původní obrázek.

3.3 Shrnutí kapitoly Rozpoznání obličeje

Z algoritmů, které se pro rozpoznávání používají, byla popsána metoda Eigenfaces, která pracuje s celým obrazem obličeje a algoritmus AAM, který bere v úvahu tvar obličeje a samotný obličej, bez okolního pozadí. V úvodu byly zmíněny metody LDA a ICA. Text se jim vůbec nevěnuje, tak alespoň stručné shrnutí jejich vlastností. Analýza nezávislých komponent ICA se často používá jako vylepšení PCA. PCA nerozlišuje mezi vnitro-třídní a mezi-třídní závislostí. ICA toto rozlišuje a je schopna nalézt bázi, ve které separuje statisticky nezávislá data.

LDA nalezne lineární kombinaci příznaků, které co nejlépe rozdělí dvě či více tříd. Metoda se snaží data promítnout tak, aby střední hodnoty tříd byly co nejrozdílnější, zatímco variance uvnitř třídy co nejmenší. Jedná se o lineární klasifikátor. Lze se setkat s pojmem Fisherface, což obdobně jako u Eigenface značí bázi, do které se data promítají.

Metoda eigenfaces je základní způsob jak rozpoznávat, používá PCA, která byla detailně popsána a která, jak se ukázalo u AAM, má použití v dalších rozpoznávacích algoritmech. U AAM byla popsána tvorba modelu a stručně způsob, jak model transformovat na vstupní obličej.

4 Návrh implementace rozpoznávání

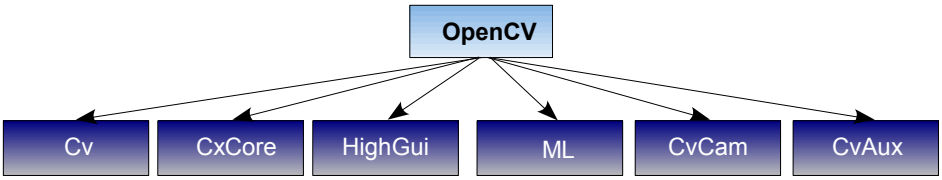
V předešlých kapitolách byly popsány algoritmy potřebné pro sestavení programu, který v obraze detekuje obličeje a na jejich základě se pokusí odhadnout identitu. V této kapitole budou zmíněny podmínky a předpoklady, ze kterých bude implementace vycházet. Budou vybrány konkrétní metody k implementaci a konkrétní platforma, na které se bude stavět.

4.1 Implementační platforma

Při zpracování digitálního obrazu je potřeba rychlého přístupu do paměti, dalším faktorem výběru je možnost použít již existující knihovny daného jazyka. V tomto směru je ideální použití C++, jazyk s možností rychlé práce s pamětí a low level operacemi, ale zároveň s prvky objektově orientovaného programování. Druhou devizou je fakt, že existuje OpenCv knihovna, která je postaven na jazyku C a C++. Použití knihovny je zaštítěno BSD licencí. Je platformně nezávislá, existuje pro Windows, Mac OS X, Linux a dokonce i pro různá embeded zařízení. Původně se jednalo o knihovnu vyvíjenou v laboratořích Intelu, později byl zveřejněn kód a změněna licenční politika.

OpenCV, Open Source Computer Vision Library, je knihovna s množstvím funkcí pro použití v počítačovém vidění. Knihovnu lze použít v několika jazycích, existují wrappery pro Python a .NET technologie, ideální ale je používat OpenCv s C/C++.

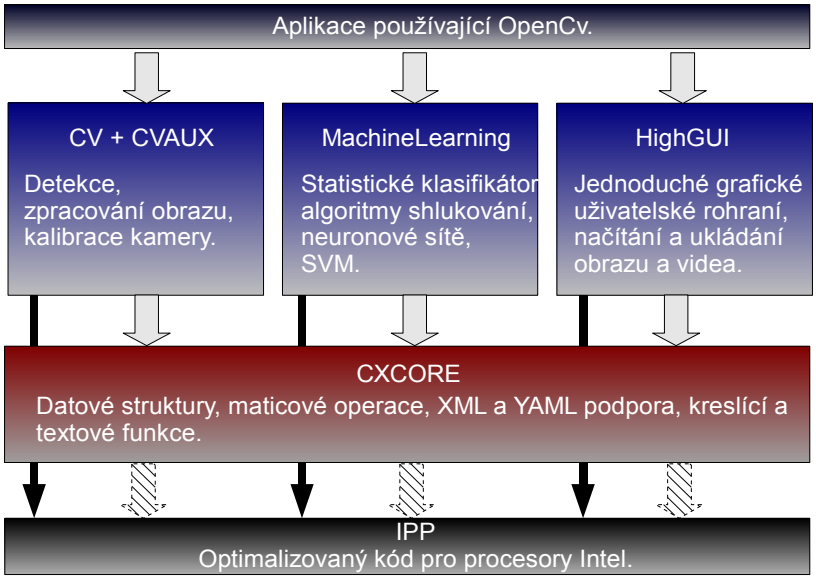
Knihovna se dělí do několika základních modulů, které mají specifickou funkčnost.



Ilustrace 4.1.1: OpenCv knihovna a její základní dělení do modulů.

Modul Cv	Obsahuje funkce pro zpracování obrazu, analýzu struktury obrazu, rozpoznávání.
Modul CxCore	Obsahuje datové struktury, funkce pro maticovou algebru, funkce pro transformaci mezi daty, funkce pro uchování dat – uložení na disk, správce paměti, práci s chybami, funkce pro kreslení a
HighGui	Uživatelské rozhraní, ukládání a načítání obrazových dat, videa.
ML	Machine Learning, funkce pro klasifikaci a shlukování. Bayesovské klasifikátory, Neuronové sítě, Support Vector Machine, rozhodovací stromy.
CvCam	Funkce pro práci s kamerou, připojení, přehrávání atd.
CvAux	Experimentální ale i „depracted“ funkce, obsahuje mimo jiné funkce pro získání vlastních vektorů pro rozpoznávání metodou Eigenfaces. V projektu z důvodů „depracted“ označení není použita.

Struktura knihovny je znázorněna na Ilustrace 4.1.2.



Ilustrace 4.1.2: Struktura knihovny OpenCV.

IPP modul poskytuje rychlejší zpracování funkcí na procesorech Intel, je již ale komerční, v projektu tedy není použitý.

Pro aplikaci tedy stačí knihovna OpenCV a programovací jazyk C++. Díky použití knihovny, která podporuje širokou škálu grafických formátů lze konstatovat, že půjde zpracovat nejrozšířenější formáty jako BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF.

4.2 Omezující podmínky a předpoklady

V práci již několikrát načaté téma podmínek a předpokladů, se kterými se pracuje, je explicitně popsáno zde. Nejdůležitější podmínkou a předpokladem je, že obličej na fotografii bude zpříma směrem ke kameře. Obličej by měl být celý viditelný, neměl by být překrytý optickou překážkou, neměl by být zahalen v burce, pokud možno bez slunečních brýlí atp. Velikost obličeje by neměla přesáhnout velikost fotografie a zároveň by neměl být příliš malý. V opačném případě oba tyto fakty predikují, že by nemusel být obličej vůbec detekován. Stejná podmínka se týká jeho natočení, vše je závislé na použitém detektoru, v aplikaci je použita kaskáda z knihovny OpenCv, jež je trénovaná na obličeje v přímé pozici. Obličej z profilu pravděpodobně nebude vůbec detekován.

Samostatnou kapitolou by mohly být světelné podmínky, se kterými se počítá jako s ideálními, což v praxi téměř nenastane. Zpracování histogramu a podobné postupy potlačení vlivu osvětlení zapříčiní menší úspěšnost identifikace obličeje.

Výraz tváře taktéž ovlivňuje míru úspěšnosti identifikace, ale tento fakt lze potlačit zvolením vhodné trénovací sady, kde obličeje budou mít výrazy pro typické nálady.

4.3 Použité algoritmy

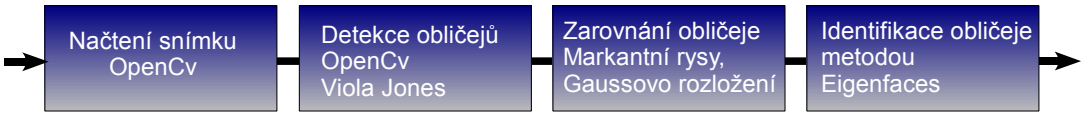
Pro detekci byl vybrán algoritmus Viola Jones, který implementuje knihovna OpenCv, tím odpadá pracné trénování vlastního klasifikátoru. Nicméně detekce je na klasifikátoru v určité míře nezávislá, neboť lze použít i jiný, než dodaný klasifikátor, podmínkou je zachovaní rozhraní pro jeho uložení.

Detekce je díky OpenCv vcelku přesná a rychlá, samozřejmě existují lepší algoritmy a rychlejší implementace, pro potřeby aplikace ale tento detektor zcela vyhovuje.

Po detekci obličeje je třeba jej zarovnat, zde se jeví nejideálnějším algoritmem ASM, pro svou jednoduchost je ale použito zarovnání podle markantních rysů, v případě ale, že detekce uvnitř obličeje selže, je velikost detekčního rámce konstantně upravena na základě empirie.

Pro detekci je použita metoda Eigenfaces, která je do jisté míra ověřená. Její úspěšnost ale rapidně klesá při nedokonalém zarovnání, což je vyváženo rychlostí, jakou je zpracována. Její použití je nicméně také dáno slušnou podporou lineární algebry knihovny OpenCv.

Aplikaci znázorňuje Ilustrace 4.3.1.



Ilustrace 4.3.1: Schematický náčrt propojení algoritmů.

4.4 Uživatelské nastavení

Každý ze systému aplikace, to jest detekce, zarovnání a rozpoznání, lze určitým způsobem konfigurovat a ovlivnit tím více či méně jeho funkčnost a výsledky. Z tohoto hlediska je výhodné mít parametry uloženy externě a aplikaci je při spouštění předávat. Knihovna OpenCv má podporu práce s XML soubory, která v tomto směru poskytuje jistou volnost.

Při detekci lze detektor parametrizovat různými proměnnými. Jedním z parametrů je dodaná natrénovaná kaskáda, kterou se klasifikuje. Ta je ve formátu XML externě mimo program. Dále lze detektoru nastavit od jaké velikosti má začít obraz skenovat, za jakých okolností má detekci ukončit a krok, se kterým má při skenování měnit velikost skenovacího výřezu.

U zarovnání detekovaného obličeje pomocí polohy markantních rysů je třeba počítat s faktem, že se nepodaří všechny detekovat, pak nezbývá než se pokusit zbývající odhadnout pomocí Gaussova rozložení. Parametry Gaussova rozložení je vhodné mít možnost specifikovat externě.

Samotnému rozpoznávání pak lze nastavit velikost obrazu, kterým je obličej reprezentován, to notně ovlivní velikost natrénovaných dat, zároveň tak lze experimentovat, jaké rozlišení obrazu obličeje stačí k pokrytí informace o identitě.

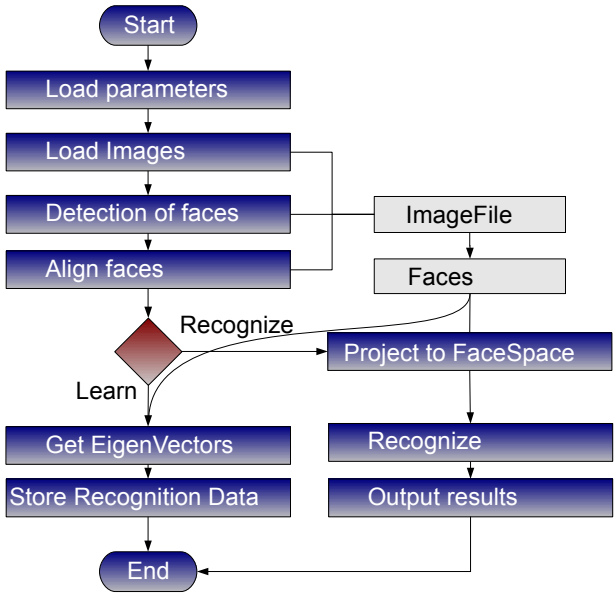
Tyto všechny parametry by měly být modifikovatelné externě mimo program, tedy ideálně uložené v XML souboru, ze kterého je aplikace načte.

5 Implementace rozpoznávání

Aplikace rozpoznávání obličejů ve fotografii je implementována v C++ za pomocí OpenCv knihovny za použití vývojového prostředí Microsoft Visual Studio 2008. Open Source Computer Vision Library – OpenCv je použita ve verzi 1.1pre1[1] z října roku 2008, nicméně byla úspěšně testována i stabilní verze 1.0.

Kapitola implementace popisuje aplikaci v tzv. „Top-Down“ modelu. Jednotlivé komponenty jsou zapojeny do abstraktního sytému, který popisuje běh aplikace. Ten se skládá ze dvou módů learn a recognize. Podrobně je popsána fáze zpracovaní a tok dat v programu. V kapitole jsou popsány všechny použité hlavní komponenty programu, které tvoří samostatné třídy použité v jednotlivých fázích zpracování. Poslední podkapitolou je popis formátu konfiguračního souboru, který nabízí modifikaci parametrů ovlivňující detekci i rozpoznávání.

5.1 Módy aplikace



Ilustrace 5.1.1: Módy aplikace - Learn a Recognize. ImageFile a Faces zaobaluje obličeje napříč aplikací.

Program pro rozpoznávání pomocí obličeje pracuje ve dvou módech learn a recognize. V módu learn se načtou obrázky, ze kterých se extrahují obličeje, vypočtou se parametry pro rozpoznávání metodou Eigenfaces a ty se uloží.

V módu rozpoznávání – recognize se načtou vstupní obrázky, extrahují se obličeje a ty se promítnou do prostoru obličejů, jehož parametry jsou získány v módu učení – learn. Výsledky se tisknou na standardní výstup. Módy aplikace znázorňuje Ilustrace 5.1.1. Obličeje napříč programem

1 Verze OpenCV 1.1pre1 je v přiložena s programem na CD médiu.

putují nejdříve pod strukturou `ImageFile`, která se vztahuje právě k jednomu obrázku a nese informaci a všech obličejích v něm. Po fáze zarovnání jsou tyto obličeje transformovány do třídy `Faces`, která se stává jejich kontejnerem a která obsahuje pole všech obličejů, ke každému z nich si uchovává potřebná metadata v podobě ID, jeho pozice v obrázku a cestu k obrázku. Jejich popis je v kapitole 5.2.6 Komunikace mezi fázemi výpočtu.

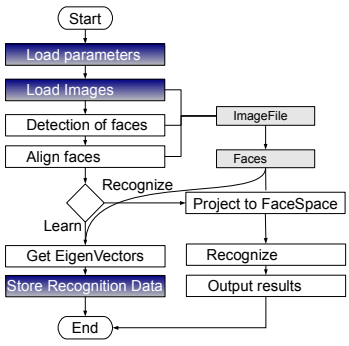
5.2 Fáze zpracování

Fáze zpracování popisuje tok dat a jejich zpracování napříč programem. Kopíruje schéma 5.1.1, a popisuje třídy, které jednotlivé kroky schématu zprostředkovávají. Zmíněné metody a popis jejich funkcí pak uceluje přehled celého zpracování.

Podkapitoly zpracování se tedy dělí na základě schématu 5.1.1. Objektem zájmu budou třídy `FileManager`, která zjišťuje vstupní a výstupní data programu, `Detection` pro detekci obličejů v obraze, `FaceAlign`, jenž obsahuje několik metod pro zarovnání detekovaného obličeje, `PCA` třídu, která zaobaluje výpočty pro získání vlastních vektorů, nebo promítnutí do báze, a konečně třída `Recognition` jenž v případě módu rozpoznávání poskytuje metody pro určení identity.

5.2.1 Načtení parametrů a obrázků

V první fázi je nutné načíst parametry potřebné pro chod programu, mezi nimi i o který mód běhu aplikace se jedná. Ve chvíli, kdy jsou veškeré potřebné parametry k dispozici, je třeba zprostředkovat načítání obrázků.



Tyto funkce a k tomu ukládání dat zajišťuje třída `FileManager`. Ta obsahuje metody pro rozbor konfiguračního souboru obsahující potřebné parametry, rozbor vstupního souboru, jenž definuje obrázky, které budou zpracovány a posléze i samotné obrázky.

```
class FileManager
{
public:
      FileManager(void);
      ~FileManager();
      void loadConfigFile(std::string path, Params *p);
      bool openFileList(std::string ListFile);
      ImageFile* operator[](unsigned int i);
      unsigned int getSize(void);

      void loadRecognitionData(std::string fileName, PCA *pca, Faces
      *faces);
      void saveRecognitionData(std::string fileName, PCA *pca, Faces
      *faces);

      void saveFaces(std::string path, Faces *faces);
      //...
```

```
private:
      std::vector<ImageFile*> __files;
};
```

FileManager je tedy s jistou dávkou abstrakce opravdu souborový manažér, který funguje jako rozhraní pro načítání obrázků, načítání parametrů a zároveň jejich ukládání..

`void loadConfigFile(std::string path, Params *p);`

Metoda provede rozbor konfiguračního souboru formátu xml definovaného řetězcem path a jeho hodnoty uloží do struktury obsahující parametry. Parametr `*p` musí mít alokovanou paměť, jinak při načítání souboru dojde k vyjímce. Struktura `Params` nese veškeré informace z konfiguračního souboru.

`bool openFileList(std::string ListFile);`

Načte soubor definující obrázky ke zpracování (ten je definován v kapitole 5.4 Shrnutí implementace), připraví cesty k obrázkům a uloží si definované identifikační číslo každého obrázku, jenž se v případě učení týká obličeje, v případě rozpoznávání se ID vztahuje pouze k obrázku. Předpokládá se, že při módu učení obsahuje jeden obrázek jeden obličej, v módu rozpoznávání pak může jeden obrázek obsahovat více obličejů. Cestu k obrázku i samotné ID uloží do struktury `ImageFile (5.2.6 Komunikace mezi fázemi výpočtu)`, kterou vloží do svého privátního vektoru `__files`.

`void loadRecognitionData(std::string fileName, PCA* pca, Faces* faces);`

Načítá dříve natrénované parametry, vektory definující bázi prostoru obličejů a parametry definující známé obličeje. Data pro zobrazení do prostoru obličejů ukládá přímo do třídy `PCA`, známé tváře ukládá do třídy `Faces`.

Při trénování jsou vypočtené parametry pro rozpoznávání uloženy do speciálního xml souboru, je nutno rozlišovat konfigurační soubor, soubor definující obrázky a soubor uchovávající data pro rozpoznávání.

`void saveRecognitionData(std::string fileName, PCA* pca, Faces *faces);`

Slouží právě k uložení výše zmíněného souboru uchovávající data pro rozpoznávání. Metoda tedy uloží natrénovaná data, těmi jsou báze prostoru obličejů a vektory parametrů popisující známé obličeje. Soubor může nabývat až několik megabajtů paměti vzhledem k počtu obličejů, které jsou v trénovací množině známých lidí. Velikost se pak přesně odvíjí od počtu vlastních vektorů, které se používají, neboť jeden vlastní vektor má stejný počet hodnot, jako obrázek s obličejem, který se zpracovává `PCA` třídou.

`ImageFile* operator[](unsigned int i);`

Přetížený operátor indexace do pole zajišťuje přístup k načteným obrázkům. Při jeho zavolání je daný obrázek přečten z disku a nahrán do paměti, struktura `ImageFile`, obsahující ukazatel na načtený obrázek, jeho ID a cestu k souboru, je vrácena ve formě ukazatele. O uvolnění paměti obrázku se stará samotná třída `FileManager` skrze destruktor struktury `ImageFile`. Díky tomuto operátoru se třída `FileManager` jeví jako kontejner dat.

`Void saveFaces(std::string path, Faces *faces);`

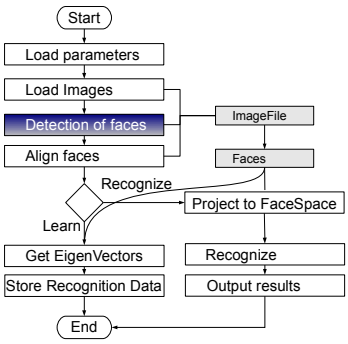
Uloží obličeje ve struktuře `faces`, pro samotný běh aplikace nemá využití, implementována je pro ilustrační použití, lze s ní ukládat mezivýsledky zpracovaných obrázků, podobně i ostatní metody, viz zdrojová dokumentace.

Perzistentní data jsou uložena v souborech formátu xml, těmi jsou konfigurační parametry aplikace i parametry pro rozpoznávání. OpenCv obsahuje podporu zpracování xml. V případě dalšího vývoje, by ale bylo vhodné použít speciální knihovny pro xml, popřípadě implementovat ručně rozbor těchto souborů a to z toho důvodu, že možnosti OpenCv i přes to, že umožňuje definici vlastních typů a tím i automatizaci uložení a načtení tohoto typu z a do xml, jsou v tomto směru znatelně omezené.

Třída `FileManager` má asociován vlastní druh vyjímek, které mohou nastat při zpracování souborů. Ne všechny případy jsou ošetřeny, výjimka je vyvolána v případě chybějících parametrů v konfiguračních souborech, chybějících souborů atd. Informaci, kterou výjimka nese, je pouze text popisující, proč byla vyvolána. Výjimky asociované s třídou `FileManager` dědí z `Exception` deklarovanou v hlavičkovém souboru *Utility.h*.

Rozhraní třídy `FileManager` je přístupné skrze svůj vlastní hlavičkový soubor *FileManager.h*.

5.2.2 Detekce obličeje



K detekci obličejů se využívá knihovna OpenCv, která obsahuje již natrénovanou kaskádu klasifikátorů. Detekci s možným uživatelským nastavením pak v projektu zaobaluje třída `Detection`. Atributy třídy definují kterým klasifikátorem se obrázek prohledává, parametry samotného prohledávání, zda má na standardní výstup tisknout související informace s detekcí jako počet detekovaných objektů a čas samotné detekce.

Vstupní data metody pro detekci jsou ve formě ukazatele na obrázek. Výstup je vektor nalezených obličejů ve formě ohraničujících čtverců.

```
class Detection
{
public:
      Detection(CvHaarClassifierCascade* cascade,
                        bool printDetails = false);
      ~Detection(void);

      void setCascade( CvHaarClassifierCascade *cascade);
      void setScale(double scale);
      void setEqHistogram(bool val);
      void find(const IplImage* img, std::vector<CvRect> &faces);
      void drawDetected(IplImage* img, const CvSeq* det);
      //...
private:
      double      __scale;
      bool  __print;
      bool  __equalizeH;
      //...
};
```

Třídu je třeba inicializovat ukazatelem na klasifikátor, kterým se bude detekovat. Ten je určen skrze parametry konfiguračního souboru, samotný klasifikátor je tedy zpracován třídou `FileManager`, která zajistí jeho načtení do paměti a předání do struktury parametrů `Params` viz 5.2.6 Komunikace mezi fázemi výpočtu.

Níže popsané metody až na výjimku v podobě metody `find()` slouží pouze k nastavení detekce.

`void find(const IplImage *img, std::vector<CvRect> &faces);`

Veřejná metoda zajišťující detekci obličeje, vstupním parametrem je načtený obrázek reprezentován datovým typem `IplImage`[1], výsledek je ukládán do druhého parametru ve formě pravoúhelníků definujících detekovaný obličej.

Metoda zaobaluje důležitou funkci zprostředkovanou díky OpenCv knihovně, jejíž nastavení ovlivní přesnost, dobu vyhodnocení a množství vrácených výsledků.

`void setScale(double scale)`

Nastavení změny měřítka skenovaného obrázku. V případě obrázků s vysokým rozlišením se podvzorkování rapidně sníží doba vyhodnocení. Parametr `scale` se udává v normalizované podobě, tj 80% původní velikosti = 0,8 zatímco 120% původní velikosti = 1,2. Parametr lze měnit skrze konfigurační soubor.

`void setEqHistogram(bool val)`

Nastaví vnitřní přepínač třídy, potlačení invariance osvětlení lze mírně dosáhnout zpracováním histogramu. Parametr je opět nastavitelný skrze konfigurační soubor.

Třída má dále vnitřní přepínač pro vypisování druhotných informací na standardní výstup jako počet nalezených objektů a čas detekce. Nastaví se při inicializaci voláním konstruktoru, lze modifikovat skrze konfigurační soubor.

Poslední uvedenou funkcí bude nativní funkce knihovny OpenCv pro detekci, kterou třída zaobaluje s dalšími režijními metodami a nastavením.

```
CvSeq* cvHaarDetectObjects(const CvArr, CvHaarClassifierCascade, CvMemSto-
rage, double scale, int min_neighbour, int flags, CvSize min_size);
```

Funkce provádí detekci nad vstupem typu `CvArr`, což je abstraktní typ zaobalující například struktury `IplImage` a `CvMat`. `CvHaarClassifierCascade` definuje kaskádu[2], která klasifikaci provádí. `CvMemStorage` je ukazatel na alokovanou paměť pro uložení výsledku detekce. Další parametr `scale` definuje velikost kroku změny měřítka při skenování obrázku. Důležitý parametr ovlivňující přesnost detekce je `min_neighbour`, který udává minimální počet úspěšných detekcí v daném regionu. Příznaky `flags` určují způsob detekce a výslednou rychlost, více viz dokumentace knihovny OpenCv.

Posledním parametrem `min_size` se definuje nejmenší velikost skenovacího výřezu.

V projektu je hodnota `scale` nastavena na `1,1`, `min_neighbour` na `3`, z příznaků je použit pouze `CV_HAAR_DO_CANNY_PRUNING`, jenž zapříčiní přeskočení oblastí bez hran, a jako minimální velikost skenovacího výřezu `min_size` je zvolena `20x20`.

Detekce je prováděna na obraze v odstínech šedi, transformaci barevného prostoru zajišťuje automaticky metoda `find`.

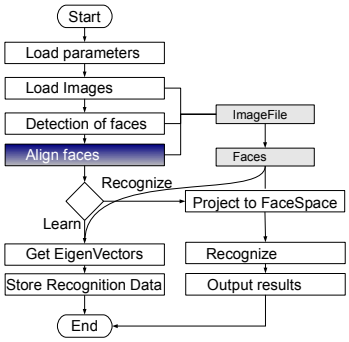
Idkyž není zarovnání obličeje součástí této kapitoly, je dobré zmínit, že pomocí zarovnání obličeje podle markantních rysů je využíváno vedlejšího efektu k validaci detekce obličeje. To

1 Struktura IplImage pochází z Intel Image Processing Library. OpenCv podporuje pouze část formátu `IplImage`. Nicméně je výchozím typem pro obrazová data a jejich metadata v celé knihovně.

2 Teoreticky popsána v kapitole 1.3.2.2 Kaskáda klasifikátorů.

znamená, že nejsou – li později při zarovnávání podle markantních rysů detekovány oči, nos či ústa, je detekovaný obličej prohlášen za nevalidní. Vy výsledcích není přímo zhodnocen tento vliv na detekci, ale s určitou obecností lze říci, že počet správného prohlášení obličeje za nevalidní mírně převyšuje počet špatného prohlášení, že obličej je nevalidní.

5.2.3 Zarovnání obličeje



Zarovnání obličeje následuje vždy před zpracováním metodou PCA Na výsledném zarovnání přímo-úměrně závisí úspěšnost identifikace. Implementovány jsou dva způsoby zarovnání. Prvním z nich je pevně zvolená transformace detekčního okna Viola - Jones detektoru, nicméně lze uživatelsky modifikovat. Druhý způsob je implementován na bázi pozice markantních rysů v obličeji, kterými jsou oči, nos a ústa. Skrze konfigurační soubor je možné měnit jeho parametry vzhledem k použitému detektoru. Zarovnání podle markantních rysů v jistým smyslu koriguje detekci samotného obličeje. Obličej je prohlášen za nevalidní, není li detekováno minimálně jedno oko a zároveň jeden koutek úst.

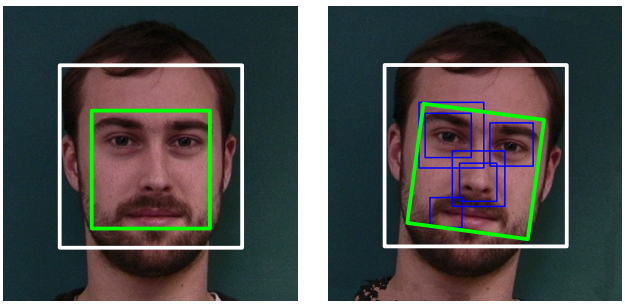
```
class FaceAlign
{
public:
      FaceAlign(bool printDetails = false);
      int noAlign(CvRect &RecFaces, AlignedFace &aligned);
      int basicAlign(CvRect &RecFaces, AlignedFace &aligned);
      int faceEyeAlign(IplImage* img, CvRect& RecFace, AlignedFace
            &aligned);
      void setCascade(CvHaarClassifierCascade* eyeCascade,
            CvHaarClassifierCascade* noseCascade,
            CvHaarClassifierCascade* lMouthCascade, CvHaarClassifier
            Cascade* rMouthCascade);
      void setGauss(Gauss &lEye, Gauss &rEye, Gauss &nose, Gauss
            &lMouth, Gauss &rMouth);
      void setFaceBasicAlignParameters(double scale, double shiftY);
      void setFaceEyeAlignParameters(double dEyes, double dMouth,
      double treshold, double tresholdRot, , CvSize faceSize);
      void setEqHistogram(bool val);

private:
      CvHaarClassifierCascade* __eyeCascade;
      Gauss lEye;
      //...
};
```

První způsob konstantně mění velikost oblasti definující obličej. Empiricky byly zvoleny hodnoty pro posun a zmenšení výřezu detekované oblasti, pracuje se tak s co nejmenším vlivem pozadí. Výsledek znázorňuje Ilustrace 5.2.1 - první zleva, kde bílý výřez definuje detekovaný obličej pomocí OpenCv detektoru, zelený je pak empiricky transformovaný tak, aby zabíral pouze obličej s potlačením pozadí. Použité hodnoty transformující detekční okno uvádí Tabulka 5.1.

Scale faktor	0,65	65% původní velikosti.
Shift faktor	0,08	Posunutí dolů o 8%.

Tabulka 5.1: Empiricky zvolené parametry pro Viola - Jones detektor.

Zarovnání podle polohy markantních rysů, Ilustrace 5.2.1 druhý zleva, tedy očí, nosu a úst je schopno se vypořádat i s mírným natočením obličeje, to jen v případě, že obličej OpenCv detektor najde. Implementace se neobejde bez externě dodaných dat. Po té, co je detekován obličej a ten je předán k zarovnání, je třeba v rámci výřezu obličeje detekovat jeho součásti, konkrétně oči, nos a koutky úst. Jedná se o stejnou proceduru, jako detekce obličeje s tím rozdílem, že se použije natrénovaný klasifikátor na jednotlivé objekty. K tomu byl použit klasifikátor, jenž byl vytvořen v rámci Ústavu Počítačové Grafiky a Multimédií při Fakultě informatiky na VUT [23]. Detekce není zcela ideální, klasifikátor vykazuje poměrně často False Positive Detection[1], proto je třeba detekci ověřit. Validace se provádí zjištěním příslušnosti do prostoru, kde se daný objekt pravděpodobně nachází. Vyhodnotí se tedy dvou-dimenzionální Gaussovo rozložení popsané vzorcem (2.1.1). Parametry takového rozložení byly získány na anotované trénovací sadě BioID [16].



Ilustrace 5.2.1: Zarovnání obličeje, zleva empirické a podle markantních rysů.

Získání parametrů Gaussova rozložení

1 Detekce obličeje v obraze.
2 Získání anotovaných souřadnic objektu: očí, koutků úst, nosu.
3 Normalizace do prostoru $\langle 0\,;\,1 \rangle$.
  3.1 Získání středu detekovaného obličeje.
  3.2 Výpočet vzdálenosti středu anotovaného objektu od středu detekovaného obličeje.
  3.3 Normalizace $\frac{Vzdálenost\ od\ středu}{šířka\ detekčního\ okna\ obličeje}$.
4 Výpočet parametrů ze získaných dat.

Normalizuje se z toho důvodu, že detekční okno obličeje může vždy nabývat jiné velikosti. Rozložení jednotlivých rysů v rámci detekčního okna a po normalizaci znázorňuje Ilustrace 2.1.1: Graf rozložení očí, nosu, levého a pravého koutku úst vzhledem k OpenCv detektoru na datech z [16]. Výsledné hodnoty parametrů shrnuje Tabulka 5.2, byly získány vyhodnocením dat z [16]. Detekce jednotlivých rysů ukazuje Ilustrace 5.2.1, modré detekční okna značí jednotlivé rysy. Lze vidět, že

1 Více v kapitole 1.1.1 Trénování klasifikátoru.

	EX	EY	Kovarianční matice Σ	
Levé oko	-0,19557	0,11748	5,12700E-04	4,50858E-05
			4,50858E-05	5,08469E-04
Pravé oko	0,19660	0,11588	6,85996E-04	4,99836E-07
			4,99836E-07	5,22912E-04
Nos	-0,00084	-0,08346	5,89710E-04	1,93110E-04
			1,93110E-04	2,70894E-03
Levý koutek úst	-0,15359	-0,29514	9,46130E-04	4,29650E-04
			4,29650E-04	2,68936E-03
Pravý koutek ús	0,15471	-0,29538	1,20636E-03	1,46798E-05
			1,46798E-05	2,78562E-03

Vzdálenost očí od horního okraje detekčního rámce.

dEyes	0,125

Vzdálenost úst od dolního okraje detekčního rámce.

dMouth	0,1

Práh odmítnutí detekce.

treshold	0,01

Práh zarovnání rotace.

tresholdRot	0,035

Tabulka 5.2: Parametry dvou-dimenzionálního Gaussova rozložení jednotlivých rysů a parametry zarovnání.

některé rysy jsou detekovány vícekrát, díky pravděpodobnostní funkci rozložení se určí, která detekce je brána v potaz.

Může se stát, že je rys detekován naprosto špatně, častým případem je například část nosu detekovaná jako oko, popřípadě koutek úst, v takovém případě je díky zvolenému prahu detekce zamítnuta. Jelikož parametry platí pro normalizovaný prostor, nemusí PDF[1] vracet hodnoty nižší než 1. Na základě těchto hodnot byl zvolen práh, který definuje přijetí detekce jako správné, nebo nesprávné.

```
int noAlign(CvRect &RecFaces, AlignedFace &aligned);
```

Metoda pouze kopíruje data z `RecFaces` do `aligned`, úhel rotace nastaví na `0`.

```
int basicAlign(CvRect &RecFaces, AlignedFace &aligned);
```

Provede zmenšení výřezu a jeho posun dolů podle hodnot, které uvádí Tabulka 5.1. Hodnoty lze měnit skrze konfigurační soubor.

```
int faceEyeAlign(IplImage* img, CvRect& RecFace, AlignedFace
      &aligned);
```

Zarovnání podle markantních rysů, před použitím metody je třeba definovat parametry Gaussova rozložení. V případě, že není detekováno ani jedno z obou očí, nos, popřípadě ústa, metoda vrací

1 Probability Density Function – pravděpodobnostní funkce rozložení.

hodnotu `-1` a zpracovávaný obličej je prohlášen za nevalidní. V případě, že vzdálenost detekovaných očí v ose y je vyšší než daný práh, je provedeno zarovnání rotace, velikost úhlu je uložena do struktury `aligned`.

```
void setEqHistogram(bool val);
```

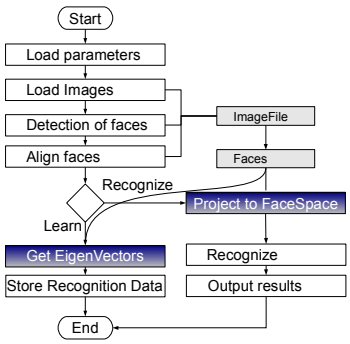
Zarovnání podle očí vyžaduje detekci markantních rysů, podobně jako u detekce obličeje, lze zvolit, zda se má před detekcí vyvážit histogram a pokusit se tím omezit vliv osvětlení a dalších nepříjemných faktorů.

```
void setFaceEyeAlignParameters(double dEyes, double dMouth, double
treshold, double tresholdRot, CvSize faceSize);
```

Parametr `dEyes` udává míru vzdálenosti očí od horního okraje detekčního rámce, `dMouth` pak vzdálenost úst od dolního rámce. Obě míry jsou normalizované `<0;1>`. Práh, který určí zda detekovaný objekt leží v předpokládaném místě určuje `treshold`. Poslední parametr udává práh vzdálenosti očí v ose Y pro provedení rotace pro zarovnání očí do vodorovné přímky. Poslední parametr definuje velikost okna, do které je detekovaný obličej promítnut a ve kterém se hledají markantní rysy.

5.2.4 Aplikace PCA

Získání vlastních vektorů a promítnutí obličeje do báze jimi definované obstarává třída `PCA`. Metody třídy pracují již se zpracovanými a zarovnanými obličeji, které jsou zapouzdřené v kontejneru `Faces`. V režimu rozpoznávání musí být objekt typu `PCA` inicializován skrze `FileManager`, který dodá data z konfiguračního a datového souboru. V případě učení se, lze získaná data uložit na disk opět pomocí objektu typu `FileManager`.



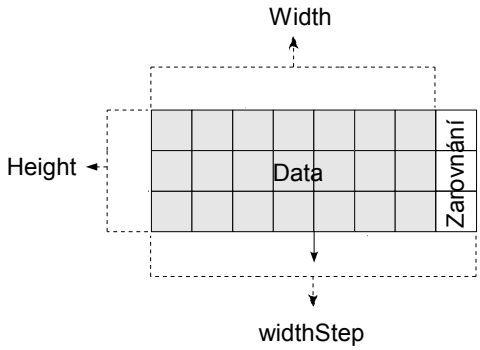

```
class PCA
{
public:
      PCA(void);
      ~PCA(void);

      void calcEigen(Faces* faces);
      void projectToEigenSp(Faces* faces);
      //...
};
```

Implementace celé třídy vychází z kapitoly 3.1.1 PCA – Principal Component Analysis. Počet uložených vlastních vektorů není striktně daný a lze nastavit.

Z velké části jde o maticové operace, které knihovna OpenCv implementuje na velmi dobré úrovni, nicméně existují implementační specifika, která je dobré zmínit. Před samotným popisem implementace je třeba nastínit strukturu dat, s nimiž se pracuje, konkrétně struktury `IplImage`, `CvMat` a `CvArr`, jenž jsou součástí OpenCv.

Přesnou definici struktur lze dohledat v dokumentaci, z hlediska implementace nejsou ani tak podstatná metadata, jako fyzický způsob, jímž jsou data uložena v paměti. Obraz lze chápat jako

matici bodů. Ta je v paměti uložena po řádcích, které jsou zarovnány na určitý počet bajtů, nejčastěji na 4. Implementace dvourozměrného pole v jazyce C je provedena díky jednorozměrnému poli, do kterého se indexuje dvěma ukazateli – ukazatel řádku a ukazatel sloupce. Stejně implementuje 2D pole knihovna OpenCv ve struktuře `IplImage` i `CvMat` s tím rozdílem, že velikost řádku je zarovnána na daný počet bajtů. Situaci znázorňuje Ilustrace 5.2.2. To znamená, že prvek i-tého řádku ve sloupci y, leží na indexu $i * widthStep + y$.



Ilustrace 5.2.2: Fyzické uložení dat v IplIamge a CvMat.

Dalším důležitým atributem je typ dat, obrázky se kterými se v objektu typu PCA pracuje, jsou v odstínech šedi, to znamená, že jeden pixel je reprezentován jedním kanálem o 8 bitech. Jeden pixel je tedy uložen v jednom bytu : 0 - 255 hodnot. OpenCv tento typ definuje jako `CV_8UC1` (8 bitů, unsigned char, počet kanálů 1). Nicméně většina funkci operující s maticemi vyžaduje vstupní typ dat `CV_32FC1` (32 bit float hodnota na jeden kanál). Je tedy nutné typ dat v případě použití takové funkce konvertovat. Důležitý je fakt, že `CvArr`, `CvMat` a `IplImage` zaobalují ta samá data ale vždy s rozdílnými metadaty. Jejich vztah znázorňuje Ilustrace 5.2.3.

Ilustrace 5.2.3: Vztah CvArr, CvMat a IplImage.

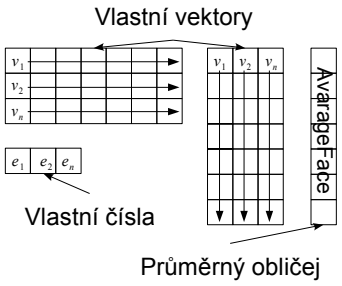
Většina funkcí operující s maticemi má typ vstupního parametru `CvArr`, díky tomu umí zpracovat data jak `IplImage` struktury, tak `CvMat` struktury. `CvArr` je pouze „abstraktní“ datový typ generalizující `CvMat` a `IplImage` struktury.

Shrnutí předešlých odstavců spočívá ve dvou významných faktech, první je, že funkce knihovny OpenCv pracující s maticemi přijímají jako parametr jak `IplImage` tak i `CvMat` díky „abstraktnímu“ typu `CvArr`. Druhým důležitým faktem je, že data v těchto maticích musí být často typu `CV_32FC1`. Tedy je nutná konverze.

```
void calcEigen(Faces* faces);
```

Metoda ze vstupního kontejneru obličejů, jenž jsou již zarovnány a jejich velikost je unifikována na hodnotu definovanou uživatelem (výchozí je 128×128), vypočítá kovarianční matici, ze které následně pomocí SVD metody zjistí vlastní vektory a vlastní čísla, která reprezentují bázi prostoru obličejů. Velikost obličeje musí být zarovnána po řádcích na 4 bajty (96×96*;* 128×128 *atd...*) a jejich počet musí být minimálně 2.

Vlastní vektory jsou uloženy do privátních atributů třídy `PCA` ve formě dvou matic. Jedna obsahuje vlastní vektory sloupcové a druhá řádkové. Do privátních atributů je také uložen průměrný obličej ve formě matice o jednom sloupci a počtu řádků, jaké je rozlišení obličeje. Tato data jsou skrze `FileManager` uložena do datového souboru typu xml pro potřeby rozpoznávání.



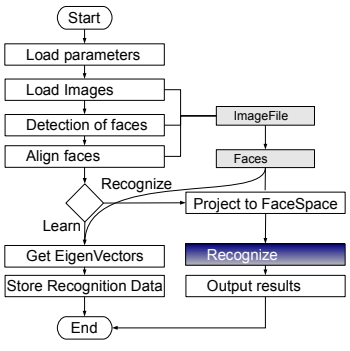
Ilustrace 5.2.4: Formát uložení vlastních vektorů v PCA třídě.

Matice vlastních čísel, která se získá pomocí SVD má velikost *počet tváří* × *počet tváří* , ze kterých je získána kovarianční matice. Relevantních hodnot, které leží na diagonále, je *počet tváří* − 1 , poslední je rovna 0 . Vlastní číslo udává míru variance, a lze uživatelsky nastavit s kolika vlastními čísly a vektory se pracuje. Nicméně počet je omezen počtem tváří, ze kterých se počítá kovarianční matice. Vlastní vektory jsou před uložením normalizovány.

```
void projectToEigenSp(Faces* faces);
```

Zajišťuje promítnutí do báze obličejů, kterou tvoří vlastní vektory. Před zavoláním této metody musí být objekt typu PCA inicializován potřebnými daty – vlastní vektory a průměrný obličej. Obličeje v kontejneru `Faces` jsou promítnuty a jejich promítnuté souřadnice jsou uloženy zpět do kontejneru. Jedná se o vektor „parametrů“ z kapitoly 3.1.1.2 Výpočet parametrů pro rozpoznávání.

5.2.5 Fáze rozpoznání



Fáze rozpoznání je vyhodnocení vzdálenosti vektoru definujícího zkoumaný obličej od vektorů definujících známé obličeje. Využívá se dvou metrik, Euklidovské a Mahalanobisovy vzdálenosti, obě jsou popsány v kapitole 3.1.2 Proces rozpoznávání. V kapitole 3.1.2, jsou popsány čtyři různé varianty, které mohou při rozpoznávání nastat, jejich řešení vyžaduje zvolení prahu, který je dost závislý na datech. Z toho důvodu nebyl práh implementován, a tak aplikace přiřadí nejpravděpodobnější ID i neznámému obličeji. Výstupem programu je pak nejpravděpodobnější ID hodnota a vzdálenost dle typu metriky – Euklidovská, Mahalanobisova. V literatuře se zmiňuje i možnost kombinace více metrik najednou a podle různě zvolených vah, které se každé metrice přiřadí, odhadnout identitu obličeje. Tento způsob ale není v implementaci zahrnut.

```
class Recognition
{
public:
```

```
        Recognition(Distance approach = EUCLIDIAN);
        ~Recognition(void);

        void recognize(std::vector<Person> &inputPeople,
        std::vector<Person> &knownPeople);
        void setDiagCovMat2(CvMat* diagCovMat2);
};
```

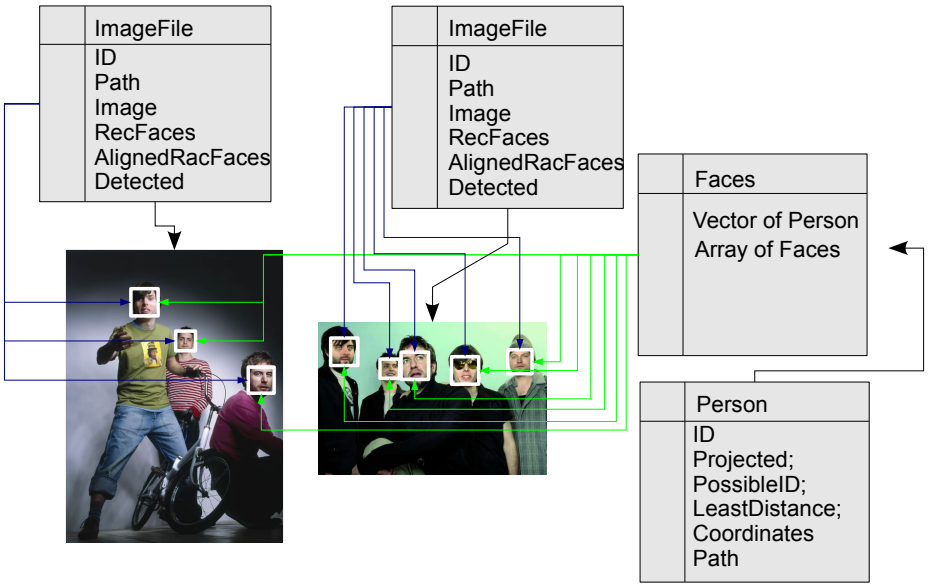
Třída `Recognition` je inicializována hodnotou definující typ vzdálenosti, kterou se měří, ta je definována ve výčtovém typu `Distance` viz dokumentace.

`void recognize(std::vector<Person> &inputPeople, std::vector<Person> &knownPeople);`

Metoda zjistí vzdálenost podle metriky, kterou byla třída inicializována. Vstupními parametry je vektor neznámých obličejů a vektor známých. Obličeje jsou zapouzdřeny ve struktuře `Person`, která mimo samotného obličeje nese informaci o jejich ID, pozici a pravděpodobném ID známého obličeje, kterému je nejvíce podobný.

Mahalanobisova vzdálenost je implementována jako normalizovaná euklidovská vzdálenost podle vzorce (3.1.16), kde singulární čísla jsou získána jako odmocniny vlastních čísel, jenž byla získána při trénování. Obdobný postup byl použit například v [24]. V tomtéž zdroji bylo rozpoznávání řešeno kombinací více metrik zároveň, což v tomto případě není implementováno.

5.2.6 Komunikace mezi fázemi výpočtu



Ilustrace 5.2.5: Význam struktury ImageFile a třídy Faces.

Mezi jednotlivými fázemi zpracování je potřeba předávat obraz s obličejem s odlišnými metadaty Ilustrace 5.1.1. K tomu slouží struktura `ImageFile` a třída `Faces`. Na začátku je potřeba načíst veškeré potřebné parametry pro detekci, zarovnání a rozpoznávání. Ty se uloží do struktury typu `Params`, jenž dále v programu distribuuje jednotlivým objektům uživatelská nastavení. O naplnění struktury `Params` se stará `FileManager`. V prvním kroku zpracování, tj načtení obrázku a detekce obličeje, je využívána struktura `ImageFile`, která nese informaci o pozici detekovaných obličejů v

obrázku, cestě k obrázku a ID obličeje. Po zarovnání detekovaného obličeje jsou do struktury ještě přidány parametry zarovnání.

Před použitím PCA algoritmu jsou data z `ImageFile` struktury transformována do třídy `Faces`. Ta obsahuje pouze zarovnané obličeje a metadata k nim. Metadata zarovnaných obličejů jsou uloženy ve struktuře Person, ta je přiřazena každému obličeji a nese informaci o ID, pozici obličeje, ID nejvíce podobného obličeje z množiny známých obličejů a cestu k obrázku. Jejich funkci ilustruje Ilustrace 5.2.5. `ImageFile` se tedy vztahuje ke každému obrázku a nese informaci o obličejích uvnitř, zatímco `Faces` je kontejner pro všechny obličeje.

5.3 Uživatelská konfigurace programu

Většina nastavení, na kterém rozpoznávání obličejů více či méně závisí je uživatelsky konfigurovatelná pomocí xml souboru. Jeho struktura odpovídá možnostem práce knihovny OpenCv s xml.

Detekci obličeje lze konfigurovat ve třech směrech. Převzorkováním obrazu podle jeho velikosti pro rychlejší prohledávání `Detection_Scale`, v případě, že se detekce provádí na obrázcích s velkým rozlišením, lze je pomocí relativního koeficientu zmenšit nebo zvýšit. Výchozí hodnota je 1.0 tudíž se pracuje nad původní velikostí obrazu. Druhým parametrem je možnost zapnout funkci, jenž vyváží histogram `Equalize_Histogram`, ta by se měla podílet na potlačení vlivu osvětlení, ale v praxi zdá se, je spíše kontraproduktivní. Nakonec ještě nechybí definovaný klasifikátor použitý pro detekci obličeje `Face_cascade`.

Zarovnání obličeje se odvíjí od typu `AlignType`, který se má provádět, ten specifikuje dva stavy, `Basic` a `EyesPos`, kde `Basic` provádí pouze empirické zarovnání, jenž je samo o sobě také konfigurovatelné, a to parametry posunutí a změna velikost `ShiftFactor` a `ScaleFactor`. Zarovnání podle markantních rysů, tedy `EyesPos` se odvíjí od dodaných parametrů pro 2D Gaussova rozložení. Samotné zarovnání pak lze měnit pomocí dvou hodnot a to vzdálenosti očí od horního okraje výřezu s obličejem `dEyes` a vzdálenosti úst od dolního okraje výřezu `dMouth`. Pro zarovnání pomocí markantních rysů je ještě třeba dodefinovat klasifikátory pro jejich detekci a velikost okna `Inner_Face_Align_Size`, do které se detekovaný obličej transformuje a ve kterém se prohledává, tím by se mělo zabránit situaci, kdy obličej bude obsahovat rysy pod rozlišovací schopnost specifických klasifikátorů Výchozí velikost okna, ve kterém se detekují markantní rysy je 200×200.

Pro výpočet PCA lze zvolit s jak velkým obličejem se má pracovat, výchozí hodnotou je 128×128. Dále pro testování lze definovat kolik se má použít vlastních vektorů ze všech možných. Toto nastavení supluje fakt, že by se vlastní vektory, jejichž přidružená vlastní čísla jsou velmi malá, neměly brát v potaz. Fáze rozpoznávání pak lze nastavit podle Euklidovské vzdálenosti, nebo Mahalanobisovy vzdálenosti. PCA taktéž závisí na nastavení vyvážení histogramu.

Zbytek parametrů se týká nastavení výpisu informací, generování obrázků mezi fázemi zpracování, ukládání Eigenfaces, průměrného obličeje atp. Definice konfiguračního souboru je na přiloženém médiu.

5.4 Shrnutí implementace

Kapitola implementace popsala způsob uvedení teorie v praxi pomocí C++ a knihovny OpenCv. Zahrnuje všechny tři základní fáze programu, detekce, zarovnání a rozpoznání. Nastíněna byla i komunikace mezi fázemi pomocí datových kontejnerů zastřešující obličeje jak z hlediska snímek → obličeje, tak i z opačného obličej → snímek. Na závěr byly popsány důležité parametry aplikace, které lze uživatelsky měnit pomocí konfiguračního souboru.

Na závěr implementace zbývá popsat práci s aplikací, zmínit její úskalí a samozřejmě klady.

Použití aplikace :

```
application configuration_file.xml [ learn | recognize ] data_file
```

Kde `application` je jméno binárního souboru – programu, `configuration_file.xml` je jméno konfiguračníhou souboru (libovolné), op definuje typ operace a tou je buď `learn` - trénování, nebo `recognize` – rozpoznávání. Poslední parametr definuje soubor, který definuje obrázky, které se mají zpracovat.

```
ID Cesta\k\obrázku
```

`ID` je celé číslo. Pozor na lomítka, zde uvedený formát platí pro os firmy Microsoft, *nix systémy používají opačná lomítka „/“.

Klady	Zápory
– Konfigurovatelnost – Relativně systémově nezávislé	– Při větším objemu obrázků – stovky, paměťová náročnost.

6 Výsledky

Výsledky popisují testování implementace na třech volně dostupných databázích obličejů. Metodika testování byla zvolena vzhledem k faktu, že není implementován práh, na jehož základě by byl vstupní obličej odmítnut jako neznámý. Tudíž aplikace vrátí na jakýkoli obličej ID nejpodobnějšího obličeje, jehož vzdálenost je minimální.

6.1 Metodika testování

Pro testovací účely byly zvoleny tři databáze obličejů.

- The BioID Face Database

  BioID obsahuje 1521 obrázků obličejů v odstínech šedi v rozlišení 384x286 pixelů. Na každém obrázku je obličej – zpříma 23 různých lidí. Některé snímky nejsou moc kvalitní, výraz obličeje je proměnlivý a na některých snímcích je část obličeje mimo záběr. [16]
- Caltech Faces

  450 barevných snímků obličejů o rozlišení 896x592 pixelů. JPEG formát, obsahuje 27 různých lidí s různým osvětlením, výrazem i pozadím. Obličeje jsou zpříma. [25]
- IMM Face Database

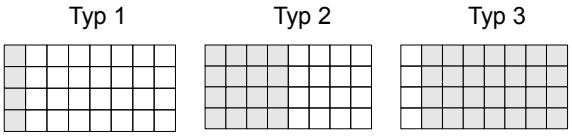
  Obsahuje 240 anotovaných snímků obličejů. Jedná se o 40 různých lidí při různém osvětlení, s emotivním výrazem – úsměv, zpříma a natočením do stran. [22]

Nad těmito daty bylo provedeno celkově 36 různých měření. Prvním krokem bylo rozdělení databáze na známé a neznámé obličeje. První dvě uvedené databáze The BioID Face Database a Caltech Faces obsahují více než tři fotky jedné osoby s obličejem zpříma, zatímco IMM Face Database obsahuje právě tři fotky daného obličeje zpříma, zbytek je různě ze stran – vzhledem k premisám aplikace nebyly použity. Rozdělení definuje Tabulka 6.1 a ilustruje Ilustrace 6.1.1.

Typ 1	Jeden obrázek pro právě jeden obličej – trénovací sada, zbytek neznámé obličeje – obličeje lidí z trénovací sady.
Typ 2	Polovina obrázků ze všech, které se vztahují k právě jedné identitě – trénovací sada, druhá polovina neznámé obličeje.
Typ 3	Jeden obrázek pro právě jeden obličej – neznámé obličeje, zbytek trénovací sada.

	Typ 1	Typ 2	Typ 3
BioID	ano	ano	ano
Caltech	ano	ano	ano
IMM	ano	ne	ano

Tabulka 6.1: Rozdělení databáze obličejů.



Ilustrace 6.1.1: Šedá část značí trénovací sadu.

Právě z důvodu „pouze“ tří snímků jednoho obličeje nebyl typ 2 rozdělení aplikován na IMM Face Database.
Rozpoznávání bylo provedeno kombinací různých parametrů, které lze nastavit pomocí uživatelské konfigurace a týkalo se především použití nebo nepoužití vyvážení histogramu, různého druhu zarovnání obličeje a nastavením typy metriky.

- Bez zarovnání s použitím i bez zpracování histogramu s Euklidovskou i Mahalanobisovou vzdáleností.
- Jednoduché zarovnání s použitím i bez zpracování histogramu s Euklidovskou i Mahalanobisovou vzdáleností.
- Zarovnání pomocí očí s použitím i bez zpracování histogramu s Euklidovskou i Mahalanobisovou vzdáleností.

Celkem tedy bylo provedeno 6 testů rozpoznávání na dvou databázích rozdělených do tří typů a jedné databázi rozdělené do dvou typů, celkově tedy $6 \times (2 \times 3 + 2) = 42$.

6.2 Výsledky testování

Výsledky testování shrnují tabulky v příloze. Zde je uvedeno několik z nich, první se týká provedení testu na obličejové databázi Caltech Faces.

Databáze : Caltech Typ dat : Typ 2 Zpracování Histogramu : NE

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		132	2	71	205
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		140	2	96	238
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	123	17	0,88	
	Mahalanobis.	132	8	0,94	

Tabulka 6.2: Výsledek testu na databázi Caltech, typ rozdělení : Typ 2.

Jedná se o výsledek testu s rozdělením trénovací a rozpoznávací sady podle typu 2, tedy cca polovina obličejů databáze je použitá pro trénování a polovina se poté rozpoznává. Polovina je volena tak, aby byly v obou polovinách pokud možno obličeje jedné osoby ve stejném počtu.

Dle tabulky nebylo použití zpracování histogramu pro detekci a následnou PCA analýzu. Tabulka se dělí do dvou hlavních částí, Zarovnání podle očí, která definuje celkový počet vstupních obličejů, odmítnuté obličeje jsou ty, u kterých se při zarovnáváni nenalezli všechny tři rysy – oči, nos a koutky úst. Často se jedná o chybnou detekci, nicméně v tomto případě bude zřejmě většina z nich odmítnutá chybně, na druhou stranu to indikuje, že zřejmě nesplňují stanovené podmínky. Nicméně nepopiratelné je, že je asi vyloučeno více obličejů, než je třeba. Sloupce „Nedetekovaných“ udává počet trénovacích obrázků, na kterých nebyl detekován obličej (jedná se o chybné rozhodnutí

detektoru, neboť každý snímek z databáze obsahuje obličej, nicméně se jej nepodařilo detekovat). Sloupce zpracovaných pak udává počet obličejů, které byly použity pro trénování. Tato část tabulky se tedy výlučně týká fáze před rozpoznáváním a zahrnuje detekci i zarovnání na trénovací množině dat.

Druhá půlka tabulky „Fáze rozpoznávání“ obsahuje v řádku „Výsledky detekce rozpoznávání“ stejné údaje jako předešlá část, ale ta se týká rozpoznávané množiny dat. Podstatné jsou pak sloupce „Rozpoznaných“ a „Rozpoznaných / Zpracovaných“, kde první udává počet rozpoznaných obličejů podle dané metriky a druhý je relativní hodnotu vzhledem ke všem obličejům, které prošli procesem rozpoznávání.

Databáze : IMM Typ dat : Typ 1 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		40	0	0	40
Fáze rozpoznávání					
		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
Výsledky detekce rozpoznávání		80	0	0	80
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	63	17	0,79	
	Mahalanobis.	44	36	0,55	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	40	40	0,5	
	Mahalanobis.	50	30	0,63	

Tabulka 6.3: Rozpoznávání na datech Caltech, typ rozdělení 1.

Další ilustrační tabulka shrnuje rozpoznávání na databázi IMM s rozdělením do typu 1, tj. v trénovací množině je právě jeden obličej od jednotlivých osob a zbytek se rozpoznává.

Tabulka udává hodnoty míry rozpoznání při použití jednoduchého a žádného zarovnání detekčního okna. Výsledky 0,5 – 0,79 jsou velmi dobré možná až podezřelé, o to více je zarážející tak vysoká hodnota úspěšného rozpoznávání bez zarovnání. S podobnými hodnotami se lze setkat ve více testech, v tomto případě lze tento výsledek podpořit argumentem, že snímky s obličeji této sady mají stejné – tmavě zelené pozadí, tudíž šum z pozadí je u všech stejný – vyruší se a na rozpoznání nemá vliv.

Testování potvrdilo domněnku, že rozpoznání je obecně úspěšnější, obsahuje-li známá množina obličejů, na které se trénuje, více obličejů identické osoby, to plyne z výsledků, získaných testováním s rozdělením databází obličejů do typu 3 podle Tabulka 6.1: Rozdělení databáze obličejů.. Naopak rozdělení do typu 1 vedlo s výjimkami k menšímu úspěchu rozpoznávání.

Docela překvapujícím se ukázal minimální rozdíl mezi použitím Euklidovské a Mahalanobisovy metriky, nicméně pro tento závěr by zřejmě bylo nutné provést více testů.

V výsledcích se objevilo několik anomálií, kdy euklidovská vzdálenost udává mnohem lepší míru rozpoznání, než Mahalanabisova, konkrétně jde například o měření na databázi BioID, typ 3 rozdělení do trénovací a rozpoznávací množiny při použití jednoduchého a žádného zarovnání. Bohužel tyto výsledky nejsem schopen vysvětlit.

Testování nezahrnuje experimentování s počtem vlastních vektorů, který lze uživatelsky měnit, nicméně lze soudit, že s nižším počtem vlastních vektorů pravděpodobnost úspěšného rozpoznávání bude klesat, neboť během experimentování to několik zde nepublikovaných testů naznačovalo, tento závěr vychází z faktu, že většina vlastních čísel získaná z kovarianční matice je velmi velká a v některých případech neklesne pod čtyřciferné celé číslo. Tyto údaje lze zjistit z vygenerovaného souboru parametrů při trénovaní.

Výsledky s použitím vyvážení histogramu se mírně liší od výsledků bez jeho použití, Jeho využití je ale sporné, statisticky počet měření nestačí pro učinění závěru v tomto směru.

Praktické využití implementace asi není zrovna vhodné, neboť odchylky v různých výsledcích jsou docela velké, nicméně mnohé jde na vrub naladění systémy pro danou databázi, zde čistě náhodně byla implementace překvapivě velmi úspěšná na trénovací sadě Caltech.

V konečném shrnutí samotné implementace a výsledků, které byly získány je nasnadě konstatovat, že na konci práce se objevilo více otázek k zamyšlení, než bylo na jejím začátku, což na základě mnoha příkladů z historie je přeci tak typické.

Ilustrace 6.2.1: Ukázky obrázků z obličejových databází. Zleva : Caltech, BioID, IMM

7 Závěr

Celou práci lze z hlediska struktury textu rozdělit do dvou hlavních částí. V první části byly zmíněny algoritmy, které se používají v počítačovém vidění. Na jejichž základě stála celá tato práce. Samozřejmě se jedná o pouhý vzorek známých algoritmů, nicméně byly zmíněny právě ty, které jsou v současnosti hojně využívané jak v oblasti detekce, tak i rozpoznávání. Oba tyto směry počítačového vidění zastřešuje obecně klasifikace, která je jedním z pilířů analytického i sémantického zpracování obrazu. Druhou část tvoří popis implementace, zhodnocení a popsání výsledků, kterých bylo dosaženo na různých obličejových datech.

Pochopení a poznání uvedených algoritmů je základní premisou k implementaci systému pro rozpoznávání tváří ve fotografii a obecně lze říci v jakémkoli obrazovém médiu. Celý proces rozpoznávání lidí ve fotografii byl v práci rozdělen do tří fází, kde ke každé fázi bylo vybráno a představeno několik algoritmů, implementace pak zahrnuje právě jeden z nich.

První fází je tedy detekce, kromě AdaBoost algoritmu pro trénování klasifikátorů byly zmíněny i Neuronové sítě. Vzhledem k implementaci za použití knihovny OpenCv pro počítačové vidění, byl vybrán klasifikátor k detekci obličejů získaný metodou AdaBoost. Druhou fází je zarovnání detekovaného obličeje. Zmíněno bylo zarovnání na základě markantních rysů, kterými jsou například oči, nos a ústa, ale také sofistikovanější algoritmus ASM, který pracuje s konturou obličeje. Výsledný program používá zarovnání na základě markantních rysů, pro jejichž použití bylo nutné vypočítat parametry Gaussova rozložení. Nabízí i jednodušší způsoby vycházející z vlastností použité detekce. Poslední fází je rozpoznávání. Podrobně byl popsán systém použití Eigenfaces, zmíněn byl i algoritmus AAM, který svou komplexností zahrnuje samotné Eigenfaces. Implementace obsahuje metodu Eigenfaces.

Výsledky, jenž celý systém rozpoznávání lidí podle obličeje ve fotografii nabízí, jsou vzhledem ke zvoleným databázím obličejů dobré, někdy až překvapující, na druhou stranu při určitých datech může rozpoznání selhat. Na konci práce tak vyvstává mnohem více otázek než na jejím začátku. Možností dalšího vývoje jsou veliké, ať už se jedná o pouhé ladění systému k dosažení lepších výsledků nebo k implementaci dalších v práci zmíněných algoritmů. Limity rozpoznávání obličejů v dvourozměrném obraze jsou dnes ale již známé, proto zřejmě hlavní vývoj vede do prostoru o dimenzi vyššího, do 3D modelování obličejů a jejich různé transformace.

Seznam použité literatury

[1] Oliva, A., Torralba, A., Castelhano, M. S., Henderson, J.M. : *Top - down control of visual attention in object detection*, Barcelona, Spain, 2003, url : http://web.mit.edu/torralba/www/ICIP03.pdf (22. 4. 2009)

[2] Stegmann, M. B. : *Active Appearance Models: Theory, Extensions and Cases*, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2000 url : http://www.imm.dtu.dk/~aam/main/ (22. 4. 2009)

[3] Cootes , T.F., Taylor, C.J. : *Statistical Models of Appearance for Computer Vision*, Imaging Science and Biomedical Engineering University of Manchester, 2004, url : http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/Papers/asm_overview.pdf (22. 4. 2009)

[4] Fawcett, T. : *ROC Graphs: Notes and Practical Considerations for Researchers*, Palo Alto, HP Laboratories, 2004, url : http://home.comcast.net/~tom.fawcett/public_html/papers/ROC101.pdf (22. 4. 2009)

[5] Škutová, J. : *Neuronové sítě v řízení systémů*, Ostrava, Ostravská univerzita, 2004, url : http://www.fs.vsb.cz/books/NeuronoveSite/ (22. 4. 2009)

[6] Šnorek, M., Jiřina, M. : *Neuronové sítě a neuropočítače*, Praha, České vysoké učení technické ČVUT, 1996

[7] Rowley, H. A., Baluja, S., Kanade, T. : *Neural Network-Based Face Detection*, San Francisco, 1996, url : http://www.informedia.cs.cmu.edu/documents/rowley-ieee.pdf (22.4.2009)

[8] Garcia, Ch., Delakis, M. : *Convolutional Face Finder:A Neural Architecture for Fastand R obust Face Detection*, 2004

[9] Velínský, T. : *Neuronová síť Neocognitron,* Praha, 2000, url : http://neuron.felk.cvut.cz/courseware/data/chapter/velinsky2000/ (23.4. 2009)

[10] Freund, Y., Schapire, R.E. : *A desicion-theoretic generalization of on-line learning and an application to boosting*, Springer, 1995, url : http://www.face-rec.org/algorithms/Boosting-Ensemble/decision-t heoretic_generalization.pdf (25.4.2009)

[11] Šochman, J. : *Cvičení z RPZ – AdaBoost*, České vysoké učení technické ČVUT, 2005 url : http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/adaboost/adaboost.pdf (26.4. 2009)

[12] Freund, Y., Schapire, R.E. : *A Short Introduction to Boosting*, Journal of Japanese Society for Artificial Intelligence, 1999, url : http://www.cs.princeton.edu/~schapire/uncompress-papers.cgi/FreundSc99.ps (26.4.2009)

[13] Šochman, J. Matas, J. : *AdaBoost*, Praha, České vysoké učení technické ČVUT, 2008, url : http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf (26.4.2009)

[14] Viola, P., Jones, J. M. : *Robust Real-Time Face Detection*, Springer, 2004, url : http://www.springerlink.com/content/q70v4h6715v5p152/fulltext.pdf (28.4.2009)

[15] Lienhart, R., Maydt, J. : *An Extended Set of Haar-like Features for Rapid Object Detection*, Santa Clara, Intel Labs, 2002, url : http://www.lienhart.de/ICIP2002.pdf (2.5.2009)

[16] BioID dataset, Databáze obličejů, HumanScan AG, Switzerland, url : http://www.bioid.com/downloads/facedb/index.php (20.5.2009)

[17] Turk, M., Pentland, A. : *Eigenfaces for Recognition*, Journal of Cognitive Neurosicence, 1991, url : http://www.face-rec.org/algorithms/PCA/jcn.pdf (8.5.2009)

[18] Cootes, T.F., Edwards, G.J., Taylor, C.J. : *Face Recognition Using Active Appearance Models*, Springer, 1998, url : http://www.cc.gatech.edu/classes/AY2002/cs7635_spring/Private/edwards98face.pdf

(8.5.2009)
[19] Strang, G. : *Introduction to Linear Algebra*, Wellesley, Wellesley-Cambridge Press, 2009
[20] Lienhart, R. : *Dimensionality reduction techniques*, Augsburg, Germany, University Augsburg 2009, url : http://math.mit.edu/linearalgebra/ (13.5.2009)
[21] Šrecha, J. : *Optimální rozhodování a řízení*, Praha, České Vysoké Učení Technické – ČVUT, 2000, url : http://dce.felk.cvut.cz/stecha/0rr_s.pdf (11.5.2009)
[22] Stegmann, M B., Nordstrom, M M., Larsen, M., Sierakowski, J. : *The IMM Face DatabaseAn Annotated Dataset of 240 Face Images*, Technical University of Denmark, 2004, url : http://www2.imm.dtu.dk/~aam/ (13.5.2009)
[23] Šašinka, O. : *Detekce obličeje*, Brno, FIT VUT v Brně, 2009, url : http://www.fit.vutbr.cz/study/DP/DP.php?id=6479&y=2008 (20.5.2009)
[24] Wendy, S., Yambor ,B. A., Draper, J., Beveridge, R. : *Analyzing PCA-based Face Recognition Algorithms: EigenvectorSelection and Distance Measures*, Fort Collins, Colorado State University, 2000, url : http://www.cs.colostate.edu/evalfacerec/papers/eemcvcsu.pdf (22.5.2009)
[25] Weber, M. : *Caltech frontal face dataset*, Pasadena, California Institute of Technology, 1999, url : http://www.vision.caltech.edu/Image_Datasets/faces/faces.tar (22.5.2009)

Seznam ilustrací

Seznam tabulek

Seznam příloh

Příloha 1. Tabulky vyhodnocení implementace.
Příloha 2. CD se zdrojovými kódy a programovou dokumentací.

Databáze : IMM Typ dat : Typ 1 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		40	0	0	40
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		80	0	0	80
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	63	17	0,79	
	Mahalanobis.	44	36	0,55	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	40	40	0,50	
	Mahalanobis.	50	30	0,63	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		37	0	3	40
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		55	0	25	80
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	27	28	0,49	
	Mahalanobis.	33	22	0,60	

Databáze : IMM Typ dat : Typ 1 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		40	0	0	40
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		82	0	0	82
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	66	16	0,80	
	Mahalanobis.	59	23	0,72	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	42	40	0,51	
	Mahalanobis.	52	30	0,63	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		35	0	5	40
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		65	0	17	82
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	34	65	0,52	
	Mahalanobis.	34	65	0,52	

Databáze : IMM Typ dat : Typ 2 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		80	0	0	80
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		40	0	0	40
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	37	3	0,93	
	Mahalanobis.	26	14	0,65	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	19	21	0,48	
	Mahalanobis.	35	5	0,88	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		55	0	25	80
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		37	0	3	40
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	16	21	0,43	
	Mahalanobis.	26	11	0,70	

Databáze : IMM Typ dat : Typ 2 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		82	0	0	82
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		40	0	0	40
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	35	5	0,88	
	Mahalanobis.	32	8	0,80	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	24	16	0,60	
	Mahalanobis.	34	6	0,85	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		65	0	17	82
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		35	0	5	40
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	16	65	0,46	
	Mahalanobis.	26	65	0,74	

Databáze : BioID Typ dat : Typ 1 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		22	1	0	23
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1369	20	0	1389
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	419	950	0,31	
	Mahalanobis.	359	1010	0,26	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	313	1056	0,23	
	Mahalanobis.	529	840	0,39	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		20	1	2	23
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1284	20	85	1389
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	314	970	0,24	
	Mahalanobis.	520	764	0,4	

Databáze : BioID Typ dat : Typ 1 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		22	1	0	23
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1372	21	0	1393
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	516	856	0,38	
	Mahalanobis.	423	949	0,31	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	596	776	0,43	
	Mahalanobis.	643	729	0,47	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		15	1	7	23
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1050	21	322	1393
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	423	627	0,4	
	Mahalanobis.	451	599	0,43	

Databáze : BioID Typ dat : Typ 2 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		688	11	0	699
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		703	10	0	713
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	403	300	0,57	
	Mahalanobis.	207	496	0,29	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	404	299	0,57	
	Mahalanobis.	445	258	0,63	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		662	11	26	699
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		642	10	61	713
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	396	246	0,62	
	Mahalanobis.	467	175	0,73	

Databáze : BioID Typ dat : Typ 2 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		690	9	0	699
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		704	13	0	717
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	357	347	0,51	
	Mahalanobis.	37	667	0,05	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	456	248	0,65	
	Mahalanobis.	402	302	0,57	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		515	9	175	699
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		550	13	154	717
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	367	183	0,67	
	Mahalanobis.	365	185	0,66	

Databáze : BioID Typ dat : Typ 3 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1369	20	0	1389
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		22	1	0	23
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	20	2	0,91	
	Mahalanobis.	9	13	0,41	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	21	1	0,95	
	Mahalanobis.	6	16	0,27	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1284	20	85	1389
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		20	1	2	23
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	18	2	0,9	
	Mahalanobis.	5	15	0,25	

Databáze : BioID Typ dat : Typ 3 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1372	21	0	1393
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		22	1	0	23
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	20	2	0,91	
	Mahalanobis.	4	18	0,18	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	22	0	1	
	Mahalanobis.	2	20	0,09	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		1050	21	322	1393
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		15	1	7	23
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	15	0	1	
	Mahalanobis.	8	7	0,53	

Databáze : Caltech Typ dat : Typ 1 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		21	0	0	21
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		416	4	0	420
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	166	250	0,40	
	Mahalanobis.	137	279	0,33	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	196	220	0,47	
	Mahalanobis.	248	168	0,60	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		17	0	4	21
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		254	4	162	420
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	156	98	0,61	
	Mahalanobis.	179	75	0,70	

Databáze : Caltech Typ dat : Typ 1 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		22	0	0	22
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		414	3	0	417
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	165	249	0,40	
	Mahalanobis.	166	248	0,40	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	246	168	0,59	
	Mahalanobis.	271	143	0,65	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		14	0	8	22
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		222	3	192	417
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	133	89	0,60	
	Mahalanobis.	141	81	0,64	

Databáze : Caltech Typ dat : Typ 2 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		203	2	0	205
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		236	2	0	238
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	155	81	0,66	
	Mahalanobis.	125	111	0,53	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	174	62	0,74	
	Mahalanobis.	173	63	0,73	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		132	2	71	205
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		140	2	96	238
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	123	17	0,88	
	Mahalanobis.	132	8	0,94	

Databáze : Caltech Typ dat : Typ 2 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		197	2	0	199
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		241	1	0	242
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	174	67	0,72	
	Mahalanobis.	136	105	0,56	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	201	40	0,83	
	Mahalanobis.	196	45	0,81	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		114	2	83	199
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		123	1	118	242
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	112	11	0,91	
	Mahalanobis.	115	8	0,93	

Databáze : Caltech Typ dat : Typ 3 Zpracování Histogramu : NE

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		416	4	0	420
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		21	0	0	21
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	17	4	0,81	
	Mahalanobis.	7	14	0,33	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	17	4	0,81	
	Mahalanobis.	19	2	0,90	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		254	4	162	420
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		17	0	4	21
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	15	2	0,88	
	Mahalanobis.	15	2	0,88	

Databáze : Caltech Typ dat : Typ 3 Zpracování Histogramu : ANO

Žádné a jednoduché zarovnání					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		414	3	0	417
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		22	0	0	22
Bez zarovnání		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	18	4	0,82	
	Mahalanobis.	14	8	0,64	
Jednoduché zarovnání					
Vzdálenost	Euklidovská	20	2	0,91	
	Mahalanobis.	19	3	0,86	

Zarovnání podle očí					
		Fáze učení			
Výsledky detekce trénování		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		222	3	192	417
Fáze rozpoznávání					
Výsledky detekce rozpoznávání		Zpracovaných	Nedetekovaných	Odmítnutých	Z celkového počtu
		14	0	8	22
Zarovnání podle očí		Rozpoznaných	Nerozpoznaných	Rozpoznaných / Zpracovaných	
Vzdálenost	Euklidovská	12	2	0,86	
	Mahalanobis.	12	2	0,86	