METODA PCA A JEJÍ IMPLEMENTACE V JAZYCE C++

Lukáš Fritsch, Ing.

ČVUT v Praze, Fakulta elektrotechnická, Katedra radioelektroniky

Abstrakt

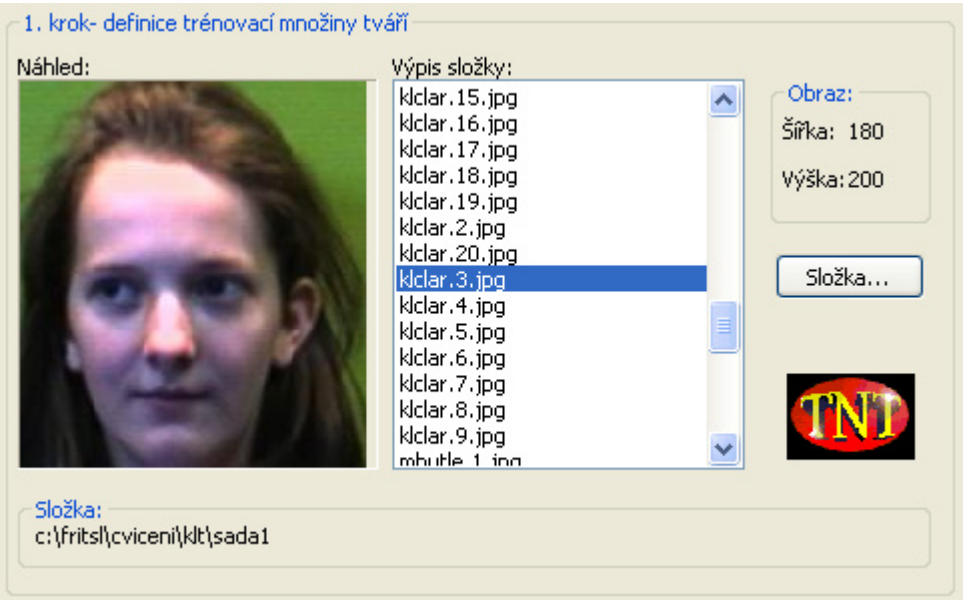
Metoda PCA (Principal Component Analysis- analýza hlavních komponent) může být využita k reprezentaci např. lidské tváře v tzv. facespace (prostoru tváří), v němž lze efektivně provádět rozpoznávání tváře. Na základě dále uvedeného popisu metody PCA byla pod operačním systémem Windows® v jazyce Visual C++ naprogramována aplikace, která demonstruje užití metody v úloze rozpoznávání a též v úloze komprese obrazové informace. Při návrhu aplikace byl kladen důraz na rychlost výpočtu a na možnost použití rozsáhlejších trénovacích sad snímků. Součástí realizace bylo ověření funkčnosti algoritmu pro obě zmiňované úlohy v programovém prostředí MATLAB.

1 Základní pojmy a úvod do problematiky

Uvažujme kolekci o počtu *P* obrázků tváří (též trénovací sadu), kdy všechny mají stejné rozměry *WxH*. Z těchto obrázků sestrojíme matici o počtu řádků rovném počtu obrazů *P* v kolekci a o počtu sloupců rovném číslu *n=WxH*. Obrázky tváří jsou si podobné, ať už ve více či méně rysech, a proto nejsou náhodně distribuovány (existuje mezi nimi nenulová korelace). Hlavní myšlenkou PCA pro námi zamýšlené účely je nalézt vektory, které nejlépe popisují distribuci obrázků tváří v celém prostoru obrázků. Tyto nalezené vektory dimenze *n* definují podprostor, který nazýváme prostor obrázků tváří (eigenspace, též facespace), jsou lineární kombinací původních vektorů a tvoří bázi eigenspace. Tyto vektory jsou vlastními vektory kovarianční matice korespondující k původním obrázkům a protože jsou podobné obrázkům tváří, nazýváme je eigenfaces. Tím jsme si na úvod uvedli základní termíny a principy, [1], [2], [4], [5], [6].

2 Proces rozpoznávání

Zabývejme se postupem, který nás dovede až do fáze rozpoznávání tváře, [1]. Nechť máme kolekci obrázků tváří $x_1, x_2, \ldots, x_P$, přičemž si představujme každý obrázek jako vektor (jednotlivé řádky obrazové matice poskládáme postupně za sebou). Na obr. 1 je uvedena část dialogového okna aplikace, kde se definuje trénovací sada. V tomto konkrétním případě sada obsahuje celkem 80 obrázků od čtyř různých osob (1 muž a 3 ženy) a lze ji získat na stránce [7].



Obr. 1 Volba trénovací sady

Dalším krokem je výpočet průměrného obrázku této sady podle vztahu:

$$M = \frac{1}{P} \cdot \sum_{i=1}^{P} x_i \ . \qquad (1)$$

Od každého obrázku sady odečteme průměrný obraz (provedeme centrování dat):

$$\overline{x}_i = x_i - M. \qquad (2)$$

Takto získané vektory uspořádejme do matice, označme ji $\overline{X}$, kdy každý řádek matice popisuje jeden z vektorů $\overline{x}_i$. Matice má tedy *P* řádků a *n* sloupců a je vstupem do analýzy hlavních komponent, která hledá množinu vektorů, které nejlépe popisují distribuci vstupních dat. Hledané vektory jsou vlastními vektory (hlavními komponentami) kovarianční matice C', kterou získáme následovně:

$$C' = \overline{X} \cdot \overline{X}^T. \qquad (3)$$

Vidíme, že kovarianční matice získaná podle vztahu (3) má *P* řádků a *P* sloupců. Může mít ovšem i rozměry *n x n*, pokud provedeme transpozici 1. matice v součinu (3):

$$C = \overline{X}^T \cdot \overline{X}. \qquad (4)$$

Zřejmě je kovarianční matice získaná podle vztahu (3) z hlediska výpočetní náročnosti vlastních čísel a vektorů výhodnější. Předpokladem ovšem musí být, že $P << n$. Tedy, že počet obrázků v kolekci je daleko menší než počet obrazových bodů jednoho z obrázků. V opačném případě bychom museli zmenšit rozměry obrázků v sadě, což je docela schůdné řešení nebo použít jiný přístup. K problematice výpočetní náročnosti, především pak z paměťového hlediska, se ještě vrátíme na konci příspěvku.

Pro rozpoznávání ovšem potřebujeme mít vlastní vektory matice *C*. Jak ale uvidíme dále, lze k výpočtu vlastních vektorů matice *C* použít vlastní vektory matice C'. Máme tedy kovarianční matici C' získanou podle (3). Tato matice je zřejmě reálná a symetrická. V dalším kroku musíme numericky vyřešit rovnici:

$$C' \cdot V' = \lambda \cdot V', \qquad (5)$$

tj. nalézt vlastní vektory, které jsou uspořádány po řádcích v matici V' a vlastní čísla λ kovarianční matice C'. Tento krok není jednoduchý, k jeho vyřešení v aplikaci používáme hlavičkový soubor *jama_eig.h* z balíku JAMAC++, který lze zdarma získat na stránce [3] . Implementační část je součástí uvedeného hlavičkového souboru. Je psána moderními programovacími technikami- pomocí šablon objektových typů.

Hlavičkový soubor *jama_eig.h* vyžaduje ještě soubory *tnt_xxx.h* obsahující implementace pro správu polí. Soubory jsou distribuovány s balíkem JAMAC++.

Po vyřešení rovnice (5) získáme vlastní čísla a vektory matice C'. Kovarianční matice C' obsahuje nejvýše *P* vlastních čísel různých od nuly a platí, že tato čísla jsou shodná s vlastními čísly matice *C*. U vlastních vektorů toto neplatí. Zřejmě matice V' obsahující v řádcích vlastní vektory má stejné rozměry jako kovarianční matice C', tedy *P x P*. K vlastním vektorům uspořádaných po řádcích v matici *V* kovarianční matice *C* se lze dostat vztahem:

$$V = \overline{X} \cdot V'. \qquad (6)$$

Dostaneme tak matici *V* obsahující v *P* řádcích vlastní vektory délky *n*, které tvoří bázi prostoru eigenspace. Vektory nyní seřadíme podle příslušných vlastních čísel od největšího k nejmenšímu pomocí vhodného řadicího algoritmu. Před započetím řazení si ještě vytvoříme pomocné hodnotové pole, které naplníme vzestupnou řadou čísel od 0 do *P-1*. Spolu s prohazováním vlastních čísel při řazení prohazujeme i obsah tohoto hodnotového pole. Zřejmě po ukončení řazení obsahuje hodnotové pole odkazy do původního nesetříděného pole. Zde používáme řazení výběrem (select sort- z posloupnosti vlastních čísel postupně vybíráme minimální prvek a ten vložíme za již seřazený úsek). Lze sloučit výpočet matice *V* s řazením vlastních vektorů do jednoho kroku- při výpočtu (6) umístíme do *i*-tého řádku matice *V* vektor získaný výpočtem vektoru z matice V' na řádku rovném hodnotě umístěné na pozici *(P-1-(i-1))* výše uvedeného pomocného hodnotového pole. Uveďme příklad. Máme posloupnost čísel *h = {2, 5, 1, 4}*. Vytvoříme pomocné hodnotové pole *p = [0, 1, 2, 3]*. Posloupnost *h* seřadíme, dostaneme *h = {1, 2, 4, 5}*, prvky pole *p* se prohodí, čili *p = [2, 0, 3, 1]*. Dále máme matici $V' = [v'_2, v'_5, v'_1, v'_4]^T$. Do 1. řádku matice *V* umístíme vektor získaný výpočtem $\overline{x} \cdot v'_5$, neboť poslední prvek (postupujeme od největšího vlastního čísla k nejmenšímu) pole *p* má hodnotu 1- tj. odkazuje na

druhý řádek v matici V'. Do 2. řádku matice *V* pak umístíme vektor získaný výpočtem $\bar{x} \cdot v'_4$, neboť předposlední prvek pole *p* má hodnotu 3 a tedy odkazuje na čtvrtý řádek matice V', atd. pro další vektory.
Po tomto kroku máme sestupně seřazené vlastní vektory matice *V*. Dále je musíme normalizovat na jednotkovou velikost. Pro každý vektor v_i z matice V provedeme přepočet podle vztahu:

$$v_i = \frac{v_i}{\|v_i\|}. \tag{7}$$

Pro normu vektoru obecně platí:

$$\|v_i\| = \sqrt{\sum v_i \cdot v_i^*}\ , \tag{8}$$

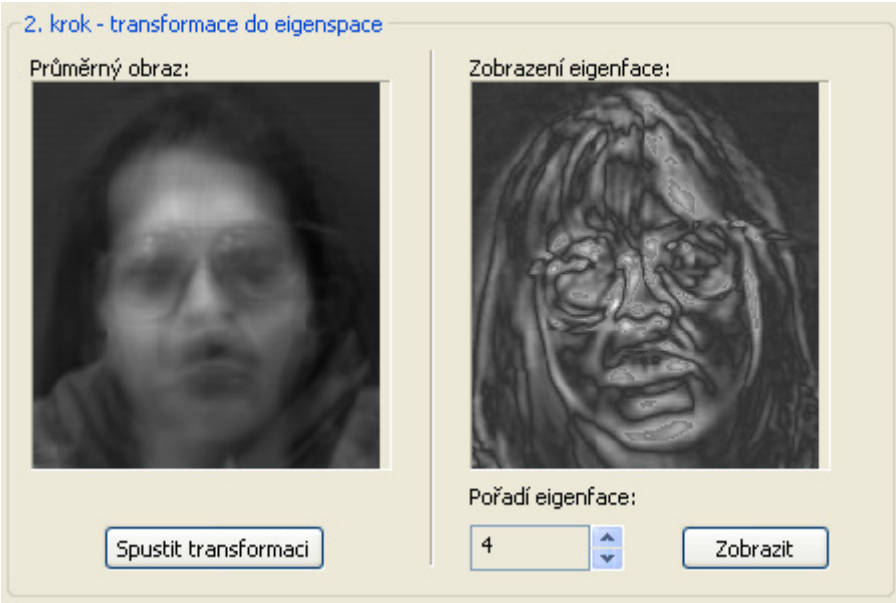
v našem případě jsou hodnoty vlastních vektorů reálné, proto:

$$\|v_i\| = \sqrt{\sum v_i^2}\ . \tag{9}$$

Po tomto kroku máme určenou bázi prostoru eigenspace, která je tvořena maticí *V*, ve tvaru potřebném pro další kroky. Nyní můžeme do eigenspace transformovat trénovací obrázky umístěné v matici $\bar{X}$. Předpokládejme, že jednotlivé transformované eigenfaces vkládáme po řádcích do matice $\tilde{X}$. Transformaci provedeme podle vztahu:

$$\tilde{x}_i = x_i \cdot V^T. \tag{10}$$

Na obr. 2 je uvedena část dialogového okna aplikace, kde je zobrazen průměrný obrázek a též zvolený eigenface. Vizualizace eigenface je provedena tak, že jsou jeho prvky lineárně přeškálovány do rozsahu hodnot 0 až 255.



Obr. 2 Průměrný obrázek a zobrazení zvoleného eigenface

Po tomto kroku jsme připraveni zahájit proces rozpoznávání obrázku neznámé tváře; označme ho *u*. Tento obrázek musíme pomocí výše získané báze převést do eigenspace, tedy:

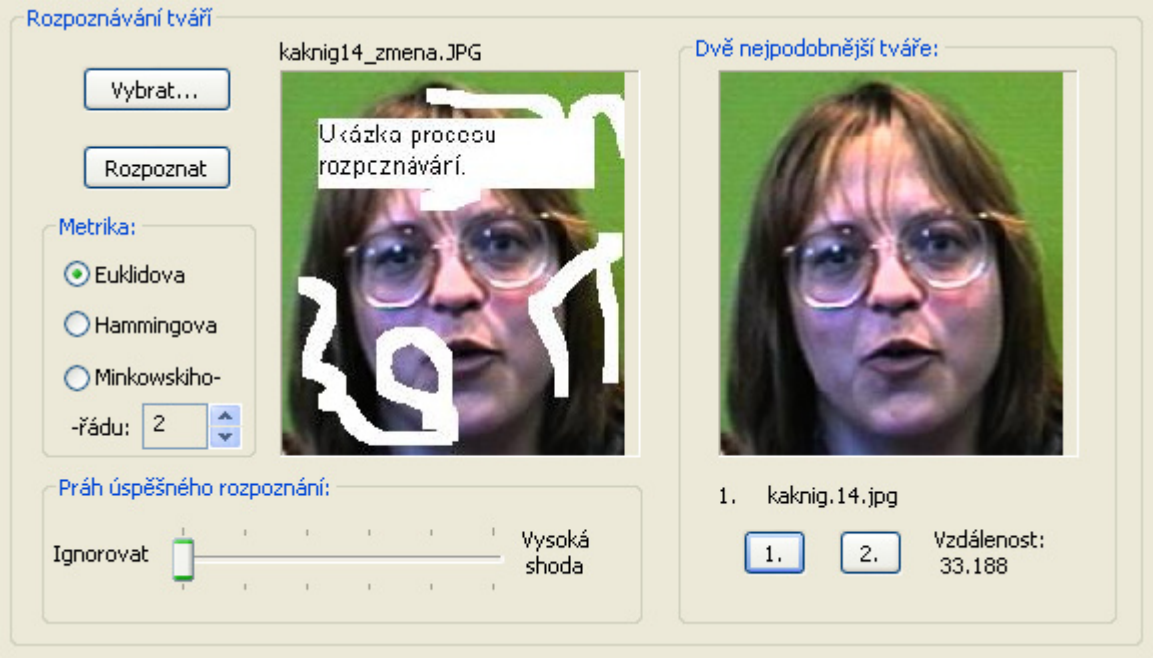
- obrázek vycentrujeme: $\bar{u} = u - M$,
- vypočteme eigenface: $\tilde{u} = \bar{u} \cdot V^T$.

Zbývá rozhodnout o nejpodobnější tváři z trénovací sady k tváři neznámé. Provedeme to tak, že určíme vzdálenosti trénovacích eigenfaces $\tilde{x}_i$ k eigenface $\tilde{u}$. Z nalezené minimální vzdálenosti nakonec určíme odpovídající tvář z trénovací sady. K určení vzdálenosti můžeme použít různé metriky. V této práci jsou na výběr metriky:

- Euklidova, $d_E(\tilde{u}, \tilde{x}_i) = \sqrt{\sum_{j=0}^{P-1} (\tilde{u}[j] - \tilde{x}_i[j])^2}$ (11)

- Hammingova, $d_H(\tilde{u},\tilde{x}_i)=\sqrt{\sum_{j=0}^{P-1}|\tilde{u}[j]-\tilde{x}_i[j]|}$ (12)
- Zobecněná Minkowskiho, $d_M(\tilde{u},\tilde{x}_i)=\sqrt[n]{\sum_{j=0}^{P-1}(\tilde{u}[j]-\tilde{x}_i[j])^n}$. (13)

Tím máme popsán postup při úloze rozpoznávání. Na obr. 3 je uvedena část dialogového okna aplikace, kde probíhá proces rozpoznávání. Vstupem je obrázek, jehož nejpodobnější vzor chceme najít v trénovací sadě a výstupem pak nalezené dva nejpodobnější vzory v sadě.



Obr. 3 Proces rozpoznávání tváří- levý obrázek je vstup, pravý pak nalezený nejpodobnější obrázek.

V rámci procesu rozpoznávání lze též nastavit práh úspěšného rozpoznání, který je chápán jako mezní hodnota vzdálenosti obrázku pro rozpoznání od obrázku z trénovací sady. Je-li vzdálenost větší než nastavený práh, vypíše se informace, že nelze nalézt nejpodobnější obrázek. Práh může být ignorován.

3 Proces zpětné rekonstrukce

Vedle rozpoznávání je v této práci naprogramována ještě zpětná rekonstrukce z eigenspace, přičemž lze si zvolit počet vektorů báze, které mají být k rekonstrukci použity, [1]. Toto je v podstatě způsob, který se dá využít ke kompresi snímků. Popišme si proto, co tato úloha vyžaduje. K dispozici máme:

- bázi eigenspace *V* o rozměrech *P x n*
- transformovaný obrázek ve formě eigenface $\tilde{u}$ resp. $\tilde{x}_i$.

Předpokládejme, že chceme zrekonstruovat *m-tý* obrázek z trénovací sady s použitím *k* vektorů báze. Pak lze rekonstrukci vyjádřit vztahem:

$$\bar{\bar{x}}_m[t]=\sum_{j=0}^{k-1}\tilde{x}_m[j]\cdot v_j[t];\ 0\le t\le n-1, \quad (14)$$

kde $\bar{\bar{x}}_m$ značí *m*-tý zrekonstruovaný obrázek,
$\tilde{x}_i$. značí *m*-tý eigenface,
k je počet použitých vektorů báze,
n je počet obrazových bodů výchozího obrázku.

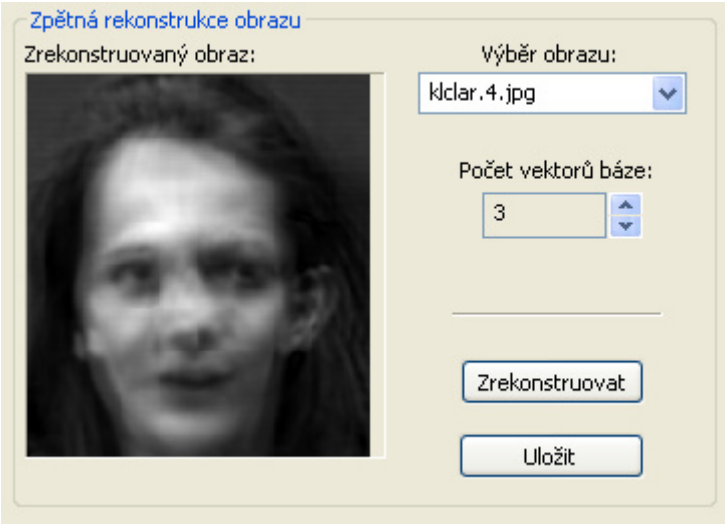
Po tomto kroku nesmíme zapomenout přičíst průměrný obrázek:

$$\hat{x}_m = \bar{\bar{x}}_m + M. \tag{15}$$

Zpětná rekonstrukce je zatížena určitou chybou ε, kterou můžeme obecně vyjádřit:

$$\varepsilon = \left\| \hat{x}_m - x_m \right\|. \tag{16}$$

Tím jsme popsali postup zpětné rekonstrukce snímků s možností volby počtu použitých bází. Na obr. 4 je uvedena část dialogového okna aplikace, kde probíhá proces zpětné rekonstrukce.



Obr. 4 Proces zpětné rekonstrukce

Pro jistou představu o vlivu počtu použitých vektorů báze na rekonstruovaný snímek uveďme na obr. 5 přehled snímků získaných z různých počtů vektorů báze.

a) b) c)

d) e)

Obr. 5 Vliv počtu vektorů báze na rekonstrukci obrázku- a) původní obrázek, b) 1 vektor, c) 3 vektory, d) 10 vektorů, e) 25 vektorů báze.

Vhodnějším měřítkem pro posouzení vlivu počtu vektorů báze na kvalitu rekonstruovaného obrázku může být zobrazení závislosti RMSE na počtu vektorů báze použitých k rekonstrukci obrázku, přičemž pro RMSE platí:

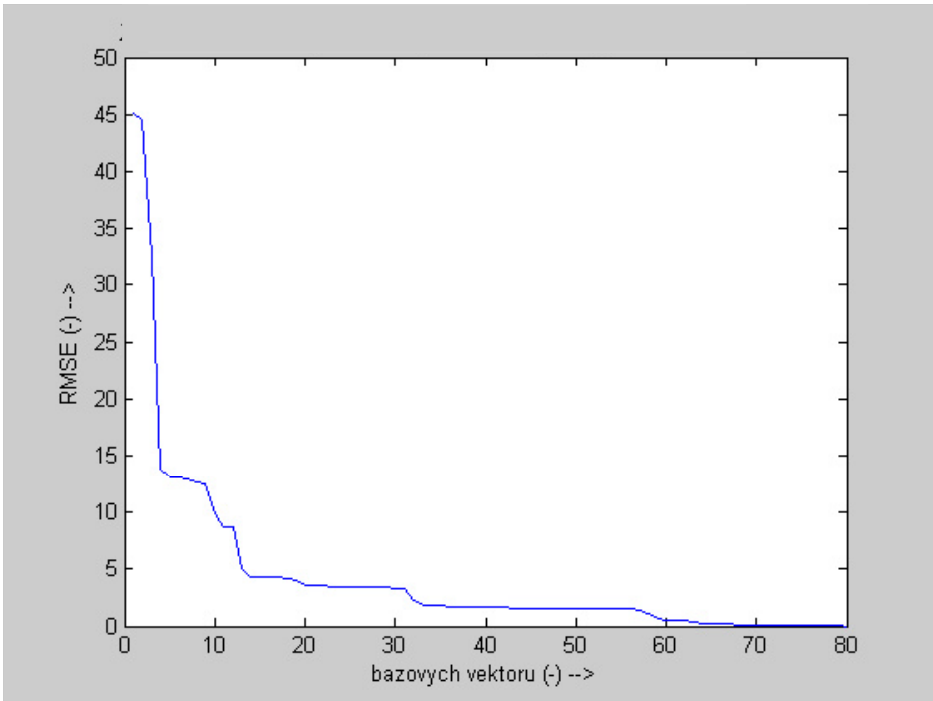
$$RMSE = \sqrt{\frac{1}{M \cdot N} \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\hat{x}_m(i,j) - x_m(i,j) \right)^2}, \qquad (17)$$

kde M resp. N je výška (počet řádků obrazové matice) resp. šířka (počet sloupců obrazové matice) obrazu,

$\hat{x}_m$ značí m- tý zrekonstruovaný obraz,

x_m značí m- tý originální obraz.

Uvažujme původní obrázek z obr. 5a) a tutéž trénovací sadu popsanou výše. Uvedenou závislost ukazuje obr. 6.



Obr.6 Závislost RMSE na počtu vektorů báze použitých k rekonstrukci obrazu

Stojí za povšimnutí, že význačný zlom v charakteristice nastává pro 5 vektorů báze, což je hodnota o 1 větší než je počet různých osob umístěných v trénovací sadě. Použijeme-li v našem případě k rekonstrukci 20 a více vektorů báze, není zde významný vizuální rozdíl mezi originálním a rekonstruovaným obrázkem.

4 Poznámky k implementaci aplikace

Na závěr popisu použití metody PCA uveďme některé poznámky z hlediska implementace, [1].

- V řadě vztahů se počítá s transponovanou formou matice. Samozřejmě není vůbec nutné mít vytvořenou vedle matice netransponované ještě matici transponovanou. Stačí u původní matice prohodit řádkový index se sloupcovým indexem.
- Pro velké trénovací sady vidíme vysoké paměťové nároky. Může se stát, že se dynamicky přidělovaná paměť vyčerpá dříve než se zahájí nějaký výpočet. Proto bychom neměli pro alokaci velkých polí a matic nutných k výpočtům používat operátory *new*, *malloc*, příp. u hodnotových polí metodu *resize*. V této práci je pro rozsáhlá pole a matice používán paměťově mapovaný soubor. Konkrétně jde o matici $\overline{X}$ uchovávající trénovací sadu obrázků a o matici V uchovávající bázi eigenspace. Zaplatíme za to ovšem určitou ztrátou rychlosti.
- Veškeré postupy předpokládaly vstupní obrázky ve stupních šedé. Na internetu lze ale nalézt rozsáhlé databáze barevných obrázků. Máme několik možností, jak se s tím vypořádat; buď budeme pracovat paralelně s každým kanálem zvlášť (pro velké časové a paměťové nároky je

to však nevhodné a též nelogické) nebo obrázky převedeme do stupňů šedé nebo si zvolíme jeden z barevných kanálů R,G,B a s ním metodu rozpoznávání provedeme. V této práci funguje poslední možnost (byl vybrán kanál R) a bez jakýchkoli problémů. Výsledky pro jeden ze zvolených kanálů zobrazujeme tak, že získané hodnoty obrazových bodů doplníme do zbývajících 2 kanálů (toto se týká zobrazení průměrného obrazu, zvoleného eigenface a zrekonstruovaného obrazu).

5 Shrnutí

Cílem tohoto příspěvku bylo ukázat využití metody PCA pro účely rozpoznávání a pro účely komprese obrazu a problematiku jejího programování v jazyce Visual C++. Zrealizovaná aplikace může sloužit ke studijním účelům v rámci seznamování se s metodou PCA nebo jako odrazový můstek pro další práci v této oblasti. Velice zajímavou návaznou prací může být oblast komprese videosekvencí, kde je nutné řešit optimalizace z hlediska vysoké časové a paměťové náročnosti metody. Lákavou oblastí je též realizace přístupového systému budov na základě rozpoznávání tváří. Zde by bylo nutné pouze vyřešit propojení snímací kamery s aplikací a ovládání nějakého elektronického zámku, neboť vše ostatní je již realizováno. V rámci realizace aplikace byly uvedené algoritmy pro kontrolu odzkoušeny v programovém prostředí MATLAB.

Tento příspěvek vznikl s přispěním grantu GAČR č.102/05/2054 a výzkumného záměru MSM 6840770014 (MŠMT).

Reference

[1] Fritsch, L. Programová podpora snímání obrazu. Praha, 2007. Diplomová práce FEL ČVUT. Vedoucí práce Mgr. Petr Páta, Ph.D.
[2] Stejskal, J. KLT obrazový kodér. Praha, 2005. Diplomová práce FEL ČVUT. Vedoucí práce Mgr. Petr Páta, Ph.D.
[3] JAMAC++ package [online].
URL: <http://math.nist.gov/tnt/overview.html>.
[4] Rozpoznávání tváří [soubor PDF].
URL: <http://pal.altmanoptik.com/download/facerec.pdf>.
[5] Efektivní vyhledávání v kolekcích obrázků tváří [soubor PDF].
URL: <http://www.cs.vsb.cz/kratky/courses/2003-04/reference/effface.pdf>.
[6] A tutorial on Principal Component Analysis [soubor PDF].
URL: <http://www.snl.salk.edu/~shlens/pub/notes/pca.pdf>.
[7] Collection of Facial Images: Faces94 [online].
URL: <http://cswww.essex.ac.uk/mv/allfaces/faces94.html>.

Ing. Lukáš Fritsch
ČVUT v Praze
Fakulta elektrotechnická, katedra radioelektroniky
Technická 2, Praha 6, 166 27
e-mail: fritsl1@fel.cvut.cz