Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Srovnání deskriptorů pro reprezentaci obrazu

Plzeň 2015 Vojtěch Košař

Originální zadání

Poděkování

Děkuji vedoucímu diplomové práce Ing. Ladislavu Lencovi, Ph.D. za jeho podporu, cenné rady, přínosné konzultace a vstřícný přístup během celé mojí práce. V neposlední řadě také děkuji mojí rodině, která mě podporovala v průběhu celého studia.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 12. května 2015

Vojtěch Košař

Abstract

Comparison of descriptors for image representation

This diploma thesis deals with comparsion of methods used for extraction of image descriptors intended to be used for face recognition. There are theoreticaly described LBP, LDP and POEM methods. These methods are implemented and their parameters are tuned on the FERET database. Software application destined for measuring of success of these methods was developed. Methods with tuned parameters are compared across different probe sets and their pros and cons are summarized. There is also proposed an experimental genetic algorithm that can be used to automatically tune weights of particular histograms in a histogram sequence.

Abstrakt

Srovnání deskriptorů pro reprezentaci obrazu

Tato diplomová práce se zabývá srovnáváním metod používaných pro extrakci obrazových deskriptorů při rozpoznávání obličejů. V práci jsou teoreticky popsány metody LBP, LDP a POEM. Tyto metody byly implementovány a jejich parametry byly naladěny na databázi FERET. Jako součást práce byla vyvinuta aplikace umožňující měření úspěšnosti při rozpoznávání obličejů implementovaných metod s různými parametry. Metody s naladěnými parametry byly následně porovnány na různých testovacích množinách a byly shrnuty výhody a nevýhody jednotlivých metod. Pro zvýšení úspěšnosti byl vyvinut experimentální genetický algoritmus, který umožňuje automaticky nastavit váhy jednotlivých histogramů.

Obsah

1 Úvod

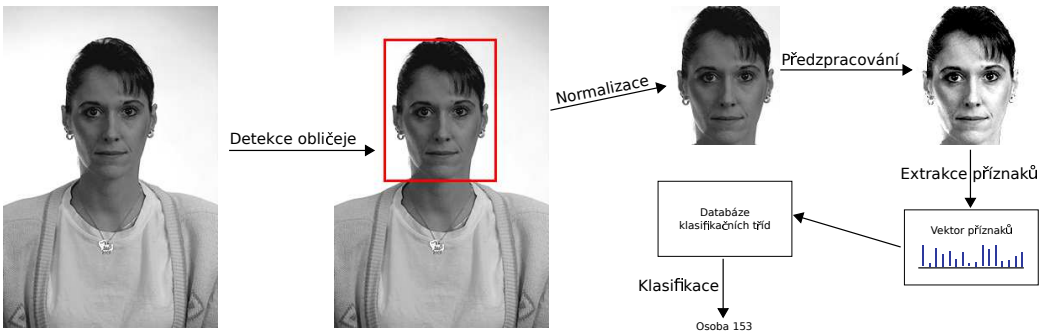
Deskriptory pro reprezentaci obrazu jsou hojně využívány v mnoha aplikacích digitálního zpracování obrazu a počítačového vidění. Uplatnění naleznou například při klasifikaci objektů, obličejů nebo textur. Jelikož je tato oblast poměrně mladá, stále poskytuje prostor pro navrhování nových a lepších obrazových deskriptorů. Zároveň zde však vzniká problém při hodnocení jejich deskriptivní síly v určité oblasti zájmu, která není dostatečně standardizovaná a poskytuje autorům nových obrazových deskriptorů velkou volnost při hodnocení jejich nově navrženého deskriptoru.

Cílem této práce je prostudovat metody pro extrakci obrazových deskriptorů a zhodnotit jejich vlastnosti při použití pro rozpoznávání obličejů. Vybrané metody budou implementovány a bude provedeno jejich objektivní porovnání formou měření jejich úspěšnosti při rozpoznávání obličejů na stejných datech. Hlavním přínosem této práce by mělo být právě objektivní srovnání vybraných metod. Autoři navržených metod obvykle detailně nepopisují způsob předzpracování snímků z veřejně dostupných databází a jejich výsledky často porovnávají s výsledky uvedenými v článku jiného autora, aniž by tuto metodu implementovali a provedli měření nad stejnými daty.

2 Teoretická část

2.1 Rozpoznávání obličeje

Rozpoznávání obličeje je úloha, při níž je obličeji ve formě dat (nejčastěji fotografie) přiřazen jednoznačný identifikátor, např. jméno. Schopnosti člověka oproti strojovému vidění jsou v tomto ohledu velmi vyspělé. V případě strojového rozpoznávání obličeje je třeba před vlastním rozpoznáváním provést mnoho úkonů, které člověk dělá automaticky (jako je například detekce obličeje, viz obrázek 2.1).



Obrázek 2.1: Schematický postup při strojové klasifikaci osoby z fotografie.

Metody rozpoznávání obličejů se v současné době těší velkému zájmu výzkumníků po celém světě. Jejich aplikace může nalézat uplatnění v mnohých odvětvích. Běžně se používá při detekci a přiřazování obličejů z fotek ke konkrétním lidem na sociálních sítích nebo při identifikaci hledaných či podezřelých osob v kriminálních databázích. Dále mohou nalézat uplatnění v autentizačních systémech, kde záznam obličeje ve formě fotografie nebo třeba 3D skenu slouží jako biometrická data podobně jako např. otisky prstů nebo snímek oční sítnice.

2.1.1 Strojová detekce obličeje

Metoda detekce obličeje spočívá v nalezení obličeje na předloženém snímku, respektive určení, zda se na snímku vůbec obličej nachází. V případě kladné odpovědi detektoru pak následuje lokalizace obličeje, která má za úkol označit část, kde se na obrázku obličej nachází, případně označit souřadnice různých částí obličeje jako např. oči, ústa, nos. V případě reálného nasazení některé z metod rozpoznávání obličejů jsou právě detekce a lokalizace první částí celého řetězu postupů.

Jednou z možných metod pro detekci obličeje je metoda založená na detekci barvy kůže. Tyto metody často používají jiný prostor barev než je běžný formát RGB, například formáty HSV [17], YIQ nebo YCbCr [16]. Zmíněné formáty jsou vhodnější pro svoji odolnost při pořizování snímků při různých světelných podmínkách.

Další z často používaných metod v této oblasti je Cascade Object Detector (COD) navržený v [6].

2.1.2 Strojové rozpoznávání obličejů

Úkolem metod pro rozpoznávání obličejů je přiřadit předloženému obličeji na obrázku nebo třeba z 3D skeneru konkrétní osobu. Jedná se o klasifikační úlohu, tzn. první fází rozpoznávání je fáze učení/trénování a druhou fází je samotná klasifikace, tedy přiřazení neznámého vzorku k některé ze známých tříd. Úspěšnost metod na rozpoznávání lze vyjádřit v % jako podíl úspěšně klasifikovaných[1] neznámých vzorků ku celkovému počtu neznámých vzorků.

[1] Přiřazených ke správné třídě.

Pro smysluplnost vyjádření úspěšnosti tímto způsobem je samozřejmě nutné, aby všechny neznámé (testovací) vzorky byly jednoznačně přiřaditelné alespoň do jedné z tříd trénovací množiny.

Jedním z postupů pro rozpoznávání obličejů je metoda zvaná Eigenfaces. Tato metoda byla poprvé prezentována v práci [8] v roce 1991. Základem této metody je hledání vhodného projektivního prostoru využitím matematického aparátu zvaného analýza hlavních komponent (Principal Component Analysis, PCA), čímž redukuje dimenzionalitu původního obrázku. Metoda ve své původní podobě má příliš mnoho nedostatků na to, aby mohla být v dnešní době reálně použitelná, ovšem její základ se využívá v mnoha dalších metodách na rozpoznávání [18]. Hlavní nevýhodou je extrémní citlivost na zarovnání obličeje a rovněž to, že pro přidání nové osoby do již natrénované databáze je třeba znovu vypočítat projektivní prostor ze všech snímků.

Existuje celá řada dalších metod. Např. metoda Linear Discriminant Analysis (LDA) [19], dále metody založené na analýze nezávislých komponent (Independent Component Analysis, ICA) [9], metody využívající umělou neuronovou síť [10] a zejména pak metody založené na extrakci obrazových deskriptorů, jimiž se tato práce zabývá a jejichž podrobný popis je uveden v sekci 2.2.

2.2 Metody pro extrakci obrazových deskriptorů

2.2.1 Příznaky

Abychom mohli objekty klasifikovat, musíme být v první řadě schopni popsat klasifikovaný objekt pomocí měřitelných vlastností (dat). Tato data mohou mít různou formu od geometrických rozměrů objektu přes sekvenci amplitud akustických tlaků (u zvuku) po 2D signál ve formě rastrového obrázku a další. V praxi se ukázalo, že klasifikace pouze na základě těchto „surových“ dat je často velmi obtížná, ale lze ji podstatně zjednodušit prostou transformací těchto dat. Tento postup se nazývá extrakce příznaků. Cílem extrakce příznaků je tedy usnadnit práci klasifikátoru vhodnou transformací naměřených dat, jejíž způsob se odvíjí od konkrétní úlohy. Výsledkem této transformace je často také změna dimenze příznakového prostoru.

V oblasti klasifikace pomocí obrazových deskriptorů (vzorů) se jako vhodná transformace ukázala konstrukce histogramu četností těchto vzorů.

2.2.2 LBP

Local Binary Pattern (LBP) je velice jednoduchý obrazový deskriptor. Je často základem dalších složitějších metod obrazových deskriptorů. Základní verze LBP byla navržena roku 1996 v [7].

Vstupem je šedotónový obrázek, tzn. že každý pixel je reprezentován pouze svojí intenzitou. Potom se pro každý pixel provede porovnání intenzity pixelu

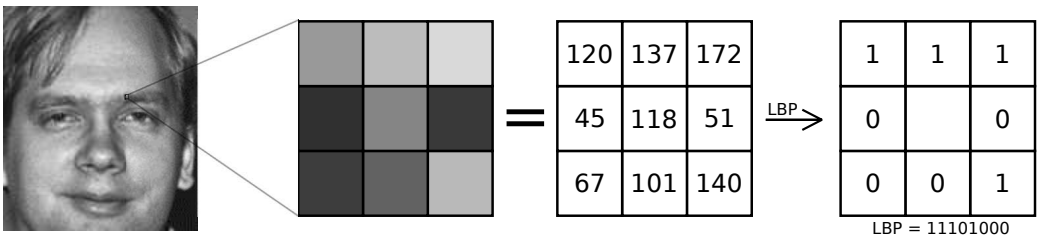
s intenzitou všech bezprostředně sousedících pixelů. Výsledek tohoto porovnání se pak binárně zakóduje funkcí $s(x)$ [1, 2]. Označíme-li p_c jako intenzitu centrálního pixelu a $p_0, p_1, ..., p_7$ jako intezity okolních pixelů jako je vidět na obrázku 2.2, pak se hodnota funkce $LBP(p_c)$ dá vyjádřit následující funkcí v rovnici 2.1. Příklad aplikace LBP operátoru je vidět na obrázku 2.3.

p_0	p_1	p_2
p_7	p_c	p_3
p_6	p_5	p_4

Obrázek 2.2: Označení okolních pixelů $p_0, p_1, ..., p_7$ kolem centrálního pixelu p_c.

$$LBP(p_c) = \sum_{i=0}^{7} s(p_i - p_c)2^i \tag{2.1}$$

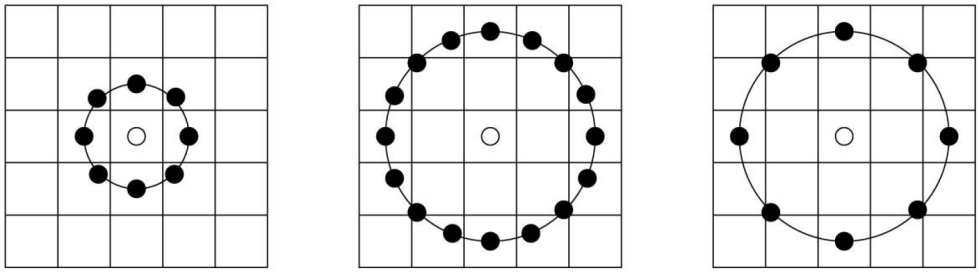
$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.2}$$



Obrázek 2.3: Příklad aplikace LBP operátoru na obrázek obličeje.

Kromě základní verze byl operátor LBP v práci [7] rozšířen tak, aby byl schopen si poradit i s popisem odlišně naškálovaných obrázků. V tomto případě se hovoří o zobecněné kruhové formě $LBP(N, R)$, kde N značí počet sousedních

bodů a R značí poloměr kružnice, na které jsou tyto sousední body rovnoměrně rozprostřeny, viz obrázek 2.4. Při zobecněné kruhové formě LBP se pro sousední body, které nemusejí být vždy přesně zarovnané s pozicí pixelů původního obrázku, používá pro výpočet intenzity těchto bodů bilineární interpolace. [15]



Obrázek 2.4: Příklad zobecněných kruhových operátorů $LBP(8,1)$, $LBP(16,2)$ a $LBP(8,2)$. Převzato z [2].

2.2.3 LDP

Obrazový deskriptor Local Derivative Pattern (LDP) je popsán v článcích [2, 3]. Jako vstupní šedotónový obrázek označíme $I(Z)$, kde Z představuje konkrétní pixel. Potom zkonstruujeme LDP prvního řádu ve směrech 0°, 45°, 90° a 135° a označíme je jako $I'_\alpha(Z)$, kde $\alpha = 0°, 45°, 90°$ a 135°. Nechť Z_0 je pixel $I(Z)$, potom $Z_i, i = 0, ..., 8$ jsou pixely sousedící s pixelem Z_0 jako je znázorněno na obrázku 2.5. Pak LDP prvního řádu ve všech čtyřech směrech pro pixel $Z = Z_0$ mohou být zapsány jako v rovnicích 2.3, 2.4, 2.5, 2.6.

Z_1	Z_2	Z_3
Z_8	Z_0	Z_4
Z_7	Z_6	Z_5

Obrázek 2.5: Označení okolních pixelů $Z_1, Z_2, ..., Z_8$ kolem centrálního pixelu Z_0.

$$I'_{0°}(Z_0) = I(Z_0) - I(Z_4) \tag{2.3}$$

$$I'_{45°}(Z_0) = I(Z_0) - I(Z_3) \tag{2.4}$$

$$I'_{90°}(Z_0) = I(Z_0) - I(Z_2) \tag{2.5}$$

$$I'_{135°}(Z_0) = I(Z_0) - I(Z_1) \tag{2.6}$$

LDP druhého řádu ve směru α v bodě $Z = Z_0$ označené jako $LDP^2_\alpha(Z_0)$ definujeme podle rovnice 2.7.

$$LDP^2_\alpha(Z_0) = f(I'_\alpha(Z_0), I'_\alpha(Z_1)), f(I'_\alpha(Z_0), I'_\alpha(Z_2)), ..., f(I'_\alpha(Z_0), I'_\alpha(Z_8)) \tag{2.7}$$

kde $f(.,.)$ je binární kódovací funkce rozhodující o typu lokálního vzoru přechodu. Tato funkce je definována v rovnici 2.8

$$f(I'_\alpha(Z_0), I'_\alpha(Z_i)) = \begin{cases} 0, & I'_\alpha(Z_i) \cdot I'_\alpha(Z_0) > 0 \\ 1, & I'_\alpha(Z_i) \cdot I'_\alpha(Z_0) \leq 0 \end{cases} \quad i = 1, 2, ..., 8. \tag{2.8}$$

Nakonec je LDP druhého řádu pro pixel Z značeno $LDP^2(Z)$, definováno jako zřetězení čtyř osmibitových směrových LDP^2, viz rovnice 2.9.

$$LDP^2(Z) = \{LDP^2_\alpha(Z) | \alpha = 0°, 45°, 90°, 135°\} \tag{2.9}$$

Obrázek 2.6 demonstruje výpočet LDP druhého řádu pro lepší představu na konkrétním příkladu.

2	5	3	5	1
6	7	9	1	5
2	3	**4**	8	2
3	2	3	2	9
1	2	3	2	1

Obrázek 2.6: Demonstrační příklad výpočtu LDP. Číselná hodnota v každé buňce představuje intenzitu příslušného pixelu.

Nejprve vypočteme LDP prvního řádu ve čtyřech směrech pro bod Z_0.

$$I'_{0°}(Z_0) = I(Z_0) - I(Z_4) = 4 - 8 = -4$$
$$I'_{45°}(Z_0) = I(Z_0) - I(Z_3) = 4 - 1 = 3$$
$$I'_{90°}(Z_0) = I(Z_0) - I(Z_2) = 4 - 9 = -5$$
$$I'_{135°}(Z_0) = I(Z_0) - I(Z_1) = 4 - 7 = -3$$

Potom vypočteme LDP prvního řádu ve čtyřech směrech pro bod Z_1.

$$I'_{0°}(Z_1) = 7 - 9 = -2$$
$$I'_{45°}(Z_1) = 7 - 3 = 4$$
$$I'_{90°}(Z_1) = 7 - 5 = 2$$
$$I'_{135°}(Z_1) = 7 - 2 = 5$$

Obdobně postupujeme pro všechny okolní body $Z_2 ... Z_8$ a pro každý bod

dostáváme vektor $\vec{z_i}$ čtyř celých čísel.

$$\begin{aligned}
I'_{0°,45°,90°,135°}(Z_0) &= \vec{z}_0 = (-4, 3, -5, -3) \\
I'_{0°,45°,90°,135°}(Z_1) &= \vec{z}_1 = (-2, 4, 2, 5) \\
I'_{0°,45°,90°,135°}(Z_2) &= \vec{z}_2 = (8, 4, 6, 4) \\
I'_{0°,45°,90°,135°}(Z_3) &= \vec{z}_3 = (-4, 0, -1, -2) \\
I'_{0°,45°,90°,135°}(Z_4) &= \vec{z}_4 = (6, 3, 7, -1) \\
I'_{0°,45°,90°,135°}(Z_5) &= \vec{z}_5 = (-7, 0, -6, -2) \\
I'_{0°,45°,90°,135°}(Z_6) &= \vec{z}_6 = (1, -5, -1, 0) \\
I'_{0°,45°,90°,135°}(Z_7) &= \vec{z}_7 = (-1, -2, -1, 0) \\
I'_{0°,45°,90°,135°}(Z_8) &= \vec{z}_8 = (-1, -6, -4, -3)
\end{aligned}$$

Nakonec vypočteme LDP druhého řádu postupně ve všech čtyřech směrech v bodě Z_0 s využitím binární kódovací funkce z rovnice 2.8.

$$\begin{aligned}
f(I'_{0°}(Z_0), I'_{0°}(Z_1)) &= 0, \text{ protože } (-4) \cdot (-2) > 0 \\
f(I'_{0°}(Z_0), I'_{0°}(Z_2)) &= 1, \text{ protože } (-4) \cdot 8 \leq 0 \\
f(I'_{0°}(Z_0), I'_{0°}(Z_3)) &= 0, \text{ protože } (-4) \cdot (-4) > 0 \\
&\vdots \\
f(I'_{0°}(Z_0), I'_{0°}(Z_8)) &= 0, \text{ protože } (-4) \cdot (-1) > 0
\end{aligned}$$

Tím získáme LDP vzor druhého řádu ve směru $\alpha = 0°$.

$$LDP^2_{0°}(Z_0) = \{01010100\}$$

Podobně postupujeme pro další směry 45°, 90°, 135°.

$$\begin{aligned}
f(I'_{45°}(Z_0), I'_{45°}(Z_1)) &= 0, \text{ protože } 3 \cdot 4 > 0 \\
&\vdots \\
f(I'_{45°}(Z_0), I'_{45°}(Z_8)) &= 1, \text{ protože } 3 \cdot (-6) \leq 0 \\
f(I'_{90°}(Z_0), I'_{90°}(Z_1)) &= 1, \text{ protože } (-5) \cdot 2 \leq 0 \\
&\vdots \\
f(I'_{90°}(Z_0), I'_{90°}(Z_8)) &= 0, \text{ protože } (-5) \cdot (-4) > 0 \\
f(I'_{135°}(Z_0), I'_{135°}(Z_1)) &= 1, \text{ protože } (-3) \cdot 5 \leq 0 \\
&\vdots \\
f(I'_{135°}(Z_0), I'_{135°}(Z_8)) &= 0, \text{ protože } (-3) \cdot (-3) > 0
\end{aligned}$$

Zřetězením těchto bitů dostaneme finální reprezentaci pro prostřední pixel Z na obrázku 2.6 $LDP^2(Z) = \{01010100\ 00101111\ 11010000\ 11000110\}$.

2.2.4 POEM

Obrazový deskriptor Patterns of Oriented Edge Magnitudes (POEM) je popsán v článku [4]. Cílem při návrhu POEM deskriptoru bylo, aby:

1. Byl robustní vůči změnám při osvětlení, kvalitě obrázku a věku osoby na snímku.
2. Doba výpočtu byla rychlá. Tzn. aby výpočet extrakce příznaků i klasifikátor byl rychlý.
3. Extrakce a klasifikace byly automatické. Tzn. aby nebylo zapotřebí ručního zásahu, např. pro stanovení výrazu v obličeji.

4. Řešení bylo snadno škálovatelné. Tzn. aby při přidání nového obličeje do databáze nebylo4 potřeba ji přetrénovat.

5. Systém nespoléhal na externí zdroje dat, jako např. negativní příklady obličejů.

6. Systém byl schopný pracovat i pouze s jedním obrázkem v galerii.

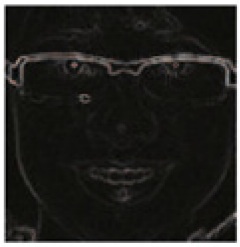
Následující část bude popisovat způsob výpočtu POEM deskriptoru. Vstupem pro metodu POEM je šedotónový obrázek obličeje o rozměrech $m \times n$, jako např. na obrázku 2.7.

Obrázek 2.7: Příklad vstupního obrázku. Převzato z [4].

Předpokládá se, že obličej na obrázku je normalizovaný (zarovnaný a naškálovaný). Výpočet POEM deskriptoru probíhá následujícím způsobem:

1. *Výpočet gradientu obrázku*

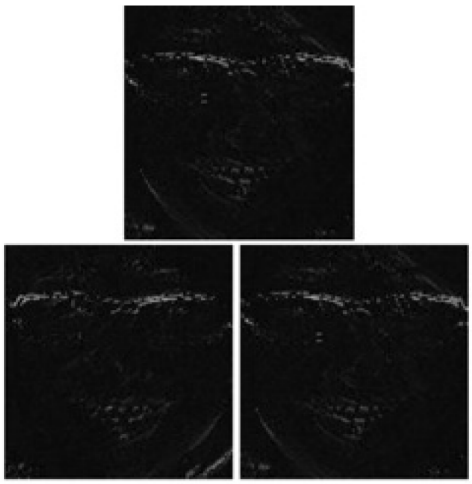
   Prvním krokem je výpočet gradientu obrázku, což lze provádět různými způsoby. Použitá metoda pro výpočet gradientu obrázku není v [4] zmíněná. Výstupem jsou pak dva obrázky dx a dy o rozměrech $m \times n$, respektive každý bod původního obrázku je reprezentován jako $2D$ vektor. Na obrázku 2.8 je zobrazen výstup, přičemž intenzita v každém pixelu představuje velikost vektoru gradientu.



Obrázek 2.8: Příklad obrázku po výpočtu gradientu obrázku. Převzato z [4].

2. *Diskretizace směru gradientů*

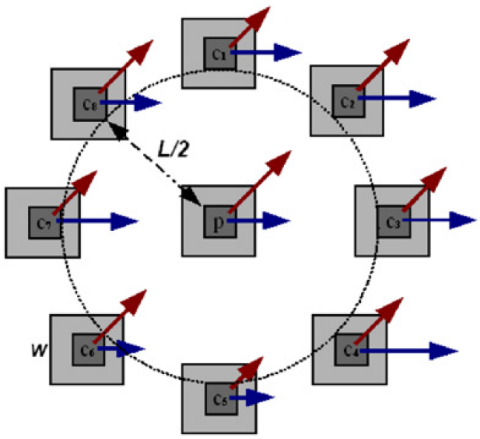
   Směr gradientu je rovnoměrně diskretizován přes interval $0 - \pi$ v případě neznaménkové reprezentace respektive $0 - 2\pi$ v případě znaménkové reprezentace. To znamená, že každý pixel obrázku je reprezentován směrem (diskretizovaným) a velikostí. Počet diskrétních směrů budeme značit d. Na obrázku 2.9 jsou zobrazeny výstupy pro zvolené tři diskrétní směry. Intenzita v každém pixelu opět představuje velikost vektoru gradientu, tentokrát pro konkrétní diskrétní směr.



Obrázek 2.9: Příklad obrázků po diskretizaci směrů gradientů. Převzato z [4].

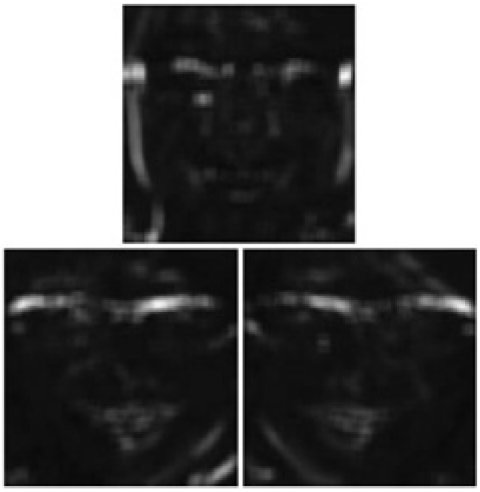
3. *Výpočet lokálního histogramu orientace gradientů z okolí*

   Každému pixelu je přiřazen vektor s počtem prvků d, který představuje lokální histogram směrů gradientů z okolí pixelu nazvaném *cell*. Toto okolí je znázorněno světle šedivou čtvercovou oblastí kolem každého pixelu $p, c_1, c_2, ..., c_8$ na obrázku 2.10.



Obrázek 2.10: Znázornění pojmů *cell* a *block* při konstrukci POEM deskriptoru. Převzato z [4].

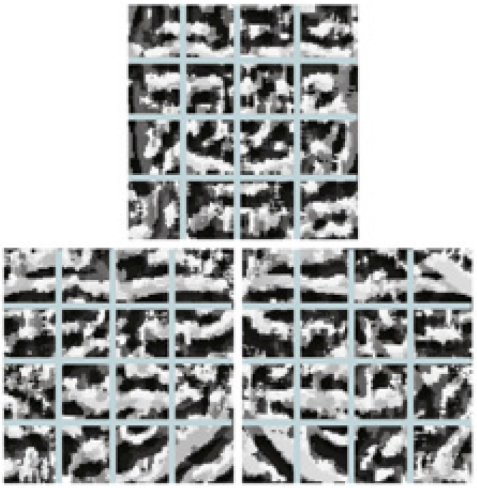
Jako váhu příspěvku každého gradientu lze zvolit buď přímo jeho velikost nebo nějakou funkci jeho velikosti. Ke zvýšení důležitosti středového pixelu je možné použít rovněž váhové okno, jako například Gaussův filtr nebo binomické jádro, avšak podle [4] to nemá vliv na popisovací schopnosti tohoto deskriptoru. Namísto přiřazení vektoru každému pixelu si lze přestavit reprezentaci dat jako d obrázků, kde každý obrázek představuje jeden diskrétní směr, respektive intenzita pixelu daného obrázku představuje průměrnou velikost gradientu z okolí původního obrázku. Ukázka po výpočtu lokálních histogramů orientace gradientů z okolí je na obrázku 2.11.



Obrázek 2.11: Příklad obrázků po výpočtu lokálních histogramů orientace gradientů z okolí. Převzato z [4].

4. *Zakódování příznaků pomocí LBP operátoru*

   Původní LBP operátor je aplikován na okolí 3×3 každého pixelu. V případě deskriptoru POEM se jedná o kruhové okolí pixelu zvané *block* s poloměrem $L/2$. Toto kruhové okolí s poloměrem $L/2$ je vidět na obrázku 2.10. Pro stanovení intenzit okolních hodnot je možné použít bilineární interpolaci. Pro zvýšení stability v téměř konstantní oblasti lze k centrálnímu prahovacímu pixelu přičítat malou konstantu τ. Ukázka výstupu po aplikaci LBP operátoru je na obrázku 2.12.



Obrázek 2.12: Obrázky po aplikaci LBP operátoru a rozdělení pravidelnou čtvercovou mřížkou. Převzato z [4].

5. *Výpočet dílčích histogramů a konstrukce globálního histogramu*

   Po aplikaci upraveného LBP operátoru podle předchozího bodu jsou všechny obrázky rozděleny pravidelnou čtvercovou mřížkou jako je vidět na obrázku 2.12. Pro každou oblast je vypočten histogram četností jednotlivých vzorů v této oblasti, přičemž jsou vybírány pouze uniformní vzory, viz podsekce 2.2.6 a 2.2.5. Tyto dílčí histogramy jsou nakonec zřetězeny přes všechny diskrétní směry do jednoho globálního histogramu, který reprezentuje celý obličej.

2.2.5 Sekvence histogramů

Histogram nese určitou informaci o zastoupení jednotlivých příznaků, respektive binárních vzorů na celém obrázku. Konstrukcí takového histogramu však ztrácíme informaci o prostorovém rozložení těchto příznaků na obrázku. Prostorová informace je ale v případě rozpoznávání obličejů nepochybně důležitá.

Její zanesení do příznaků ve formě histogramů řeší sekvence histogramů (Histogram Sequence, HS). HS se konstruuje tak, že se histogram nevytváří nad celým obrázkem, ale nad jeho různými částmi. Tyto jednotlivé histogramy se zřetězí, čímž se vytvoří globální příznak celého obrázku ve formě histogramu, který již obsahuje i prostorovou informaci. Jednotlivé rozdělení na části je již závislé na konkrétním použití (např. formátu dat). Nejčastěji se používá čtvercová mřížka s konstantním rozměrem přes celý obrázek, ale je možné použít i různě velké obdélníky nebo i jiné tvary. Oblasti dílčích histogramů se mohou i překrývat. Příklad konstrukce pravidelné čtvercové mřížky je na obrázku 2.13. [1]

Obrázek 2.13: Ukázka konstrukce pravidelné mřížky s různou velikostí. Převzato z [1].

2.2.6 Uniformní vzory

Možným rozšířením základní verze konstrukce histogramu je vybírání pouze tzv. uniformních vzorů. To může být použito k podstatné redukci délky příznakového vektoru. Toto rozšíření je inspirováno faktem, že některé binární vzory se na běžných obrázcích vyskytují častěji, než jiné. V článku [7] bylo experimentálně zjištěno, že kolem 90% všech vzorů na zkoumaných obrázcích je uniformních. Uniformní vzory jsou takové hodnoty čísla[2], kdy dochází k ma-

[2]Číslo z pohledu binární reprezentace.

ximálně dvěma změnám[3] (přechodům) z 0 na 1 a nebo opačně. Například 00011110 a 10000001 jsou uniformní, zatímco 01101111 není. Uniformních hodnot je v tomto případě pouze 58 a při konstrukci histogramu se všechny neuniformní hodnoty přiřazují do jednoho vyhrazeného binu. Takto dojde k redukci rozměru histogramu z 256 na 59 binů. [1]

2.3 Porovnávání histogramů

Metod pro měření podobnosti histogramů je celá řada. V následujícím výčtu metod předpokládejme značení H_i pro histogram, n pro počet prvků (binů) histogramu. Mezi nejčastěji používané patří:

- Eukleidovská vzdálenost

$$d(H_1, H_2) = \sqrt{\sum_{i=1}^{n}(H_1(i) - H_2(i))^2}$$

- χ^2 vzdálenost

$$d(H_1, H_2) = \sum_{i=1}^{n} \frac{(H_1(i) - H_2(i))^2}{2(H_1(i) + H_2(i))}$$

- Intersekce

$$s(H_1, H_2) = \sum_{i=1}^{n} min(H_1(i), H_2(i))$$

[3] V případě pohledu na hodnoty vzoru ve smyslu kruhu.

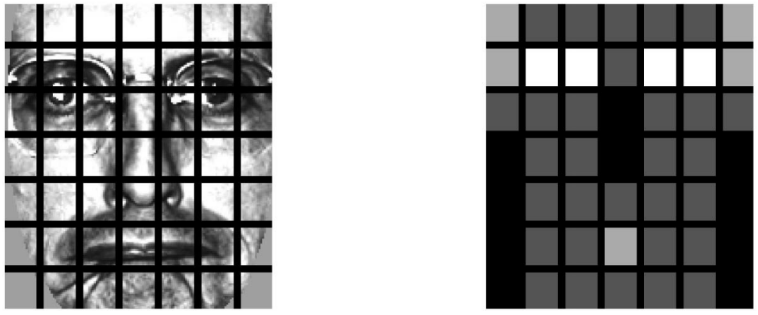
2.4 Modifikace metod s cílem zlepšení úspěšnosti

2.4.1 Předzpracování

Úspěšnost metod pro rozpoznávání je možné zvýšit vhodným předzpracováním původního obrázku ještě před extrakcí příznaků obrazovým deskriptorem. Často používanou metodou je např. gamma korekce nebo adaptivní gamma korekce, které vyrovnávají intenzity pixelů, což je vhodné zejména u fotografií pořízených při špatném (nevhodném) osvětlení. Další častou metodou je odstranění vysokofrekvenčního šumu na obrázku aplikací Gaussova filtru (rozostření) případně použitím rozdílu gaussiánů (Difference of Gaussians, DoG), který funguje při určitém nastavení jeho parametrů jako hranový detektor. [20]

2.4.2 Váhování histogramů

Při konstrukci sekvence histogramů popsané v sekci 2.2.5 může vyvstat myšlenka, že každý dílčí histogram nepřispívá k odlišující informaci mezi obličeji stejnou vahou. V článku [1] byly jednotlivým histogramům ručně přiřazeny váhy, jak je možné vidět na obrázku 2.14, s cílem vyzdvihnout rozlišovací schopnost určitých částí obličeje a naopak potlačit některé nedůležité části.



Obrázek 2.14: Manuální přiřazení vah dílčím histogramům. Černá barva představuje váhu 0, tmavě šedá barva váhu 1, světle šedá barva váhu 2 a bílá barva váhu 4. Převzato z [1].

2.4.3 Učení

Všechny zmíněné popisy metod se zabývaly výhradně tzv. extrakcí příznaků. Účelem extrakce příznaků je získání informace ze snímku a je to v podstatě nejnižší úroveň zpracování vstupu, která předchází komplexním úlohám. Kvalita příznaků pak výrazně ovlivňuje výsledek zpracovávané úlohy. Zatímco cílem této práce je porovnat diskriminační schopnost samotných deskriptorů (LBP, LDP, POEM), tak v praxi po fázi extrakce příznaků často následuje fáze učení, která má jako vstup právě tyto extrahované příznaky a snaží se na jejich základě pomocí méně či více sofistikovaných postupů klasifikovat neznámé obrázky. Základem všech učících se algoritmů je fakt, že se vzrůstající velikostí trénovací množiny by se měla zvyšovat úspěšnost při klasifikaci nebo rozpoznávání. Nevýhodou je pak jistě výpočetní složitost a doba trénování, která je při použití učících se algoritmů větší než při obyčejné extrakci příznaků a klasifikaci formou hledání nejbližší třídy v příznakovém prostoru.

2.5 Databáze FERET

Databáze obličejů FERET vznikla v rámci sponzorovaného programu FERET [11]. Cílem bylo na jejím základě vyvinout algoritmy na rozpoznávání obličejů. Před vznikem databáze FERET nebyl žádný standard na vyhodnocení úspěšnosti algoritmů pro detekci a rozpoznávání obličejů. Mnoho výzkumníků si pro svoje experimenty vytvářelo vlastní databáze, ale jejich velikost byla málokdy větší než 50 osob. Databáze FERET vznikala během 15 sezení provedených mezi roky 1993 a 1996. Databáze obsahuje celkem 14126 obrázků 1199 individuálních osob. Duplikátní sady obrázků jsou většinou pořízeny v různý den, za jiných světelných podmínek nebo s jiným výrazem v obličeji.

Zveřejnění této databáze umožnilo výzkumníkům vyvíjet algoritmy a porovnávat výsledky na společné databázi. Výsledky zveřejněné v různých článcích však přesto nejsou zcela porovnatelné, jelikož záleží na výběru části obličeje z původního obrázku na základě metadat obsahujících pozice různých částí obličeje (oči, nos, ústa). Také záleží na volbě parametrů pro použitou metodu, které mohou být naladěny na míru této databáze. I přes to je FERET databáze často používána jako referenční a lze ji použít při srovnávání úspěšnosti při rozpoznávání obličejů různých metod při souměřitelných volbách parametrů a stejném výřezu a předzpracování obrázků. Nejčastěji se při měření úspěšnosti metod používají podmnožiny obrázků *fa*, *fb*, *fc*, *dup1* a *dup2*, které jsou foceny zepředu pod nulovým úhlem. Podmnožina *fa* obsahuje 1196 obrázků osob, *fb* obsahuje 1195 obrázků osob s jiným výrazem v obličeji, *fc* obsahuje 194 obrázků osob pořízených při jiných světelných podmínkách než u *fa*. Podmnožina *dup1* obsahuje 722 obrázků 243 osob, které byly pořízeny později v čase v rozmezí 0 až 34 měsíců a *dup2* obsahuje 234 obrázků 75

osob, které byly pořízeny alespoň 18 měsíců déle než v *fa*. [12]

2.6 Výsledky prezentované v literatuře

V článku [1] jsou prezentovány výsledky úspěšnosti klasifikace používající zobecněnou kruhovou metodu $LBP(8,2)$ a její váženou verzi. Pro měření podobnosti histogramů se používá χ^2 vzdálenost. V článku není popsán způsob normalizace obličejů. Váhy dílčích histogramů byly zvoleny manuálně.

Metoda	**fb**	**fc**	**dup1**	**dup2**
LBP(8,2)	0.93	0.51	0.61	0.50
Vážená LBP(8,2)	0.97	0.79	0.66	0.64
POEM-HS	0.976	0.960	0.778	0.765
LBP v článku [2]	0.91	0.63	0.53	0.38
LDP 2. řádu	0.93	0.82	0.61	0.52
LDP 3. řádu	0.90	0.88	0.63	0.61
LDP 4. řádu	0.86	0.80	0.55	0.52

Tabulka 2.1: Přehled výsledků prezentovaných v literatuře.

V článku [2] jsou prezentovány výsledky úspěšnosti klasifikace používající metodu LDP. Přesný popis normalizace a škálování obličejů zde není popsán, ale jsou zde uvedeny výsledky rovněž pro metodu LBP. Výsledky pro databázi FERET jsou prezentovány pouze ve formě grafů a tak hodnoty v tabulce 2.1 pro LDP nemusejí být zcela přesné. LDP druhého řádu si vedlo na všech množinách dat lépe než obyčejné LBP. V případě třetího řádu LDP je na množině *fc* zlepšení oproti LDP druhého řádu, avšak na množině *fb* dochází ke zhoršení úspěšnosti při rozpoznávání. U čtvrtého řádu LDP pak dochází

ke zhoršení na všech množinách.

V článku [4] jsou prezentovány výsledky úspěšnosti klasifikace používající metodu POEM. Parametry metody POEM byly naladěny na databázi FERET s normalizovanými obrázky o rozměrech 110×110 pixelů. Nejlepších výsledků bylo dosaženo použitím neznaménkové reprezentace gradientu o třech diskrétních směrech s velikostí bloku (*block*) 10×10 pixelů a velikostí okolí (*cell*) 7×7 pixelů. V histogramech se pak pracovalo pouze s uniformními vzory.

3 Realizační část

3.1 Volba metod pro implementaci

Po prostudování materiálů a konzultaci s vedoucím mojí práce jsem se rozhodl pro implementaci metod LBP, LDP a POEM. Metodu LBP jsem zařadil do výběru jako referenční ke zbývajícím složitějším metodám. Metody LDP a POEM byly vybrány zejména z důvodu jejich vysoké úspěšnosti prezentované v literatuře a možnosti výpočtu v reálném čase. Je zde také přímá možnost porovnání, protože zveřejněné výsledky k nim byly srovnávány se základní metodou LBP. V článcích se však jejich autoři odkazovali pouze na výsledky provedené jinými autory, kteří pravděpodobně používali jiný způsob normalizace obličejů. Pro objektivní srovnání je však nezbytné, aby se tato měření úspěšností při rozpoznávání prováděla nad stejnými daty a se stejným způsobem předzpracování.

Pro implementaci těchto metod a měření jejich úspěšnosti byl zvolen jazyk C++ a multiplatformní knihovna OpenCV zejména kvůli jejich rychlosti, jelikož při měření bude třeba provádět mnoho výpočetně a časově náročných testů. Popisu knihovny OpenCV se věnuje sekce 3.2.

3.2 OpenCV

Open source Computer Vision (OpenCV) je softwarová knihovna vydávaná pod licencí BSD a je tak zdarma dostupná pro akademické i komerční účely.

Je k dispozici pro programovací jazyky C, C++, Python a Java a podporuje operační systémy Windows, Linux, Mac OS, iOS a Android. OpenCV byla navržena jako výpočetně efektivní knihovna se silným důrazem na výpočty prováděné v reálném čase. Své uplatnění nalézá od zpracování videa a fotografií v reálném čase, přes rozpoznávání gest na dotykových obrazovkách po rozpoznávání objektů a mnoho dalšího.

3.2.1 Instalace na operačním systému Windows

Tato část stručně popisuje instalaci a nastavení *OpenCV v.2.4.9* knihovny na operačním systému *Windows 7* s nainstalovaným vývojovým prostředím *Visual Studio 2010 Service Pack 1*.

1. *Stažení knihovny*

   Z adresy `opencv.org/downloads.html` stáhnout instalační soubor *OpenCV for Windows VERSION 2.4.9*.

2. *Instalace*

   Po stažení spustit samorozbalovací archiv s právy administrátora (spustit jako správce). Zvolit vhodnou cestu, např. `C:\OpenCV\`.

3. *Nastavení systémových proměnných*

   Následující postup předpokládá umístění rozbaleného archivu do `C:\OpenCV\`.

   Stačí spustit konzoli s právy administrátora (např. nabídka start, zadat `cmd.exe`, kliknout pravým tlačítkem myši a spustit jako správce) a zadat: `setx -m OPENCV_DIR C:\OpenCV\Build\x86\vc10`.

4. *Nastavení cesty k dynamickým knihovnám*

   Nabídka start, kliknout pravým tlačítkem myši na „Počítač“ a zvolit „Vlastnosti“, potom kliknout na „Změnit nastavení“. Následně překliknout na záložku „Upřesnit“ a zvolit „Proměnné prostředí“. V části „Uživatelské proměnné“ do proměnné „PATH“ přidat nakonec řetězec `;%OPENCV_DIR%\bin`. V části „Systémové proměnné“ vytvořit novou proměnnou nazvanou `OPENCV_DIR` s hodnotou `C:\opencv\build\x86\vc10`.

5. *Nastavení projektu ve Visual Studiu*

   Po založení projektu (např. konzolové aplikace) je třeba zobrazit „*Property Manager*“, což se dá udělat např. překliknutím záložky v oblasti „*Solution Explorer*“. Dále kliknout na složku „*Debug | Win32*“ pravým tlačítkem myši a zvolit „*Add New Project Property Sheet*“. Jméno zvolit třeba „*OpenCV_Debug*“ a přidat do projektu tlačítkem „*Add*“.

   Dvojklikem otevřít právě přidaný soubor a vybrat
   `Common properties > C/C++ > General`.
   Do kolonky „*Addition Include Directories*“ vpravo vložit řetězec `$(OPENCV_DIR)\..\..\include`.

   Potom vybrat `Common properties > Linker > General` a do kolonky „*Addition Library Directories*“ vpravo vložit řetězec `$(OPENCV_DIR)\lib`.

   Nakonec vybrat `Common properties > Linker > Input` a do kolonky „*Addition Dependencies*“ přidat seznam souborů :

   ```
   opencv_core249d.lib
   opencv_imgproc249d.lib
   ```

```
opencv_highgui249d.lib
opencv_ml249d.lib
opencv_video249d.lib
opencv_features2d249d.lib
opencv_calib3d249d.lib
opencv_objdetect249d.lib
opencv_contrib249d.lib
opencv_legacy249d.lib
opencv_flann249d.lib
```

Zcela stejný postup je třeba ještě provést ve složce „*Release | Win32*“ s jediným rozdílem, že výše zmíněné soubory `*d.lib` musejí být přidány bez posledního „*d*“ před tečkou (např. pouze `opencv_core249.lib`).

3.2.2 Použití

Následující kód ukazuje jednoduché použití knihovny OpenCV. Program pouze načte obrázek specifikovaný v prvním argumentu programu a zobrazí ho v okně, dokud uživatel nestiskne klávesu. Pro jednoduchost zde nejsou nijak ošetřené vstupy programu a předpokládá se, že se zvolený obrázek podaří načíst.

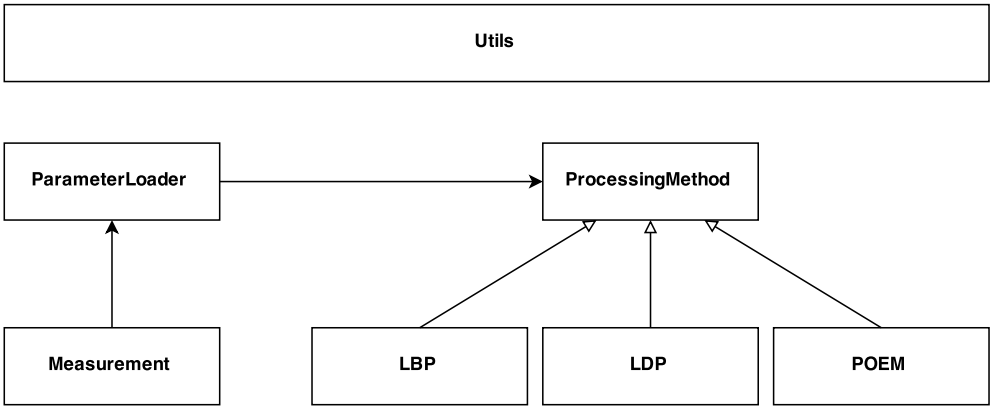
```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, char** argv )
{
    Mat image;
    /* Nacte obrazek specifikovany v prvnim argumentu programu. */
    image = imread(argv[1], CV_LOAD_IMAGE_COLOR);
    /* Vytvori okno. */
    namedWindow("Display window", WINDOW_AUTOSIZE);
    /* Ve vytvorenem okne zobrazi nacteny obrazek */
    imshow("Display window", image);
    /* Cekani na stisk klavesy. */
    waitKey(0);
    return 0;
}
```

3.3 Architektura aplikace

Architektura aplikace je logicky rozdělena do několika částí, viz obrázek 3.1. V dalších částech bude funkce těchto částí stručně popsána.



Obrázek 3.1: Architektura aplikace.

3.3.1 ProcessingMethod

Třída `ProcessingMethod` je zastřešující rodičovskou třídou pro jednotlivé metody pro výpočet obrazových deskriptorů. Obsahuje veřejné metody pro načtení obrázku ze souboru podle zadané cesty, metody pro nastavení a aplikaci předzpracování na načtené obrázky a další. Dále obsahuje implementaci pro výpočet dílčích histogramů a konstrukci sekvence histogramů. V neposlední řadě obsahuje čistě virtuální metodu `ProcessImage`, kterou musí všechny oděděné třídy implementovat.

3.3.2 ParameterLoader

Třída `ParameterLoader` slouží k načítání inicializačního souboru, viz podsekce 3.5.1. Její nejdůležitější metodou je metoda `LoadParametersFromFile`, která má jako parametr cestu k inicializačnímu souboru. V případě, že se načtení inicializačního souboru podaří (metoda vrátí 0), je možné následně volat její veřejné metody jako např. `GetHistSimilarityMethod`, která vrací typ metriky používané při porovnávání histogramů nebo `GetProcessingMethod`, která vrací instanci používané metody s již správně nastavenými parametry.

3.3.3 Measurement

Třída `Measurement` slouží pro měření úspěšnosti klasifikace při rozpoznávání obličejů. Tato třída obsahuje metody `LoadFiles`, která načítá galerii a testovací množinu obrázků. Dále metodu `Process`, která provede měření úspěšnosti a vrátí ji nebo je možné zavolat pouze metodu `LoadAndProcessFiles`,

která očekává jako svůj jediný parametr instanci `ParameterLoader`. Tato naposledy zmíněná metoda postupně provede načtení galerie i testovací množiny a následně provede měření úspěšnosti, kterou vrátí. Všechno potřebné nastavení získá pouze z instance `ParameterLoader`.

Následující část kódu demonstruje použití vyvinuté aplikace jako knihovny.

```
...

ParameterLoader pl;
/* Nacteni inicializacniho souboru pomoci ParameterLoaderu. */
if (pl.LoadParametersFromFile("parameters.ini") == 0)
{
  Measurement measurement;
  /* Zmereni uspesnosti pri rozpoznavani <0; 1>. */
  double success = measurement.LoadAndProcessFiles(&pl);
}

...
```

3.3.4 Načítání

Načítání obrázků pro další zpracování obstarává třída `Measurement`. Obrázky musejí splňovat určitá pravidla a musejí být uloženy v adresářích dodržujících předepsanou strukturu, viz podsekce 3.5.2. Samotné načtení pak obstarává metoda `LoadFiles`, která jako parametry potřebuje prefix cesty ke galerii, prefix cesty k testovací množině, počáteční a koncový index adresáře a příponu souborů s obrázky. Význam prefixu cesty a indexů adresářů je popsán v podsekci 3.5.2. K uchovávání dat k načteným obrázkům slouží privátní proměnné `gallerySet` a `probeSet` typu `std::vector<FileImage>`. `FileImage` je struktura obsahující cestu k obrázku, jeho index, který představuje jednoznačný identifikátor osoby na obrázku a vektor histogramů, který je po zpracování obrázku naplněn daty, představujícími globální histogram. Metoda `LoadFiles` postupně zavolá metodu `FillFileImages` nejprve pro

galerii a potom pro testovací množinu, čímž dojde k naplnění proměnných `gallerySet` a `probeSet`, respektive cest k souborům s obrázky a indexů osob na nich. V této metodě se postupně pro všechny indexy (od počátečního po koncový) prochází zvolený prefix cesty s konkatenovaným indexem představující adresář a v případě jeho existence se následně prohledává jeho obsah, respektive všechny soubory mající zvolenou příponu. Tyto nalezené soubory, respektive cesty k nim, se pak přidávají do vektoru množiny (galerie nebo testovací) i s příslušným indexem osoby.

3.3.5 Zpracování

Zpracování načtených obrázků obstarává metoda `Process` třídy `Measurement`. Ta jako parametry očekává instanci použité metody pro výpočet deskriptorů obrázků `ProcessingMethod` a typ použité metriky pro měření podobnosti histogramů. Třída `Measurement` rovněž obsahuje přetíženou variantu metody `Process`, která má jako svůj parametr pouze instanci typu `ParameterLoader`, ze které potřebné parametry sama získá. Tato metoda vrací úspěšnost klasifikace při rozpoznávání na intervalu $\langle 0; 1 \rangle$. Na začátku metody `Process` se postupně prochází oba vektory testovací množiny a galerie a pro každý obrázek se volá metoda `ProcessImage`. Tato metoda očekává jako parametry instanci použité metody `ProcessingMethod` a referenci na příslušný `FileImage`. Metoda `ProcessImage` nejprve načte obrázek ze souboru do paměti voláním metody `LoadImageFromFile` nad parametrem `method`. V případě, že se načtení nepodaří, vrátí `false`. V opačném případě nad parametrem `method` postupně zavolá metody `PreprocessImage`, která aplikuje na načtený obrázek řetěz úkonů pro předzpracování a následně `ProcessImage`, která naplní proměnnou `vectorOfHistograms` struktury

typu `FileImage`. Nakonec vrací `true`. Po načtení, předzpracování a zpracování všech obrázků z galerie i testovací množiny se provádí klasifikace, která je popsána v podsekci 3.3.6

3.3.6 Klasifikace

Klasifikace, respektive v tomto případě spíše výpočet úspěšnosti při rozpoznávání, probíhá jednoduchým způsobem. Postupně se prochází zpracované obrázky (`FileImage`) z testovací množiny a jejich globální histogram (`vectorOfHistograms`) se postupně porovnává se všemi globálními histogramy z galerie. Porovnání, respektive míru podobnosti histogramů mají na starost statické metody ze třídy `Utils`, jejichž parametry jsou pouze dva globální histogramy. O volbě volané metody na podobnost histogramů rozhoduje `histSimilarityMethod`, což je parametr metody `Process`. Implementace metod na podobnost histogramů je taková, že čím menší číslo vrátí, tím jsou si histogramy podobnější. Cílem je tedy najít nejpodobnější histogram z galerie ke každému histogramu z testovací množiny. Nakonec stačí porovnat indexy nejpodobnější dvojice a v případě jejich shody pokládat klasifikaci za úspěšnou a v případě neshody za neúspěšnou. Po provedení všech porovnání se pak celková úspěšnost klasifikace s na zvolené testovací množině vypočte jako v rovnici 3.1, kde $hits$ představuje počet úspěšně klasifikovaných obrázků a $misses$ počet nesprávně klasifikovaných obrázků.

$$s = \frac{hits}{hits + misses} \tag{3.1}$$

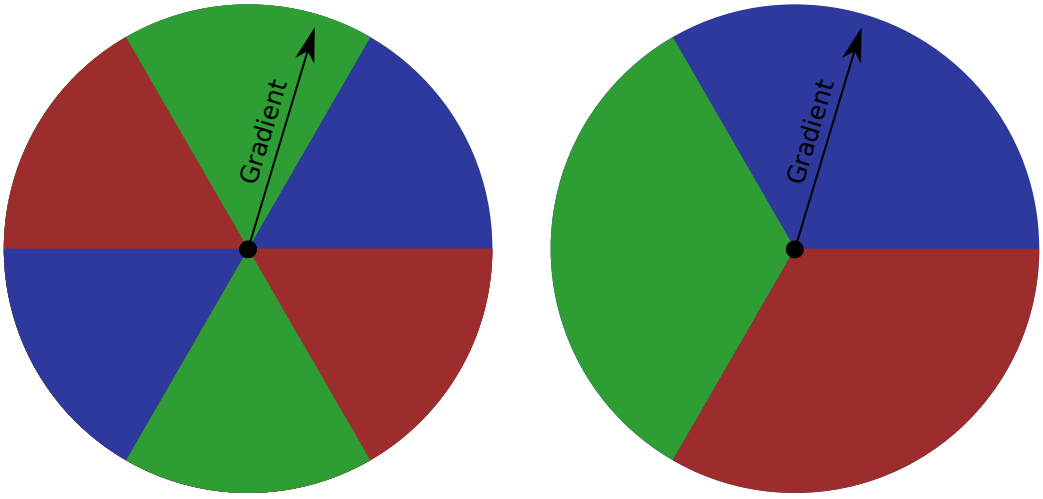
3.3.7 Extrakce příznaků

Způsob extrakce příznaků pomocí implementovaných metod byl popsán v sekci 2.2, avšak některé implementační detaily nebyly v literatuře zmíněny. V této části budou jednotlivé implementační detaily stručně popsány. Grafické znázornění implementovaných deskriptorů je v příloze A.

Způsob dělení obrázku při konstrukci sekvence histogramů HS byl implementován jiným způsobem než v [1]. Zatímco v [1] je parametrem pro rozdělení obrázku počet dělení, v případě mojí implementace je parametrem velikost strany čtverce mřížky. Tato pravidelná mřížka je zarovnána vždy na střed obrázku, pokud velikost strany čtverce mřížky celočíselně nedělí výšku nebo šířku obrázku. Na okrajích tak v případě nesoudělnosti zůstává okraj, který se do globálního histogramu nedostane. V dalším textu se bude jako strana čtverce mřížky používat pojem *grid size*.

U metody LDP byl do implementace přidán parametr na volbu pouze uniformních vzorů, který v [2] nebyl uvažován.

Pojmy znaménková a neznaménková reprezentace gradientu u metody POEM jsou znázorněny na obrázku 3.2. V článku [4] rovněž není zmíněn způsob výpočtu gradientu obrázku. Při implementaci byl přidán do metody parametr na způsob výpočtu gradientu. První způsob je aplikace filtru $(-1, 0, 1)$ na obrázek pro získání obrázku *dx* a filtru $(-1, 0, 1)^T$ pro získán obrázku *dy*. Druhý způsob je aplikace Scharrova operátoru [13] a posledním implementovaným způsobem je aplikace Sobelova operátoru [14].

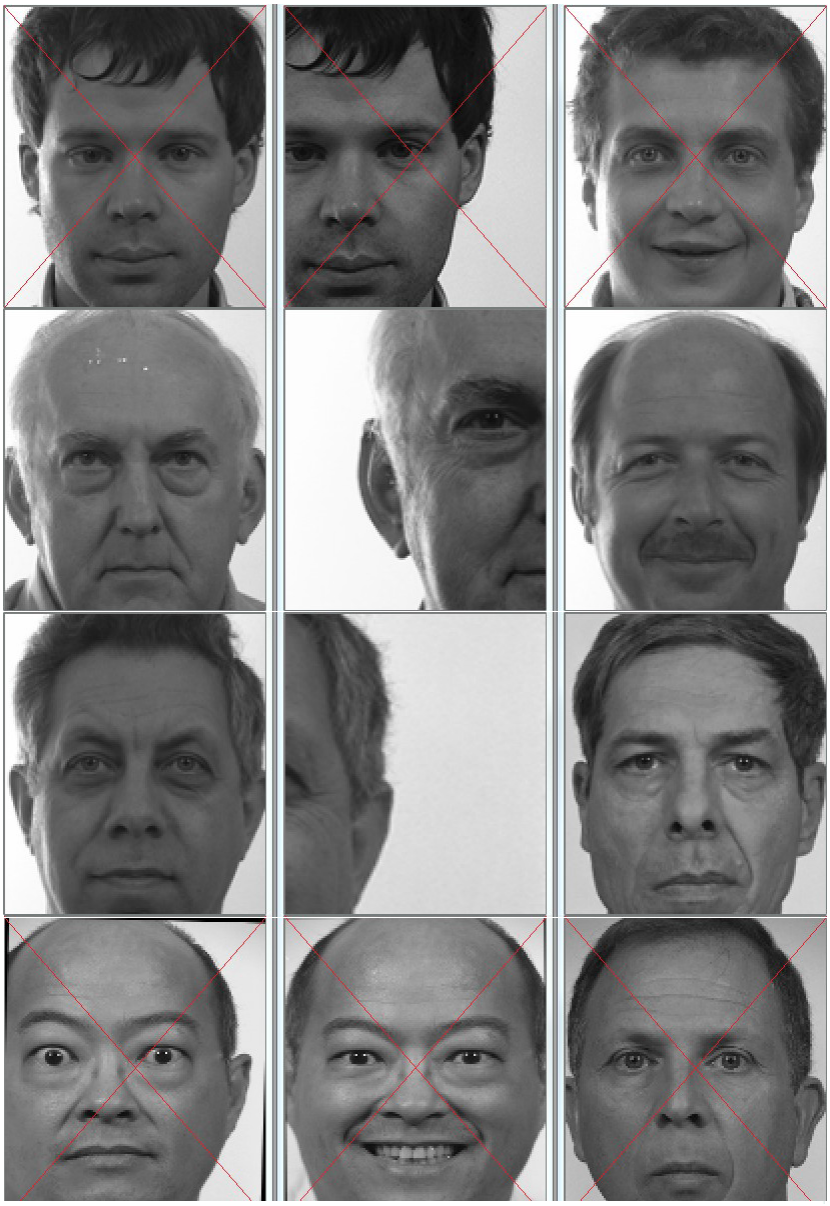


Obrázek 3.2: Grafické znázornění pojmů znaménková a neznaménková reprezentace gradientu o třech diskrétních směrech.

3.4 Chyby v testovacích datech

Během prvních testů, které byly prováděny na testovací množině *fb*, jsem kontroloval správnost implementovaných metod. Při neúspěšné klasifikaci jsem si nechal zobrazovat tři fotografie (testovanou osobu z *fa*, stejnou osobu z *fb* a nesprávně nalezenou osobu z *fb*). Zjistil jsem, že některé obličeje na fotografiích jsou špatně ořezány. Nakonec se ukázalo, že chyba nebyla na straně skriptu, který z původních fotografií na základě metadat o pozicích očí apod. vyřezává obličej, ale v samotných metadatech. O to víc překvapivé je, že v žádném z nastudovaných článků na tuto chybu autoři neupozorňují a není jasné, jestli ji nějak řeší. Na následujícím obrázku 3.3 jsou zobrazeny špatně ořezané obličeje na základě chybných metadat. První sloupec fotek představuje fotku z galerie, tedy z množiny *fa*, druhý sloupec přestavuje stejnou osobu z množiny *fb*, která měla být nalezena a třetí sloupec přestavuje osobu

z množiny *fb*, která byla ve skutečnosti nalezena. Většina z těchto nesprávně ořezaných obličejů, byla před závěrečným měřením opravena.



Obrázek 3.3: Ukázka nesprávně ořezaných obličejů na základě chybných metadat.

3.5 Ovládání

3.5.1 Parametry použitých metod

Následující výčet obsahuje seznam a popis parametrů, jejich datový typ v závorce a v některých případech i rozsah jejich možných hodnot. Parametry se do programu načítají z inicializačního souboru, jehož cesta je jediný argument programu. Tento soubor je ve formátu `<klíč>=<hodnota(y)>` na každé řádce. Před ani za znakem „=“ nesmí být mezera (např. `usedMethod=0`). V případě, že v souboru není některý parametr nastaven, si program sám zvolí defaultní hodnotu. Řádky inicializačního souboru které začínají znakem „;“ jsou programem při načítání ignorovány a lze tak do těchto řádek psát komentáře. Volitelně je možné uvést jako druhý parametr cestu k souboru, do kterého bude po provedení měření na základě parametrů z inicializačního souboru uložena naměřená úspěšnost při klasifikaci.

- **usedMethod** (int) - rozhoduje o typu obrazového deskriptoru aplikovaného na obrázky
  - 0: deskriptor LBP
  - 1: deskriptor LDP
  - 2: deskriptor POEM
- **galleryPrefixPath** (string) - prefix cesty k adresáři galerie obsahujícímu číslované podadresáře (např. `feret\gallery\s`)
- **probePrefixPath** (string) - prefix cesty k adresáři testovací množiny obsahujícímu číslované podadresáře (např. `feret\fb\s`)

- **extension** (string) - přípona souborů obrázků (např. `tif`)
- **startIndex** (int) - počáteční index adresáře při procházení galerie a testovací množiny
- **endIndex** (int) - koncový index adresáře při procházení galerie a testovací množiny
- **experimentalga** (int) - hodnota určuje, zda se má používat experimentální genetický algoritmus pro naladění vah dílčích histogramů
  - 0: nepoužívat genetický algoritmus
  - 1: používat genetický algoritmus
- **experimentalgaSymetric** (int) - hodnota určuje, zda se má symetrická nebo obecná váhová maska
  - 0: hledat obecnou masku
  - 1: hledat symetrickou masku
- **experimentalgaStop** (int) - hodnota určuje, po kolika iteracích nevedoucích ke zlepšení fitness má experimentální genetický algoritmus skončit
- **preprocessing** (int) - hodnota určuje způsob předzpracování, na rozdíl od ostatních parametrů, kde nezáleží na jejich pořadí je v případě parametru *preprocessing* možné tento parametr uvést vícekrát (na samostatné řádce) a vytvořit z nich celý řetěz, řetěz předzpracování se pak v programu provede v pořadí, jak bylo uvedeno v inicializačním souboru
  - 0: Gaussovo rozostření - parametrem je velikost filtru

  - 1: rozdíl gaussiánů (DoG) - parametrem jsou velikosti obou filtrů
  - 2: přeškálování obrázku (změna velikosti) - škálovací faktor
  - 3: gamma korekce - parametrem je hodnota gamma
  - 4: ekvalizace histogramu - nemá parametr

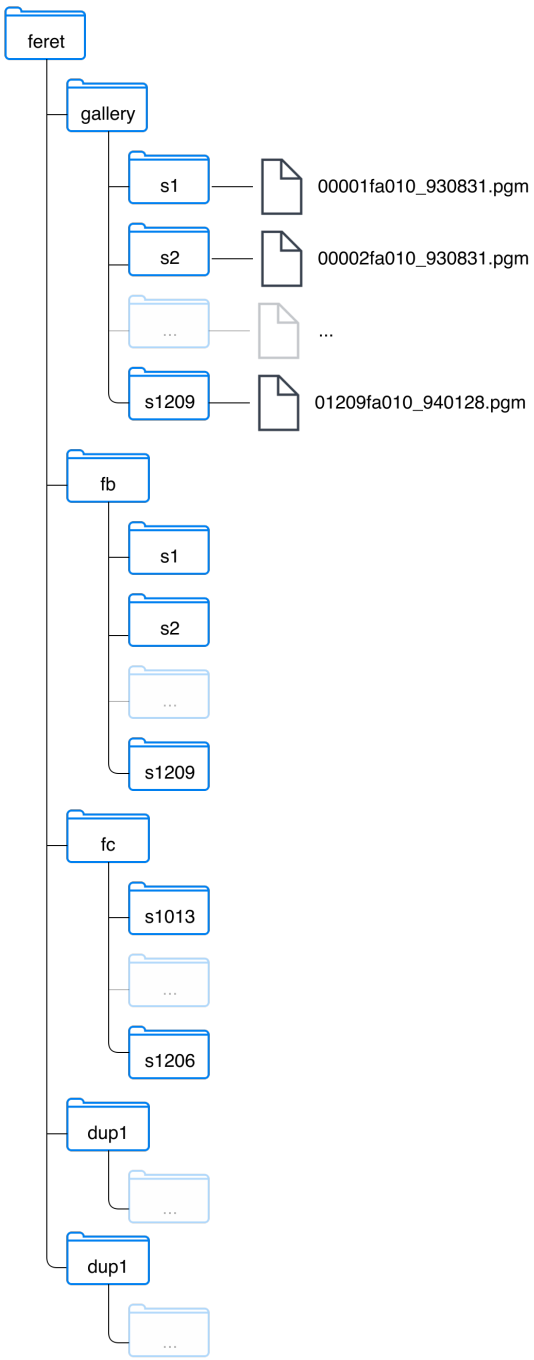
- **lbpParameters.gridSize** (int) - velikost strany čtverce v pixelech při konstrukci sekvence histogramů
- **lbpParameters.useUniformBinsOnly** (int) - rozhoduje o použití pouze uniformních vzorů bitů
  - 0: používat všech 256 vzorů
  - 1: používat pouze uniformní vzory bitů
- **ldpParameters.gridSize** (int) - velikost strany čtverce v pixelech při konstrukci sekvence histogramů
- **ldpParameters.useUniformBinsOnly** (int) - rozhoduje o použití pouze uniformních vzorů bitů
  - 0: používat všech 256 vzorů
  - 1: používat pouze uniformní vzory bitů
- **ldpParameters.order** (int) - řád metody LDP
- **poemParameters.gradientOrientationCount** (int) - počet diskrétních orientací gradientů
- **poemParameters.gradientType** (int) - určuje způsob výpočtu gradientu obrázku
  - 0: aplikace 2D filtrů $(-1, 0, 1)$ a $(-1, 0, 1)^T$

  - 1: aplikace Scharrova operátoru
  - 2: aplikace Sobelova operátoru

- **poemParameters.gradientSignedRepresentation** (int) - rozhoduje o znaménkové či neznaménkové reprezentaci gradientů
  - 0: neznaménková reprezentace (gradient na intervalu $\langle 0; \pi)$)
  - 1: znaménková reprezentace (gradient na intervalu $\langle 0; 2\pi)$)
- **poemParameters.cellSize** (int) - velikost čtvercového okolí pro výpočet histogramu směrů gradientů pro každý pixel
- **poemParameters.blockSize** (int) - poloměr kružnice při aplikování LBP operátoru
- **poemParameters.useBilinearInterpolation** (int) - rozhoduje o tom, zda se bude při aplikaci LBP operátoru používat bilineární interpolace mezi sousedními pixely
  - 0: nepoužívat bilineární interpolaci
  - 1: používat bilineární interpolaci
- **poemParameters.gridSize** (int) - velikost strany čtverce v pixelech při konstrukci sekvence histogramů
- **poemParameters.tauConst** (float) - konstanta zajišťující stabilitu v téměř konstantních oblastech obrázku
- **poemParameters.useUniformBinsOnly** (int) - rozhoduje o použití pouze uniformních vzorů bitů
  - 0: používat všech 256 vzorů
  - 1: používat pouze uniformní vzory bitů

3.5.2 Požadavky na data

Data pro program musejí dodržovat určitou adresářovou strukturu. Jednotlivé testovací množiny a galerie musí být v samostatných adresářích a prefix cesty k adresářům s obrázky musí být neměnný (v tomto případě jsou prefixy cest např. `feret\gallery\s` nebo `feret\fc\s`). Jednotlivé adresáře s obrázky mohou obsahovat i více obrázků, avšak musí platit, že každý index napříč adresářovou strukturou jednoznačně představuje jednu a tu samou osobu. Dále musejí mít všechny obrázky stejnou příponu definovanou parametrem `extension` a rozměry těchto obrázků musejí být stejné. Na samotném názvu obrázků, vyjma jejich přípony nezáleží. Obrázky obličejů by pro správnou funkčnost implementovaných metod měly být normalizované, tzn. že by měly mít zarovnané všechny obličejové části (zejména oči) a měly by mít stejné měřítko. Ukázka možné adresářové struktury je zobrazena na obrázku 3.4.



Obrázek 3.4: Příklad přípustné adresářové struktury.

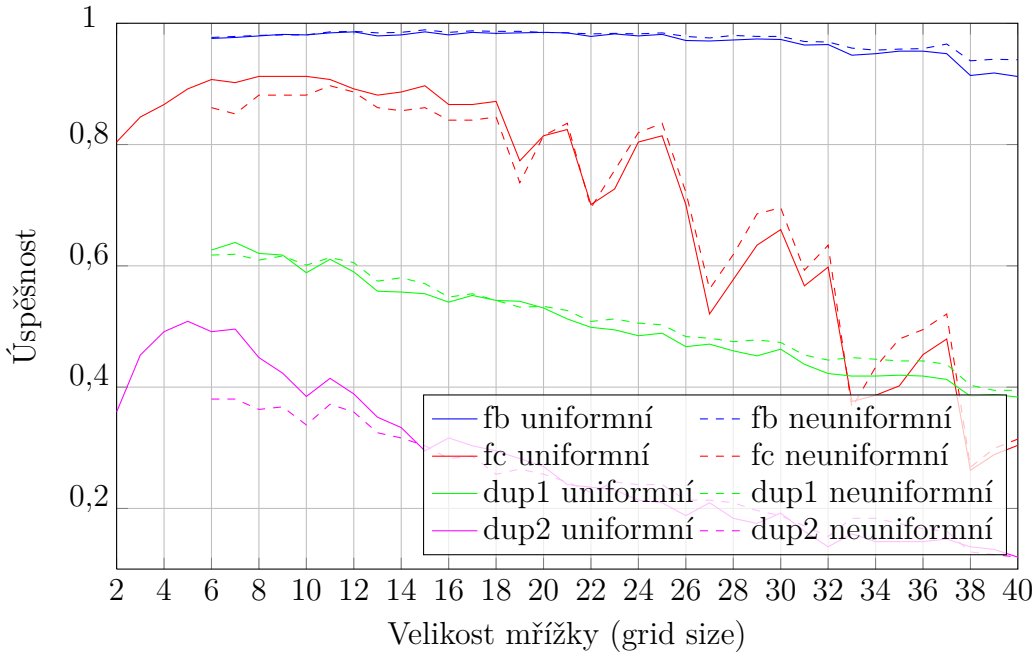
4 Dosažené výsledky

4.1 Naladění parametrů

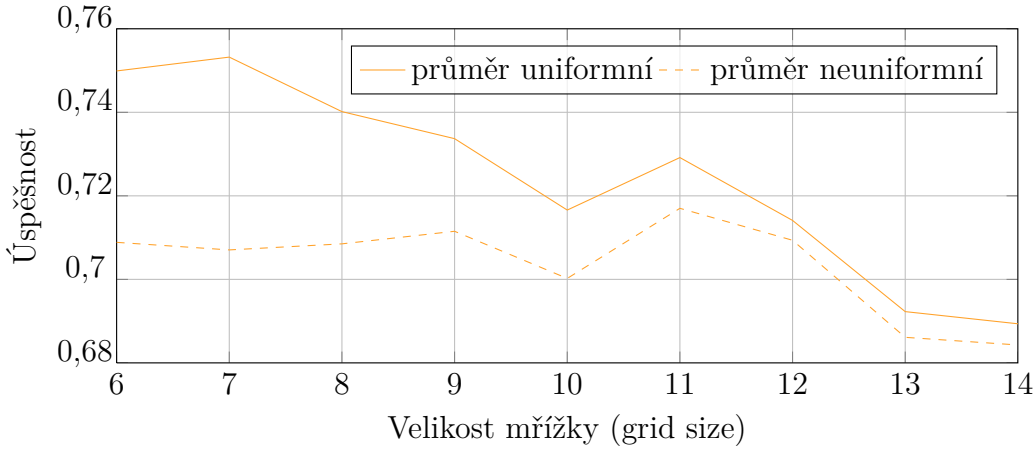
V následující části bude popsán způsob volby parametrů jednotlivých metod pro jejich závěrečné a objektivní srovnání. Pro provádění velkého množství měření úspěšnosti jednotlivých metod byl vytvořen program, který generuje inicializační soubory a společně s nimi dávkové soubory umožňující sekvenční provádění velkého množství měření s různými parametry a jejich průběžné ukládání do souborů. V případě, že v následujících obrázcích chybí část grafu, znamená to, že pro danou volbu parametrů pro zvolenou metodu překračuje výpočet maximální paměťové nároky pro 32-bitovou aplikaci, tzn. 4 GB. Za zmínku stojí, že ve Visual Studiu 2010 je defaultně nastaven limit pro paměť u 32-bitové aplikace na 2 GB a pro využití 4 GB paměti je potřeba nastavit přepínač linkeru *Enable large address* na *Yes*. Způsob ladění parametrů těchto metod se v literatuře různí. Často je naladění provedeno pouze na testovací množině *fb*. Já jsem se po konzultaci s vedoucím práce rozhodl pro výběr nejlepších parametrů jednotlivých metod průměrováním výsledků na všech testovacích množinách. Jako optimální parametry metody budou tedy považovány takové, na kterých bude naměřena nejvyšší vážená průměrná úspěšnost na všech testovacích množinách z databáze FERET. Jako váha jednotlivých množin bude sloužit počet fotografií v dané množině.

4.1.1 Naladění parametrů LBP

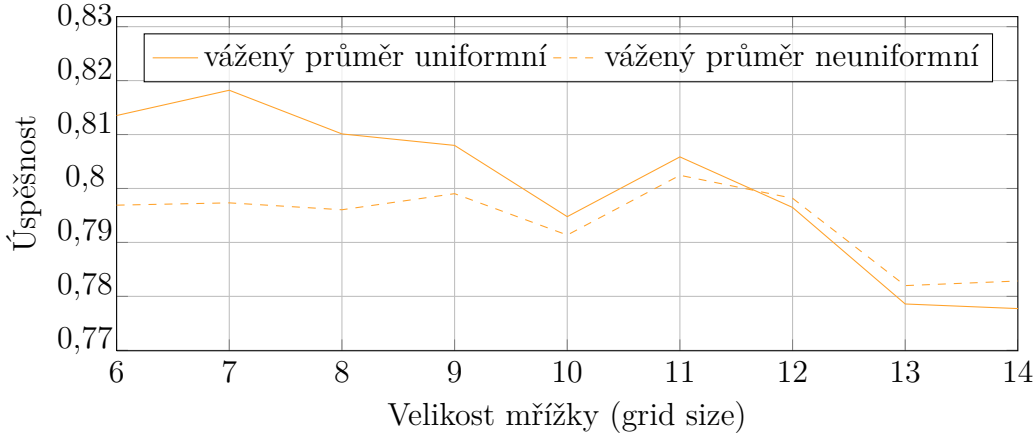
Naladění parametrů LBP je ze všech tří metod nejjednodušší, protože obsahuje pouze dva volitelné parametry, jelikož byla implementována pouze základní, nikoli zobecněná kruhová verze LBP, kde by přibyly ještě dva další parametry. Prvním parametrem je velikost mřížky `lbpParameters.gridSize`. Druhým je přepínač na používání pouze uniformních vzorů `lbpParameters.useUniformBinsOnly`. Další parametry jako např. předzpracování a způsob měření podobnosti histogramů je společný pro všechny tři metody a jejich význam bude zhodnocen až po naladění základních parametrů. Vliv parametrů na úspěšnost při rozpoznávání obličejů na všech čtyřech testovacích množinách zobrazuje obrázek 4.1.



Obrázek 4.1: Grafické znázornění úspěšnosti základní metody LBP na všech testovacích množinách v závislosti na velikosti mřížky a uniformní/neuniformní variantě.



Obrázek 4.2: Aritmetický průměr nad všemi testovacími množinami při použití metody LBP.

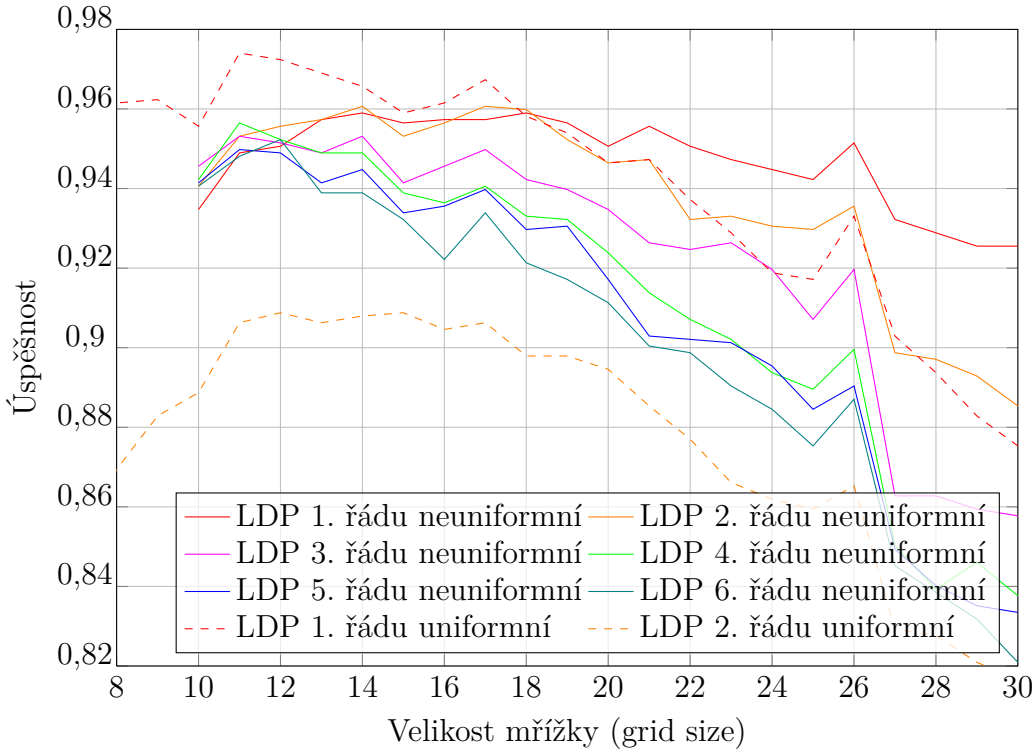


Obrázek 4.3: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody LBP.

Z tabulek na obrázcích 4.2 a 4.3 vyplývá, že optimální je použít uniformní verzi metody LBP s velikostí mřížky 7 na testovacích fotografiích o rozměrech 130×150 pixelů.

4.1.2 Naladění parametrů LDP

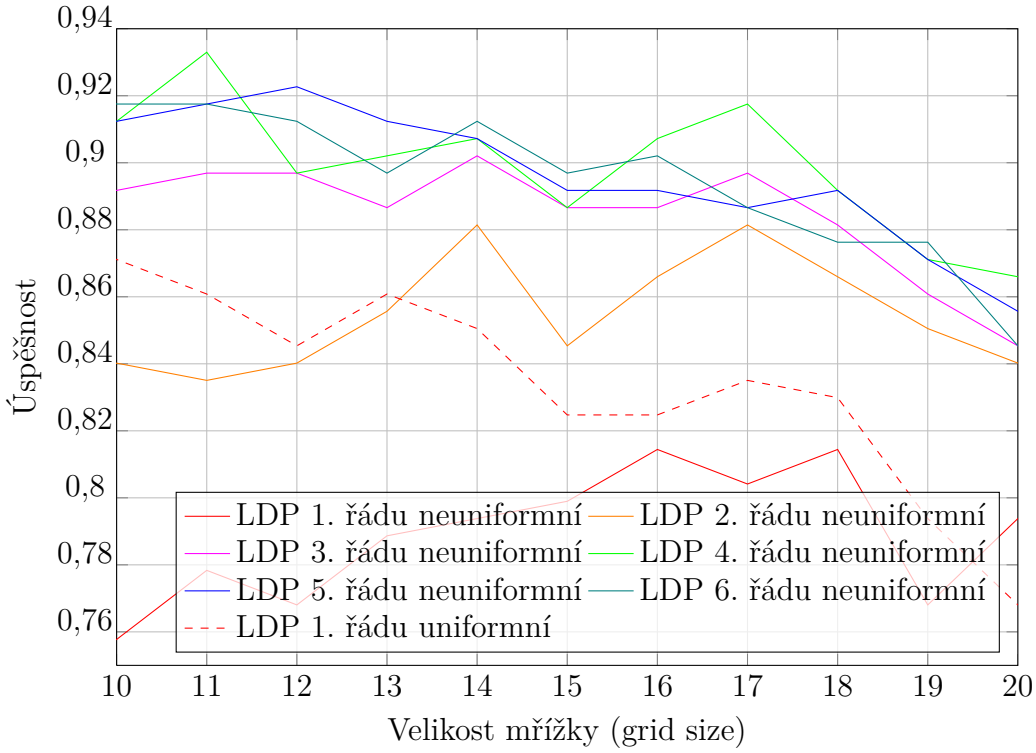
U LDP je třeba naladit stejné parametry jako u LBP ještě navíc parametr `ldpParameters.order`.



Obrázek 4.4: Grafické znázornění úspěšnosti metody LDP na testovací množině *fb* v závislosti na velikosti mřížky, uniformní/neuniformní variantě a řádu metody.

Zvyšování řádu u uniformního LDP na testovací množině *fb* výrazně snižuje úspěšnost při rozpoznávání a v grafu 4.4 proto nejsou zaneseny.

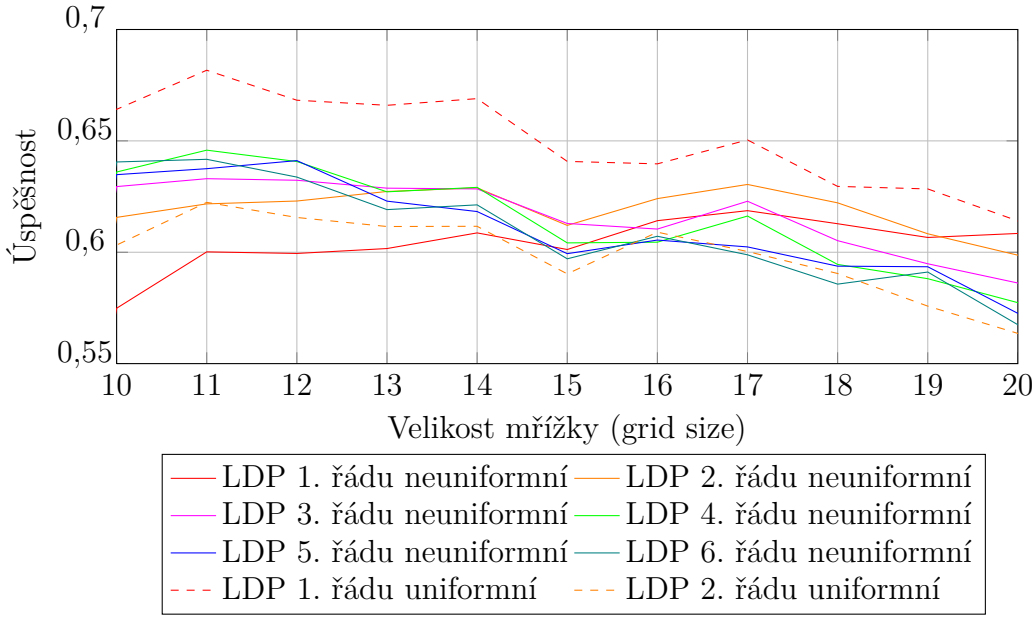
Parametr pro volbu uniformní/neuniformní varianty v původním článku [2] není uveden. Do ladění parametrů byl ale pro zajímavost zařazen, protože vykazuje poměrně zajímavé výsledky.



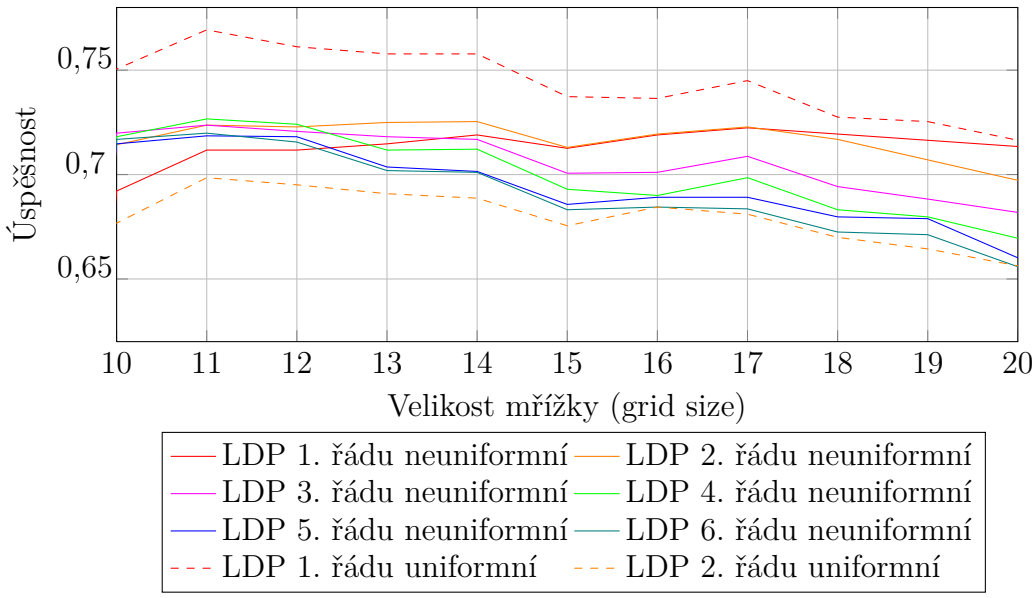
Obrázek 4.5: Grafické znázornění úspěšnosti metody LDP na testovací množině *fc* v závislosti na velikosti mřížky, uniformní/neuniformní variantě a řádu metody.

Z grafu 4.4 je patrné, že použití uniformních vzorů u metody LDP výrazně snižuje úspěšnost při klasifikaci s výjimkou LDP 1. řádu, kde použití uniformního vzoru, zde na testovací množině *fb*, úspěšnost naopak zvyšuje.

Před znázorněním průměrných hodnot je ještě pro zajímavost na obrázku 4.5 ukázáno měření na testovací množině *fc*. Zatímco u množiny *fb* u neuniformní varianty řády 1 a 2 poměrně dominují nad ostatními, na množině *fc* naopak zcela propadají a do popředí se dostávají řády 4 a 5.



Obrázek 4.6: Aritmetický průměr nad všemi testovacími množinami při použití metody LDP.



Obrázek 4.7: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody LDP.

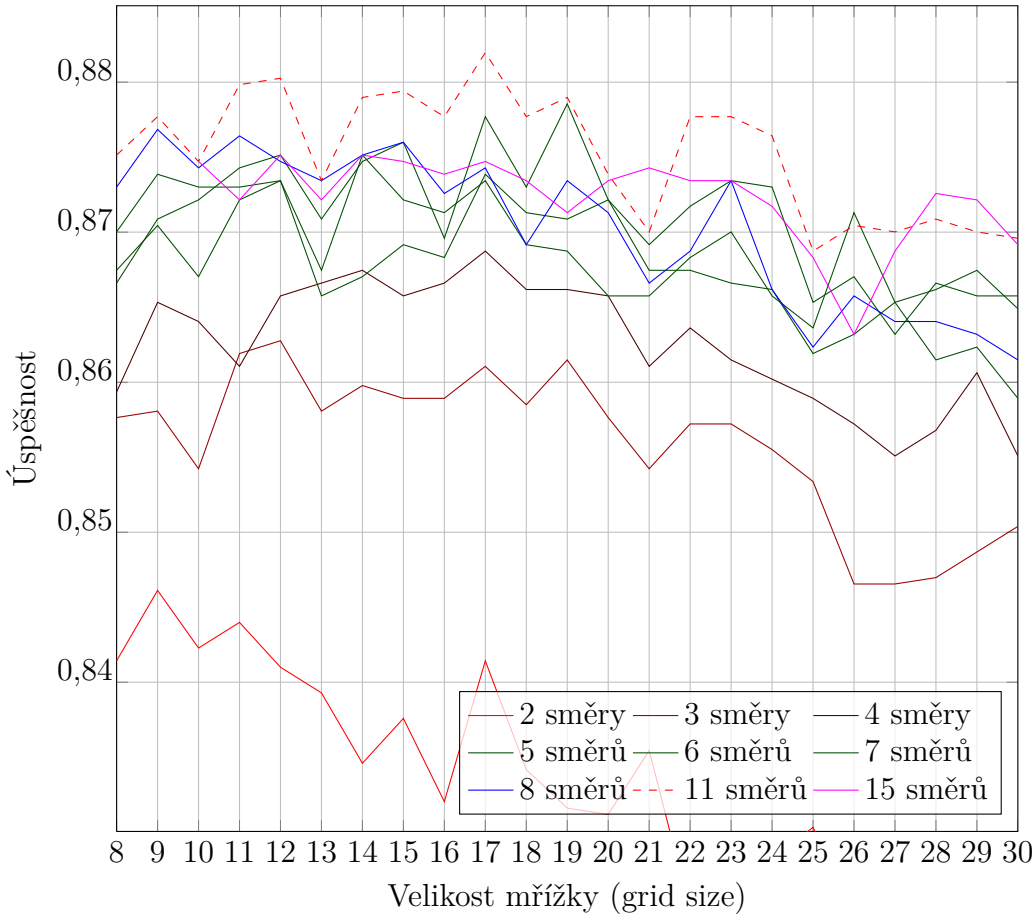
V dalším textu a grafech bude metoda LDP s naladěnými parametry uniformní verze prvního řádu s velikostí mřížky 11 značena jako „LDP“ a neuniformní verze druhého řádu s velikostí mřížky 17 jako „LDP2“.

4.1.3 Naladění parametrů POEM

Naladění metody POEM je ze všech implementovaných metod nejsložitější, protože obsahuje nejvíce parametrů. Provedení měření nad všemi kombinacemi parametrů je z časových důvodů téměř nemožné, proto byla při ladění použita určitá forma heuristiky. Počáteční nastavení parametrů se bude inspirovat hodnotami parametrů vypočtenými v článku [4].

První měření má za cíl určit vhodný počet diskrétních směrů v závislosti na velikosti mřížky (grid size). Ostatní parametry jsou zvoleny podobně jako v článku [4]. Z prvních měřeních vyplynulo, že je lepší použít neznaménkovou verzi diskrétních gradientů.

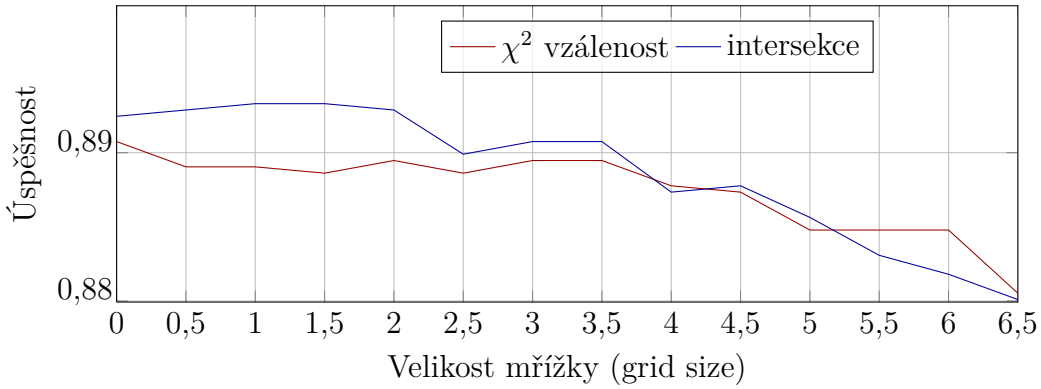
V grafu na obrázku 4.8 jsou celkem zajímavé výsledky, které jsou částečně v rozporu s výsledky prezentovanými v článku [4]. Z naměřených výsledků nejlépe vychází volba 11 diskrétních směrů gradientů. Pro přehlednost byly v grafu na obrázku 4.8 vynechány některé hodnoty parametru `poemParameters.gradientOrientationCount`, které však byly méně úspěšné, než pro hodnotu 11.



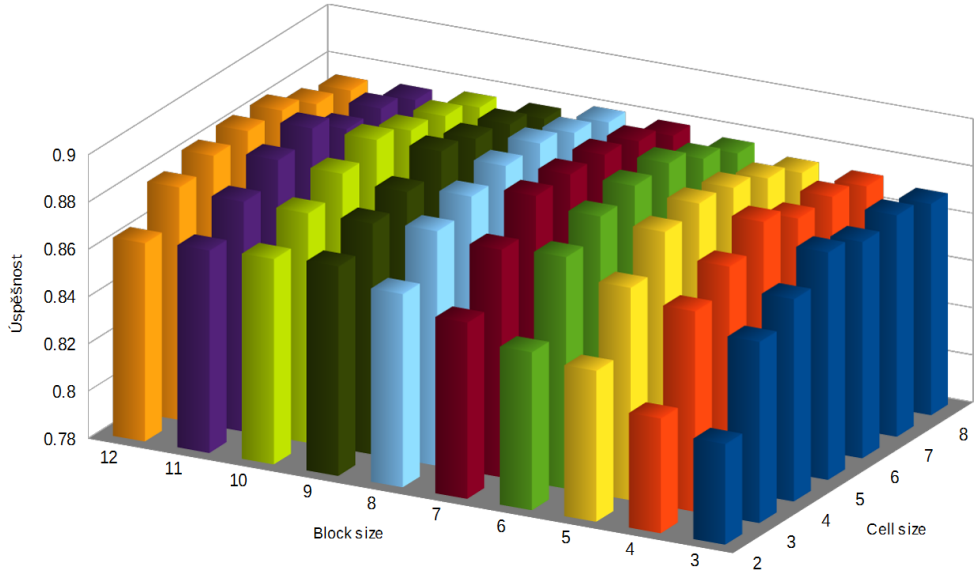
Obrázek 4.8: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody POEM a různým počtem diskrétních směrů gradientů.

Následující parametry byly nastaveny pro měření v grafech na obrázcích 4.8 a 4.9:

```
poemParameters.gradientOrientationCount=<2 - 15>
poemParameters.gradientSignedRepresentation=0
poemParameters.cellSize=7
poemParameters.blockSize=10
poemParameters.useBilinearInterpolation=0
poemParameters.gridSize=<8 - 30>
poemParameters.tauConst=0.0
poemParameters.gradientType=0
poemParameters.useUniformBinsOnly=1
```



Obrázek 4.9: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody POEM, různou konstantou τ a dvěma různými metrikami podobnosti histogramů.

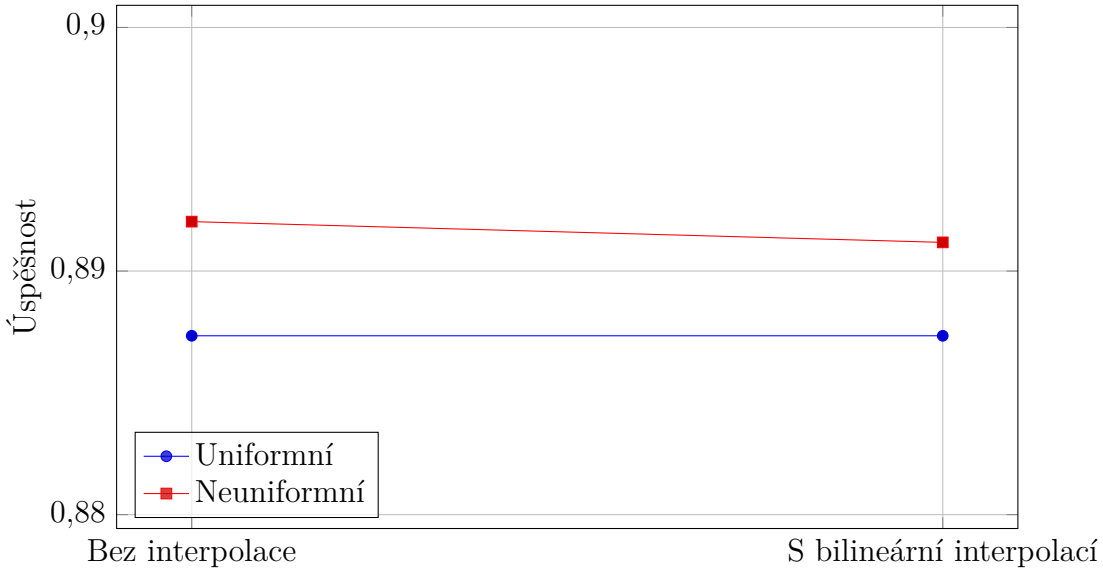


Obrázek 4.10: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody POEM pro kombinace parametrů „cell size“ a „block size“.

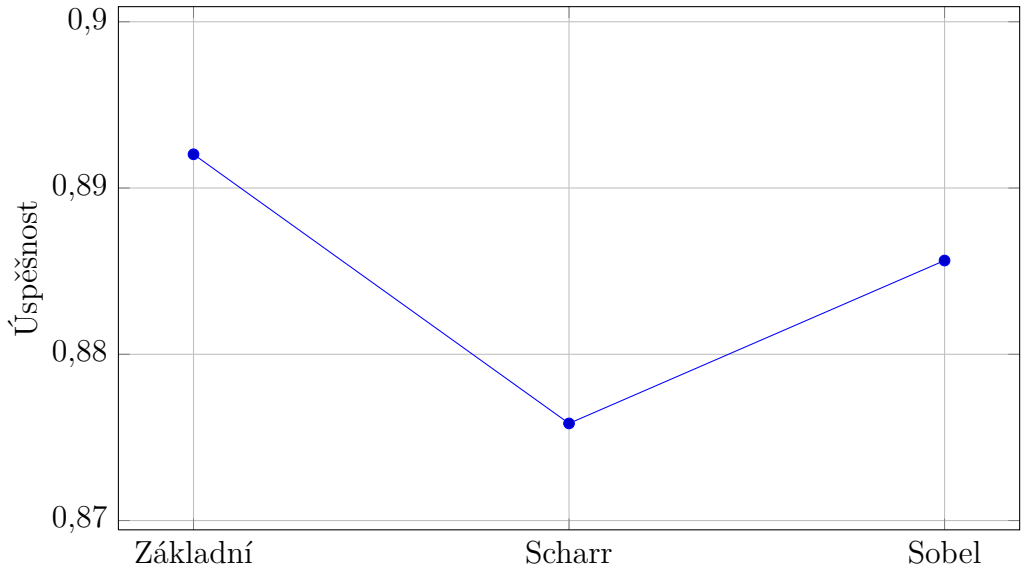
V grafu na obrázku 4.9 je znázorněn vliv hodnoty parametru `poemParameters.tauConst` na výslednou úspěšnost při rozpoznávání. Při

měření byly použity dvě různé metriky podobnosti histogramů. V případě intersekce je optimální hodnota 1 a v případě použití metriky χ^2 je optimální hodnota 0. Vyšší hodnoty tohoto parametru již vedou ke snížení úspěšnosti.

V grafu na obrázku 4.10 je znázorněn vliv kombinací hodnot parametrů `poemParameters.blockSize` a `poemParameters.cellSize`. Při měření byl nastaven parametr `poemParameters.gridSize` na hodnotu 17 a diskrétní počet směrů `poemParameters.gradientOrientationCount` na hodnotu 11. Další parametry byly nastaveny stejně, jako při měření na obrázku 4.8. Maxima je dosaženo při volbě parametrů `poemParameters.blockSize=6` a `poemParameters.cellSize=6`.



Obrázek 4.11: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody POEM pro kombinace parametrů bilineární interpolace a uniformní/neuniformní varianty.



Obrázek 4.12: Vážený aritmetický průměr nad všemi testovacími množinami při použití metody POEM pro různý způsob výpočtu gradientu obrázku.

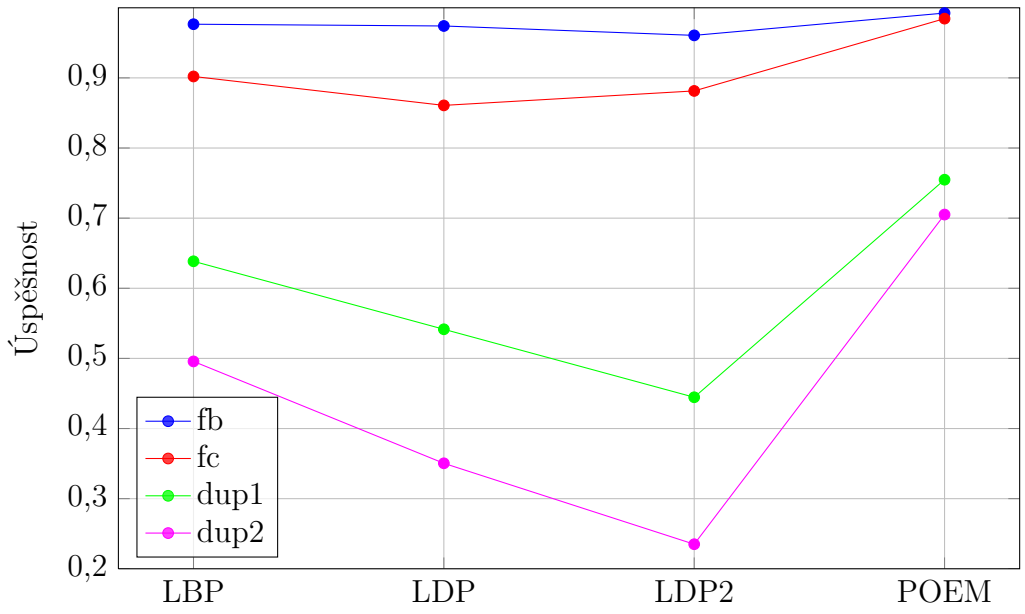
Následující parametry na základě měření představují optimální nastavení parametrů metody POEM na testovacích datech:

```
poemParameters.gradientOrientationCount=11
poemParameters.gradientSignedRepresentation=0
poemParameters.cellSize=6
poemParameters.blockSize=6
poemParameters.useBilinearInterpolation=0
poemParameters.gridSize=17
poemParameters.tauConst=0.0
poemParameters.gradientType=0
poemParameters.useUniformBinsOnly=0
```

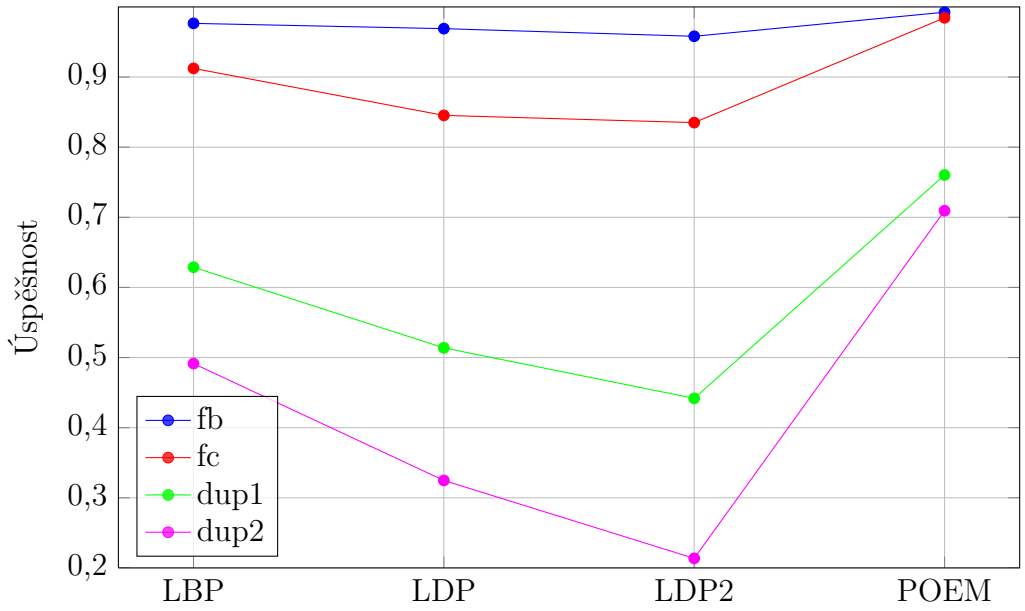
4.2 Srovnání úspěšnosti metod

V této části budou porovnány metody mezi sebou s nejlepšími naměřenými parametry. V grafu na obrázku 4.13 jsou znázorněny výsledky při použití metriky χ^2 vzdálenost. V grafu na obrázku 4.14 jsou znázorněny výsledky při použití intersekce histogramů a na obrázku 4.15 jsou znázorněny výsledky při použití eukleidovské vzdálenosti.

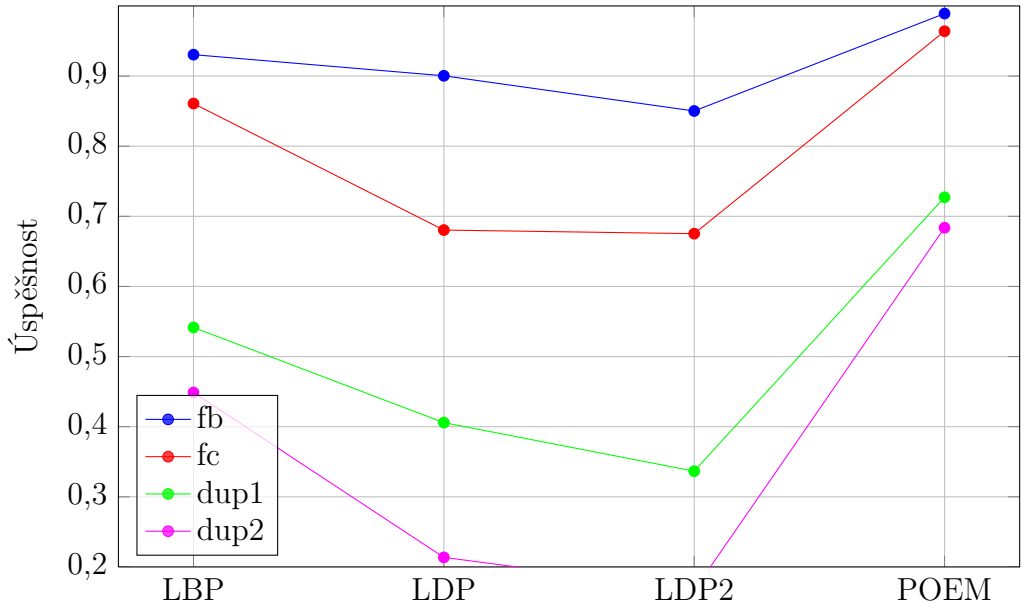
Z těchto grafů je zřejmé, že metoda POEM má na testovacích datech jednoznačně nejvyšší úspěšnost. Zároveň je velice robustní vůči použité metrice a ani použití eukleidovké vzdálenosti jako metriky na metodě POEM nevede k výraznému snížení úspěšnost na rozdíl od ostatních metod. Na druhé straně je metoda LDP jednoznačně nejhorší a to i oproti základní verzi metody LBP a vykazuje velmi nevyrovnané výsledky při různých parametrech.



Obrázek 4.13: Srovnání úspěšnosti jednotlivých metod s použitou metrikou χ^2 vzdálenost.



Obrázek 4.14: Srovnání úspěšnosti jednotlivých metod s použitou metrikou intersekce histogramů.



Obrázek 4.15: Srovnání úspěšnosti jednotlivých metod s použitou metrikou eukleidovké vzdálenosti.

V tabulce 4.1 jsou číselné hodnoty úspěšností metod vlastní implementace a pro srovnání také výsledky prezentované v literatuře.

V případě metody LBP jsou výsledky vlastní implementace a výsledky prezentované v literatuře srovnatelné v výjimkou množiny *fc*, kde je úspěšnost při vlastní implementaci výrazně vyšší.

V případě metody LDP jsou výsledky vlastní implementace srovnatelné na testovacích množinách *fb* a *fc* i když jsou v případě vlastní implementace o něco lepší, ovšem v případě testovacích množin *dup1* a *dup2* se již poměrně dost odlišují a jsou v případě vlastní implementace výrazně horší než výsledky uváděné v článku [2]. Vzhledem k tomu, že na rozdíl od množin *fb* a *fc* obsahují množiny *dup1* a *dup2* často více fotografií od stejné osoby, výsledky naznačují, že byly v článku [2] z množin *dup1* a *dup2* vybrány pouze některé fotografie, ale já jsem vždy pracoval s celou množinou. O to více je překvapující, že při implementaci metody LPB v článku [2], jsou prezentované výsledky na množinách *dup1* a *dup2* ještě horší, než u vlastní implementace a než výsledky uvedené v článku [1].

V případě metody POEM jsou výsledky vlastní implementace téměř totožné s výsledky uvedenými v článku [4]. Mírné rozdíly jsou na množinách *dup1* a *dup2*. Pro srovnání jsou kromě vlastnoručně nalezených parametrů uvedeny i výsledky při použití parametrů uvedených v článku [4], které jsou velmi odlišné. Z výsledků je zřejmé, že metoda POEM je velmi robustní i vůči svým parametrům.

Metoda	fb	fc	dup1	dup2
LBP [1]	0.93	0.51	0.61	0.50
LBP [2]	0.91	0.63	0.53	0.38
LBP vlastní implementace	0.977	0.902	0.639	0.496
LDP 2. řádu [2]	0.93	0.82	0.61	0.52
LDP 3. řádu [2]	0.90	0.88	0.63	0.61
LDP 4. řádu [2]	0.86	0.80	0.55	0.52
LDP vlastní implementace[a]	0.961	0.881	0.445	0.235
LDP vlastní implementace[b]	0.974	0.860	0.542	0.350
LDP vlastní implementace[c]	0.956	0.933	0.446	0.248
LDP vlastní implementace[d]	0.953	0.897	0.460	0.222
POEM [4]	0.976	0.960	0.778	0.765
POEM vlastní implementace	0.992	0.985	0.760	0.709
POEM vlastní implementace[e]	0.989	0.948	0.706	0.632

Tabulka 4.1: Srovnání naměřených výsledků a výsledků prezentovaných v literatuře.

[a] Neuniformní veze LDP 2. řádu s velikostí mřížky (grid size) 17.

[b] Uniformní veze LDP 1. řádu.

[c] Neuniformní veze LDP 3. řádu s velikostí mřížky (grid size) 11.

[d] Neuniformní veze LDP 4. řádu s velikostí mřížky (grid size) 11.

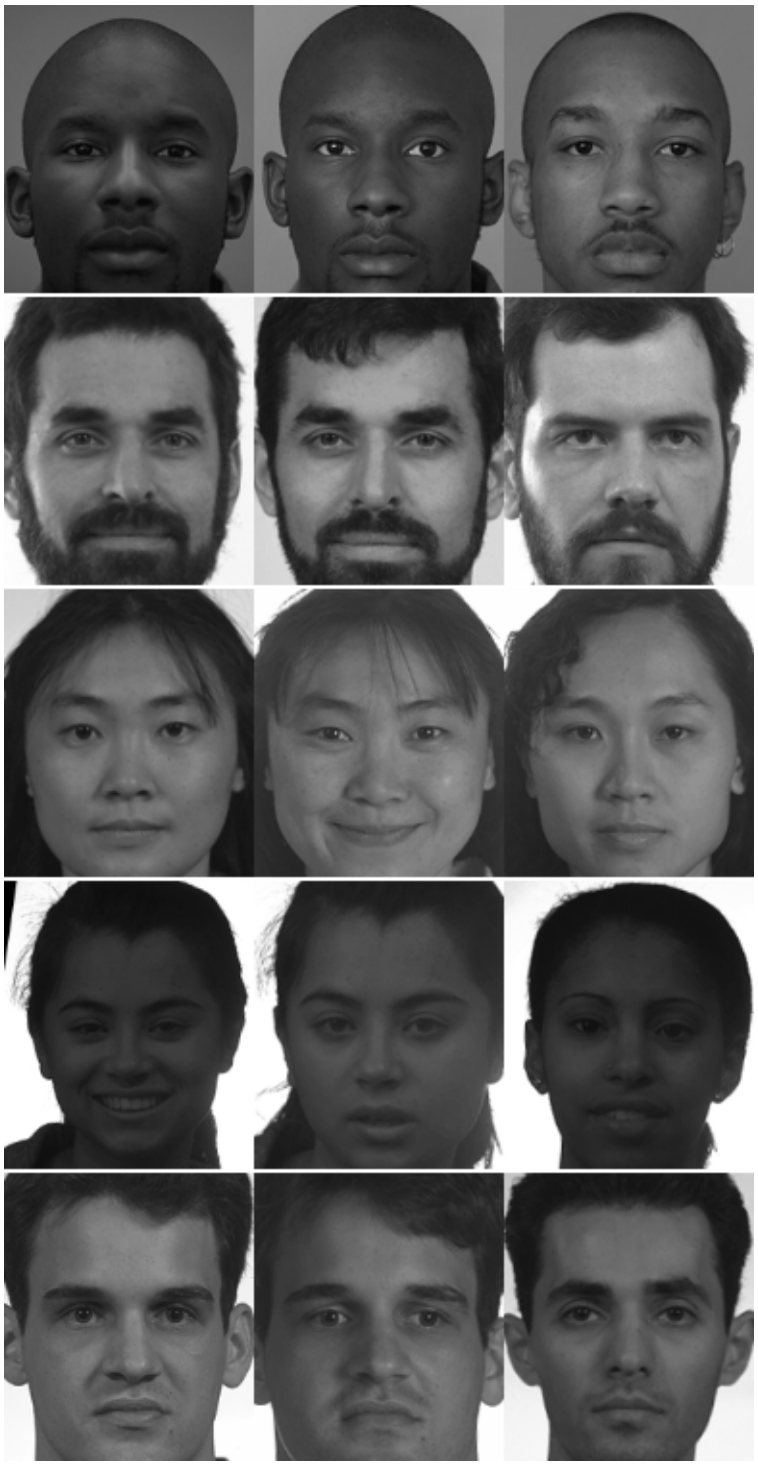
[e] Parametry byly zvoleny stejné, jako v článku [4].

4.3 Problematické obrázky

V následující části budou ukázány některé problematické obrázky z databáze FERET. Vždy se jedná o obrázky, u nichž se nezdařila klasifikace pomocí metody POEM. První z trojice obličejů v řádce je vždy ten, který se klasifikuje, druhý obličej je obličej z galerie, který by měl být prvnímu obličeji nejpodobnější a měl být tedy nalezen a třetí obličej je ten, který byl ve skutečnosti nalezen (podle metody POEM byl nejpodobnější tomu prvnímu).

Nejčastější problémy při klasifikaci obličejů jsou zřejmě způsobeny odlišným mimickým výrazem na fotografiích v testovací množině a v galerii, zejména pak výrazným úsměvem, viz obrázek 4.16. Problémy způsobují rovněž fotografie pořízené pod mírným úhlem. Dále pak odlišné oříznutí obličeje v testovací množině a v galerii, např. na jedné je vidět celá horní část hlavy, ale na druhé je oříznutá apod. V neposlední řadě jsou pak špatné klasifikace způsobeny tím, že někteří lidé v databázi FERET jsou si podobní a s jejich klasifikací by dle mého názoru měl problémy i člověk, viz 4.17. U některých obrázků jsou tyto potenciální problematické části označeny červeně.

Obrázek 4.16: Příklady nesprávně klasifikovaných obličejů metodou POEM, pravděpodobně v důsledku odlišného výrazu v obličeji.



Obrázek 4.17: Příklady nesprávně klasifikovaných obličejů metodou POEM na základě podobnosti osob a odlišného oříznutí obličeje.

4.4 Doby běhů

V této části budou prezentovány doby výpočtu jednotlivých metod v milisekundách s různými parametry, respektive doby výpočtu samotného deskriptoru (extrakce příznaků) v tabulce 4.2 a následné klasifikace na základě těchto příznaků v tabulce 4.3. Měření probíhalo na osobním počítači s čtyřjádrovým procesorem Intel® Core™ i5 2.67GHz s 8 GB RAM.

Metoda / Grid size	**11**	**17**	**30**
LBP neuniformní	1.85	1.41	1.15
LDP 1. řádu neuniformní	6.54	4.67	3.99
LDP 3. řádu neuniformní	8.28	6.37	5.69
POEM neuniformní (3 směry)	18.55	17.00	16.45
POEM neuniformní (8 směrů)	48.48	44.02	42.60
POEM neuniformní (11 směrů)	-	61.03	58.76
LBP uniformní	1.38	1.19	1.08
LDP 1. řádu uniformní	4.69	3.96	3.81
POEM uniformní (3 směry)	17.07	16.59	16.64
POEM uniformní (8 směrů)	43.62	42.27	42.99
POEM uniformní (11 směrů)	59.91	58.02	57.55

Tabulka 4.2: Průměrná doba výpočtu obrazového deskriptoru jednoho obrázku o rozměrech 130×150 pixelů v milisekundách.

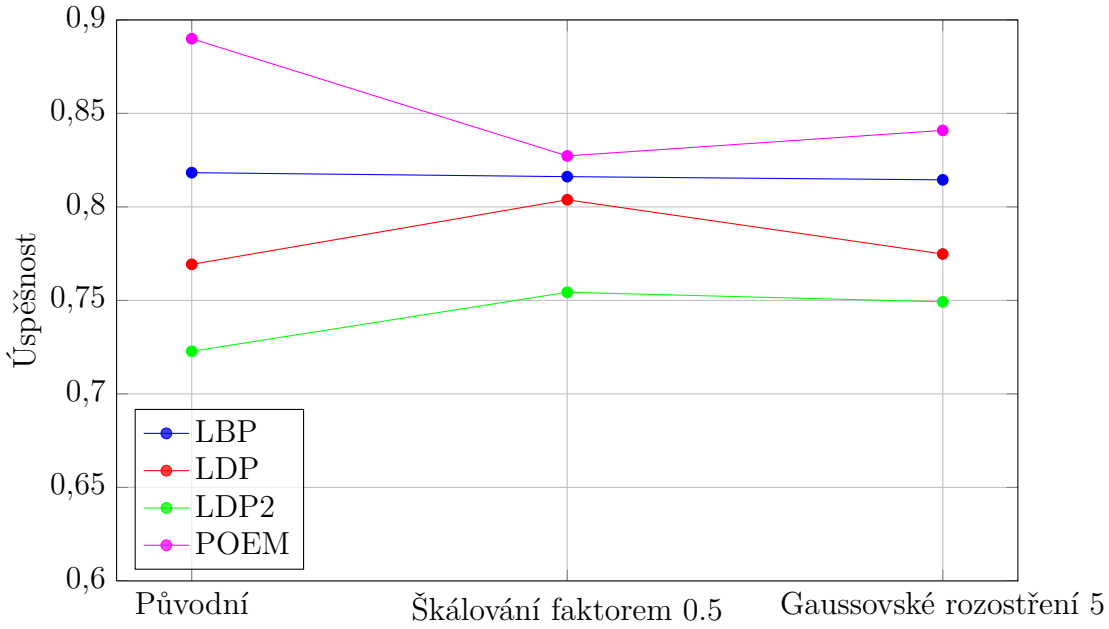
Metoda / Grid size	11	17	30
LBP neuniformní	263	119	43
LDP 1. řádu neuniformní	1169	497	169
LDP 3. řádu neuniformní	1210	499	170
POEM neuniformní (3 směry)	561	284	126
POEM neuniformní (8 směrů)	1611	815	343
POEM neuniformní (11 směrů)	-	1144	475
LBP uniformní	75	29	12
LDP 1. řádu uniformní	311	110	43
POEM uniformní (3 směry)	215	89	33
POEM uniformní (8 směrů)	578	235	88
POEM uniformní (11 směrů)	796	318	121

Tabulka 4.3: Průměrná doba klasifikace jednoho obrázku o rozměrech 130×150 pixelů do tříd z množiny *fa* v milisekundách.

4.5 Předzpracování

V této části jsou prezentovány výsledky při použití některých metod předzpracování, respektive jejich vliv na úspěšnost klasifikace. V grafu na obrázku 4.18 jsou prezentovány výsledky bez a s předzpracováním. Úspěšnost v tomto grafu představuje vážený aritmetický průměr ze všech testovacích množin. Při škálování byl u jednotlivých metod parametr `gridSize` snížen na polovinu a zaokrouhlen nahoru. Zatímco předzpracování u metody LBP nemá na úspěšnost téměř žádný vliv, tak u metody POEM dochází k výraznému zhoršení a u metody LDP naopak k výraznému zlepšení při škálování s koeficientem 0.5. Aplikování Gaussovského rozostření způsobuje

většinou spíše zhoršení, pouze na množině *fb* u metod LBP a POEM dochází k mírnému zlepšení.



Obrázek 4.18: Vliv předzpracování na celkovou úspěšnost metod.

Metoda	fb	fc	dup1	dup2
LBP bez předzpracování	0.977	0.902	0.639	0.496
LBP škálování faktorem 0.5	0.978	0.881	0.644	0.466
LBP Gaussovo rozostření 5	0.982	0.866	0.644	0.444
LDP bez předzpracování	0.974	0.861	0.542	0.350
LDP škálování faktorem 0.5	0.982	0.876	0.602	0.453
LDP Gaussovo rozostření 5	0.971	0.851	0.566	0.355
LDP2 bez předzpracování	0.961	0.881	0.445	0.235
LDP2 škálování faktorem 0.5	0.962	0.835	0.544	0.274
LDP2 Gaussovo rozostření 5	0.956	0.866	0.518	0.312
POEM bez předzpracování	0.992	0.985	0.755	0.705
POEM škálování faktorem 0.5	0.985	0.902	0.645	0.521
POEM Gaussovo rozostření 5	0.993	0.881	0.676	0.538

Tabulka 4.4: Úspěšnost metod při rozpoznávání s aplikovaným předzpracováním.

4.6 Experimentální použití genetického algoritmu

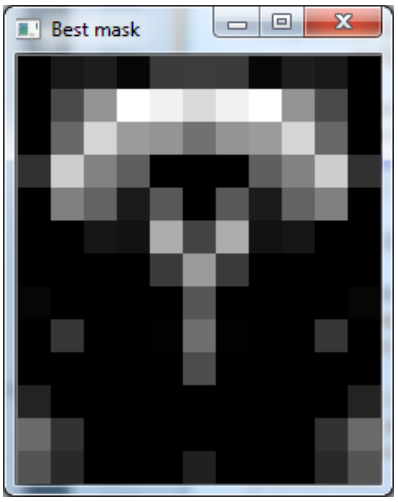
V části 2.4.2 je popisováno manuální nastavování vah k dílčím histogramům v závislosti na jejich důležitosti při rozpoznávání obličejů. Zde je prezentován experimentální genetický algoritmus sloužící k automatickému naladění těchto vah na testovací množině. Tento doprovodný experiment k práci má značný potenciál pro hlubší zkoumání a další testování. Generalizační schopnost algoritmu je v současné podobě diskutabilní, nicméně je s jeho pomocí

možné například identifikovat problematické části obrázků, které jsou na prezentovaných obrázcích tmavé.

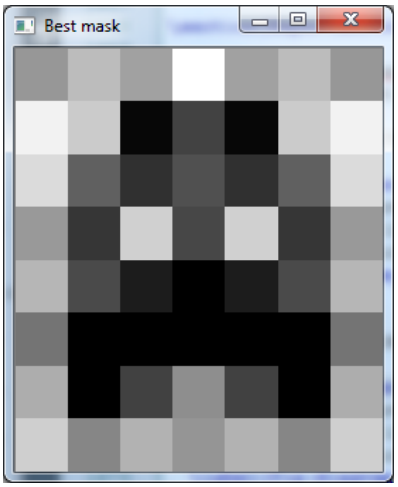
Jako fitness funkce pro genetický algoritmus byla zvolena funkce v rovnici 4.1, kde *hits* představuje počet správně klasifikovaných obličejů, *misses* představuje počet chybně klasifikovaných obličejů, *classDistance* představuje průměrnou vzdálenost všech obličejů stejných osob a *nonClassDistance* představuje průměrnou vzdálenost všech obličejů různých osob.

$$fitness = 10000000 * \frac{hits}{hits + misses} + \frac{nonClassDistance}{classDistance} \tag{4.1}$$

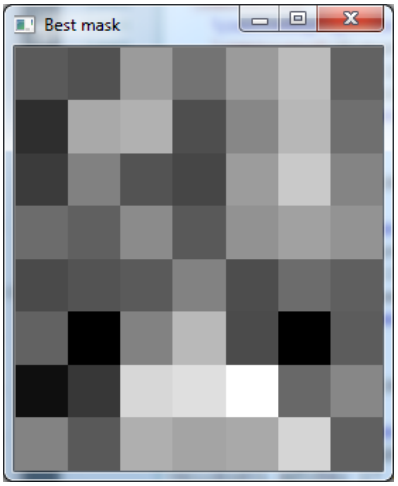
Obrázky v této části zobrazují některé nalezené masky pomocí tohoto algoritmu a jejich v jejich popisu jsou uvedeny použité parametry a zlepšení při použití této masky.



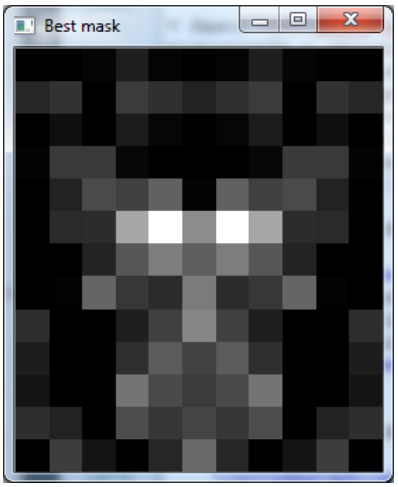
Obrázek 4.19: Naladění symetrické váhové masky při použití uniformní metody LBP s velikostí mřížky 11 na testovací množině *fb*. Z původních 20 špatně klasifikovaných obličejů bylo po aplikování masky špatně klasifikováno pouze 8.



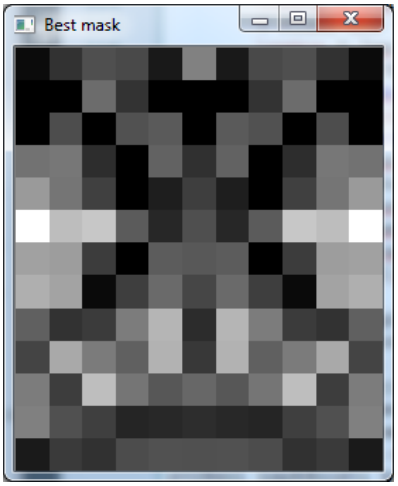
Obrázek 4.20: Naladění symetrické váhové masky při použití uniformní metody LBP s velikostí mřížky 17 na testovací množině *fc*. Z původní úspěšnosti 87% bylo po aplikování masky dosaženo 95% úspěšnosti.



Obrázek 4.21: Naladění obecné váhové masky při použití uniformní metody POEM s velikostí mřížky 17 o třech diskrétních směrech na testovací množině *dup1*. Z původní úspěšnosti 74% bylo po aplikování masky dosaženo 77% úspěšnosti.



Obrázek 4.22: Naladění symetrické váhové masky při použití uniformní metody POEM s velikostí mřížky 11 o třech diskrétních směrech množině *dup2*. Z původní úspěšnosti 72% bylo po aplikování masky dosaženo 81% úspěšnosti.



Obrázek 4.23: Naladění symetrické váhové masky při použití uniformní metody POEM s velikostí mřížky 11 o třech diskrétních směrech na testovací množině *dup2*. Z původní úspěšnosti 54% bylo po aplikování masky dosaženo 61% úspěšnosti.

4.7 Shrnutí vlastností metod

Metoda LBP je ze všech tří implementovaných metod nejjednodušší a zároveň výpočetně nejrychlejší. Doba výpočtu příznaků jednoho obrázku o rozměrech 130×150 pixelů trvá v řádu jednotek milisekund v závislosti na velikosti mřížky a doba klasifikace je řádově v desítkách milisekund při zařazování do cca 1200 klasifikačních tříd. Parametry této metody jsou ze všech implementovaných metod nejsnadněji naladitelné. Metodu LBP je možné rovněž rozšířit o její zobecněnou kruhovou formu a je tedy škálovatelná na různě velké obrázky. Bez problémů se dá použít její uniformní varianta, která úspěšnost při rozpoznávání obličejů většinou ještě zvyšuje, zároveň snižuje dobu extrakce příznaků i klasifikace a snižuje paměťovou náročnost.

Metoda LDP je oproti LBP pomalejší a paměťově náročnější. Doba výpočtu příznaků jednoho obrázku o rozměrech 130×150 pixelů trvá v řádu jednotek až desítek milisekund v závislosti na velikosti mřížky a doba klasifikace je řádově ve stovkách milisekund při zařazování do cca 1200 klasifikačních tříd. Na základě měření bylo zjištěno, že použití její uniformní varianty přináší zlepšení pouze u LDP 1. řádu a u vyšších řádů úspěšnost naopak výrazně snižuje. Její parametry jsou velmi obtížně naladitelné a metoda se při různě nastavených parametrech chová nevyrovnaně napříč testovacími množinami. Metoda se zároveň nedá snadno škálovat a při různě velkých obrázcích je možné pouze škálovat samotný obrázek, čímž ovšem dochází k určité ztrátě informace. Celkově tato metoda dopadla při srovnání nejhůře ze všech implementovaných metod.

Metoda POEM je ze všech implementovaných metod výpočetně nejnáročnější. Doba výpočtu příznaků jednoho obrázku o rozměrech 130×150 pixelů

trvá v řádu desítek milisekund v závislosti na velikosti mřížky a použitých parametrech a doba klasifikace je řádově v desítkách až stovkách milisekund při zařazování do cca 1200 klasifikačních tříd. Z naměřených výsledků je tato metoda jednoznačně nejúspěšnější na všech testovacích množinách. Metoda POEM je rovněž velmi dobře škálovatelná na různě velké obrázky a použití její uniformní verze sice způsobí v některých případech nepatrné snížení úspěšnosti, ale zároveň přináší výrazné snížení paměťové a výpočetní náročnosti. Tato metoda je rovněž velice robustní vůči použité metrice při měření podobnosti histogramů, kde i při použití prosté eukleidovské vzdálenosti nedochází k tak výraznému snížení úspěšnosti jako u ostatních dvou metod. V neposlední řadě je metoda POEM robustní i vůči nastavení svých parametrů, kde i přes jejich výrazně odlišné hodnoty se její úspěšnost příliš nemění.

5 Závěr

Prostudované metody používané pro rozpoznávání obličejů byly popsány v teoretické části práce, konkrétně se jedná o metody LBP, LDP a POEM, které využívají extrakci příznaků ve formě obrazových deskriptorů. Dále byla v této části stručně popsána problematika týkající se detekce a rozpoznávání obličejů obecně.

Po provedené analýze byly zvolené metody implementovány v jazyku C++ s využitím knihovny OpenCV. Následně byl vyvinut software, který umožňuje srovnávat úspěšnost implementovaných metod při rozpoznávání obličejů. V práci je rovněž popsán způsob instalace a použití knihovny OpenCV a ovládání vyvinutého programu.

Na základě naměřených výsledků byly u jednotlivých metod naladěny optimální parametry na testovacích množinách databáze FERET. Metody s naladěnými parametry byly vzájemně porovnány z hlediska jejich úspěšnosti při rozpoznávání obličejů, doby extrakce příznaků a klasifikace a jejich paměťové náročnosti. V této části byl rovněž představen experimentální genetický algoritmus umožňující automatické naladění vah dílčích histogramů tak, aby došlo ke zvýšení úspěšnosti jednotlivých metod. Nakonec bylo provedeno závěrečné zhodnocení výhod a nevýhod všech metod. Nejhůře si z hlediska úspěšnosti a škálovatelnosti vedla metoda LDP. Nejvyšší úspěšnost při rozpoznávání obličejů byla naměřena pomocí metody POEM. Ta je zároveň díky množství svých parametrů sice obtížněji naladitelná než ostatní metody, ale je možné ji lépe přizpůsobit testovacím datům. Její výpočetní složitost je ovšem nejvyšší ze všech implementovaných metod.

Slovníček pojmů a použitých zkratek

LBP	Local Binary Patterns.
LDP	Local Derivative Patterns.
POEM	Patterns of Oriented Magnitude Edges.
HS	Histogram Sequence. Sekvence histogramů představující globální popis obrázku nesoucí kromě statistické informace i prostorovou informaci.
RGB	Barevný model. Popisuje míchání (zastoupení) aditivních barev ve formátu (červená, zelená, modrá).
HSV	Barevný model. Popisuje barvu ve formátu (odstín, sytost, jas).
YIQ	Barevný model používaný v NTSC barevném TV signálu převážně v Americe a Japonsku.
YCbCr	Barevný model.

COD, CoD Cascade Object Detector. Učící se algoritmus umožňující rychlou detekci objektů.

PCA Principal Component Analysis. Analýza hlavních komponent.

ICA Independent Component Analysis. Analýza nezávislých komponent.

uniformní Obsahující nejvýše dva přechody z 0 na 1 a naopak ve spojení s binárními vzory.

DoG Difference of Gaussians. Rozdíl Gaussiánů.

FERET Databáze obrázků používaná pro hodnocení úspěšnosti metod při rozpoznávání obličejů.

fa, galerie Podmnožina obrázků z databáze FERET obsahující 1196 obličejů individuálních osob.

fb Podmnožina obrázků z databáze FERET obsahující 1195 obličejů individuálních osob.

fc Podmnožina obrázků z databáze FERET obsahující 194 obličejů individuálních osob, kde fotografie byly pořízeny při jiném osvětlení než v množině *fa*.

dup1 Podmnožina obrázků z databáze FERET obsahující 722 obličejů 243 individuálních osob, kde fotografie byly pořízeny s časovým odstupem 0 až 34 měsíců od fotografií v množině *fa*.

dup2 Podmnožina obrázků z databáze FERET obsahující 234 obličejů 75 individuálních osob, kde fotografie byly pořízeny s časovým odstupem alespoň 18 měsíců od fotografií v množině *fa*.

cell size Jeden z parametrů metody POEM popsaný v sekci 2.2.4.

block size Jeden z parametrů metody POEM popsaný v sekci 2.2.4.

grid size Velikost čtvercové mřížky v pixelech při konstrukci sekvence histogramů.

BSD Berkeley Software Distribution. BSD licence je licence pro svobodný software, mezi kterými je jednou z nejsvobodnějších.

Literatura

[1] Timo Ahonen, Student Member, IEEE, Abdenour Hadid, and Matti Pietikainen, Senior Member, IEEE. Face Description with Local Binary Patterns: Application to Face Recognition, 2006.

[2] Zhang, B., Gao, Y., Zhao, S., Liu, J. (n.d.). Local Derivative Pattern Versus Local Binary Pattern: Face Recognition With High-Order Local Pattern Descriptor. IEEE Transactions on Image Processing, 533-544.

[3] Raju, Dr USN, et al. Texture classification with high order local pattern descriptor: local derivative pattern. Global Journal of Computer Science and Technology 10.8 (2010).

[4] Vu, N., Dee, H., Caplier, A. (n.d.). Face recognition using the POEM descriptor. Pattern Recognition, 2478-2488.

[5] Opencv.org, (2015). OpenCV. [online] Available at: http://opencv.org/ [Accessed 9 Feb. 2015].

[6] Paula Viola and Michael Jones. Robust real-time object detection. In InternationalJournal of Computer Vision, 2001.

[7] Ojala, T., Pietikäinen, M. and Harwood, D. (1996), A Comparative Study of Texture Measures with Classification Based on Feature Distributions. Pattern Recognition 19(3):51-59.

[8] Turk, Matthew a Alex Pentland. Eigenfaces for Recognition. Journal of Cognitive Neuroscience [online]. 1991, vol. 3, issue 1, s. 71-86 [cit. 2015-03-15]. DOI: 10.1162/jocn.1991.3.1.71.

[9] Bartlett, M.S., J.R. Movellan a T.J. Sejnowski. Face recognition by independent component analysis. IEEE Transactions on Neural Networks [online]. 2002, vol. 13, issue 6, s. 1450-1464 [cit. 2015-03-16]. DOI: 10.1109/tnn.2002.804287.

[10] Stojilkovic, Jovana. Face recognition using Neural network [online]. Faculty of Organizational Sciences, University of Belgrade, 2013 [cit. 2015-03-16]. Dostupné z: `http://neuroph.sourceforge.net/tutorials/FaceRecognition/FaceRecognitionUsingNeuralNetwork.html`

[11] Face Recognition Technology (FERET) [online]. [cit. 2015-03-16]. Dostupné z: http://www.nist.gov/itl/iad/ig/feret.cfm

[12] Phillips, P. J. and Wechsler, H. and Huang, J. and Rauss, P. The FERET Database and Evaluation Procedure for Face Recognition Algorithms. Image and Vision Computing. 1998, vol. 16, s. 295-306.

[13] Image filtering: OpenCV 2.4.11.0 documentation. 2015. OpenCV [online]. [cit. 2015-05-06]. Dostupné z: `http://docs.opencv.org/modules/imgproc/doc/filtering.html?highlight=scharr#scharr`

[14] Sobel edge detector. 2003. HIPR [online]. [cit. 2015-05-06]. Dostupné z: `http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm`

[15] Nanni, Loris and Lumini, Alessandra and Brahnam, Sheryl. Survey on LBP based texture descriptors for image classification. Expert Systems with Applications. 2012, vol. 39, s. 3634-3641.

[16] Hsu, Rein-Lien and Abdel-Mottaleb, Mohamed and Jain, Anil K. Face detection in color images. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2002, vol. 24, s. 696-706.

[17] Wang, Yanjiang and Yuan, Baozong. A novel approach for human face detection from color images under complex background. Pattern Recognition. 2001, vol. 34, s. 1983-1992.

[18] Liu, Chengjun, Gabor-based kernel PCA with fractional power polynomial models for face recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2004, vol. 26, s. 572-581.

[19] Belhumeur, Peter N. and Hespanha, João P and Kriegman, David. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1997, vol. 19, s. 711-720.

[20] Tan, Xiaoyang and Triggs, Bill. Enhanced local texture feature sets for face recognition under difficult lighting conditions. Image Processing, IEEE Transactions on. 2010, vol. 19, s. 1635-1650.

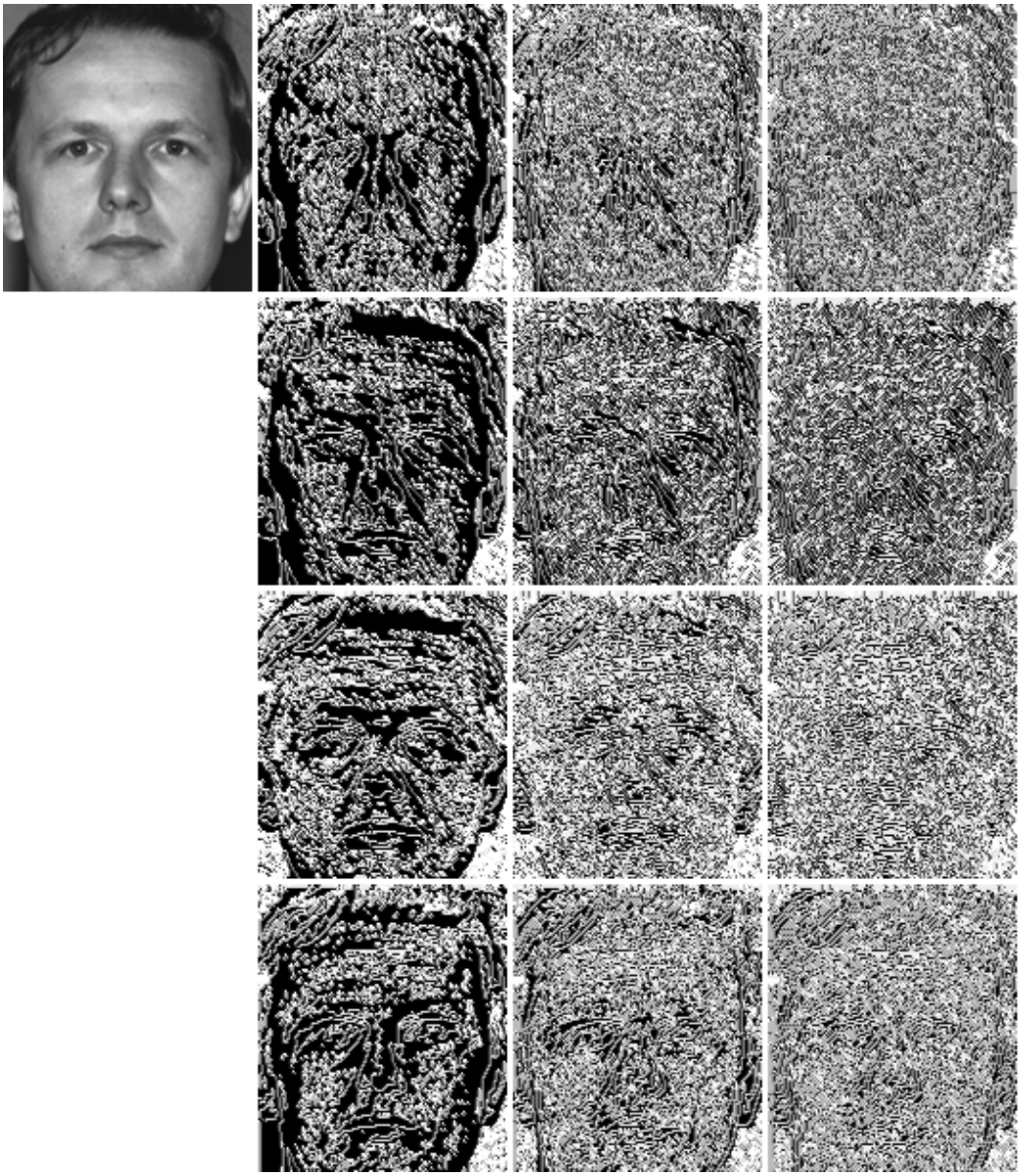
Přílohy

A Grafické znázornění implementovaných deskriptorů

Grafický výstup metody LBP je znázorněn na obrázku A.1. První obrázek je vstupní, druhý je po aplikaci LBP operátoru.

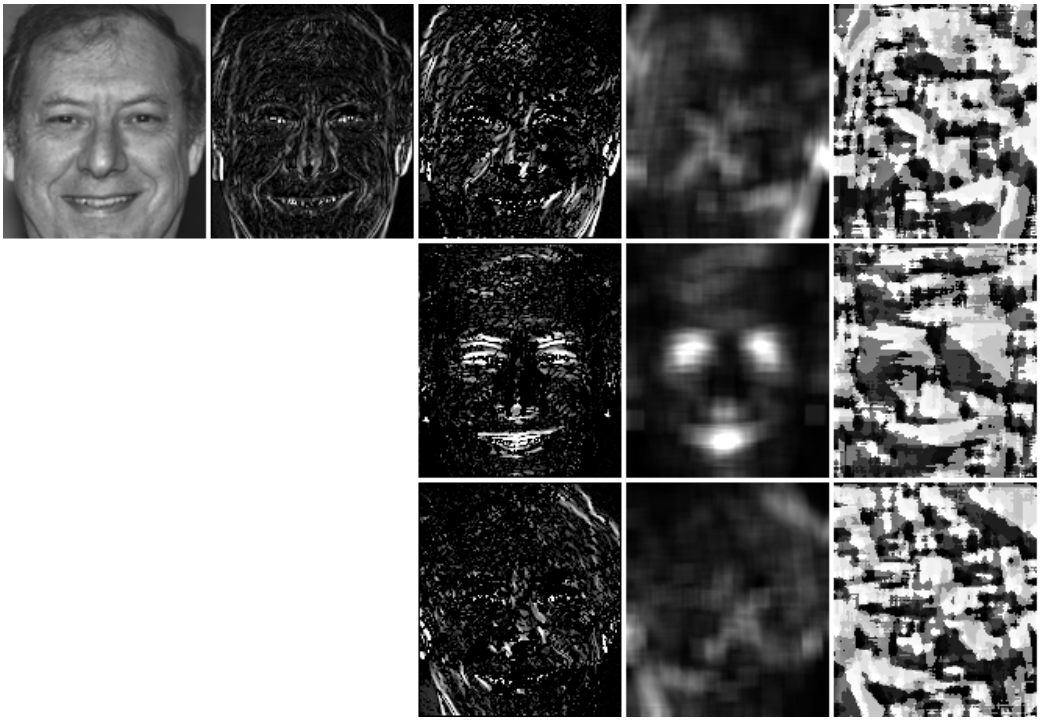
Obrázek A.1: Grafické znázornění obrázku po aplikaci LBP operátoru.

Grafický výstup metody LDP je znázorněn na obrázku A.2. V prvním sloupci je původní obrázek, v druhém sloupci je znázorněn původní obrázek po aplikaci LDP operátoru prvního řádu postupně ve směrech 0°, 45°, 90° a 135°. Ve třetím sloupci jsou znázorněny obrázky po aplikaci LDP operátoru druhého řádu a v posledním sloupci čtvrtého řádu.



Obrázek A.2: Grafické znázornění obrázku po aplikaci LDP operátoru.

Grafický výstup metody POEM je znázorněn na obrázku A.3. První obrázek zleva je vstupní, druhý je gradient obrázku, kde velikost gradientu v každém pixelu je znázorněna jako intenzita. V třetím sloupci je gradient diskretizován ve třech směrech. Ve čtvrtém sloupci jsou vypočteny lokální histogramy orientace gradientů z okolí a v posledním sloupci je na aplikován obecný kruhový operátor LBP s poloměrem o velikosti *block size*.



Obrázek A.3: Grafické znázornění obrázku po aplikaci POEM operátoru.

B Obsah přiloženého DVD

- Software
  - src - zdrojové soubory aplikace a soubory projektu VS
  - bin - spustitelná verze aplikace
- Dokumentace
  - Tisk - vygenerovaný PDF soubor s textem diplomové práce
  - Zdroj - zdrojové soubory diplomové práce a obrázky