Cvičení z předmětu Biometrie
Úloha 2: Identifikace osoby pomocí otisku prstu

Pavel Vostatek, kontaktní email: vostapav@fel.cvut.cz

8. ledna 2013

1 Úvod

Rozpoznávání otisků prstů lze rozdělit na dvě hlavní části: předzpracování otisku a rozpoznávání otisku. Předzpracování zahrnuje segmentaci otisku od pozadí, úpravu kontrastu, výpočet natočení papilárních linií v ploše otisku, výpočet hustoty papilárních liní, filtrace Gaborovou maskou pro vyhlazení drobných chyb, vytvoření kostry otisku a výsledně nalezení markantů. Pro každou část existuje bezpočet způsobů jejího zpracování. My se budeme zabývat pouze vybranými state of the art metodami. Vyčerpávající přehled metod lze nalézt v [1].

Z problematiky rozpoznávání otisku se budeme ve cvičení zabývat pouze identifikací totožnosti člověka. V této části srovnáme dvě metody, které identifikace využívá. První z nich, **markatové srovnávání**, vychází z klasického porovnávání otisků v daktyloskopii a srovnává rozložení markantů na otiscích. Druhá metoda, **Finger Code**, kvantuje po plochách mezikruží dané velikosti okolo otisku a z každé plochy vypočítá jednu hodnotu do příznakového vektoru. Vektory jsou poté srovnány mezi otisky.

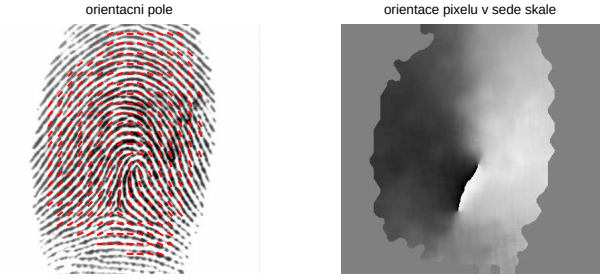
Úkolem bude v první části prakticky otestovat silné a slabé stránky předzpracování na pěti různých otiscích. Některé metody budete muset vymyslet a implementovat. V druhé části budete muset implementovat obě metody pro porovnání otisků, jejichž podrobný popis je součástí zadání, a následně zhodnotit sílu obou metod statisticky na databázi. K dispozici je Fingerprint Toolbox pro Matlab, který byl pro cvičení vytvořen. Obsahuje funkce pro načítání i zpracování otisku. Funkce potřebné pro každou operaci jsou uvedeny v příslušných kapitolách.

2 Předzpracování otisku

2.1 Orientace papilárních linií

První technikou předzpracování otisku je výpočet náklonu tečny papilárních liní v bodě $[x, y]$. Tyto hodnoty nejsou tradičně počítány pro každý pixel obrazu, ale vzorkovány s daným krokem, přičemž v případě potřeby pro libovolné místo v prostoru je hodnota získána interpolací. Úhel pro bod $[x, y]$ značíme Θ_{xy} a je počítán v otisku po blocích

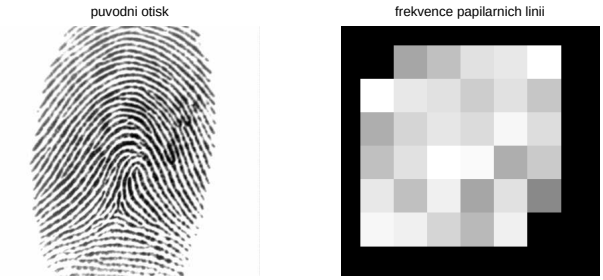
(průměrná hodnota v bloku dané velikosti), čímž je dáno vzorkování Θ_{xy} v obraze. klasickou metodou pro výpočet Θ_{xy} je použitím gradientu. Označme $\nabla(x, y)$ dvouprvkový vektor, který tvoří parciální derivace obrazu ∇_x, ∇_y podle souřadnic x a y. Ten je získán pomocí Prewittovy nebo Sobellovy masky filtrací (v Matlabu příkaz gradient). Vektor $\nabla(x, y)$ z definice míří ve směru nejvyššího růstu jasové intenzity — tedy ve směru nejvyššího kontrastu. Směr papilárních linií je poté brán jako kolmý směr na gradient. Ukázka výsledku výpočtu je na Obr. 2.1.



Obrázek 1: Orientační pole, `computeorientationarray(im, imSegmented, 10)`

2.2 Výpočet frekvenčního pole

Frekvenční pole papilárních liní můžeme popsat jako v prostoru vzorkovanou funkci $f(x, y)$, která určuje počet pap. linií na jednotku délky ve směru kolmém na směr linií. Opět exituje několik zajímavých algoritmů. Ve Fingerprint Toolboxu je implementovaný algoritmus využívající orientované okno. Okno definované velikosti je umístěno do polohy (x, y) a natočeno podle směru papilárních liní (pokud jde o asymetrické okno, je natočeno širším rozměrem kolmo na směr linií). Jasové hodnoty v každém sloupci jsou zprůměrovány, takže výsledkem je jednorozměrný vektor honot periodického průběhu odpovídající reliéfu prohlubní a výstupků linií. V tomto vektoru je poté spočítán počet vrcholů a délka mezer mezi nimi, frekvence je poté spočítána: $f = \frac{pocet\ mezer}{\sum(delka\ mezer)}$. Ukázka je na Obr. 2.2



Obrázek 2: Frekvenční pole, `computelocalfrequency(im, imSegmented, orientationArray)`

2.3 Segmentace otisku

Segmentace otisku se provádí s cílem porovnávat pouze relevantní části obrazu. Segmentaci může předcházet nejprve upravení kontrastu, což ale není nezbytně nutné. Z bezpočtu algoritmů sloužících k oddělení otisku od pozadí si zde několik zevrubně představíme.

Jádrem všech segmentačních algoritmů je zpracování otisku adaptivně — po blocích. Každý blok je samostatně zpracován a klasifikován na **otisk** nebo **pozadí**, případně je možné vytvořit více hodnocení podle kvality pozadí a rozpoznávat také nekvalitní nebo rozmazané části otisku. První nasnadě pro segmentaci by se nabízelo jednoduché prahování podle hodnoty jasové intenzity. Tento přístup ale není nejvhodnější, protože to co charakterizuje plochu otisku je hlavně vzor papilárních linií, který je doplněn šumem způsobeným nahrávacím čipem či nečistotami.

Používané metody pro segmentaci otisku lze shrnout do následujících skupin:

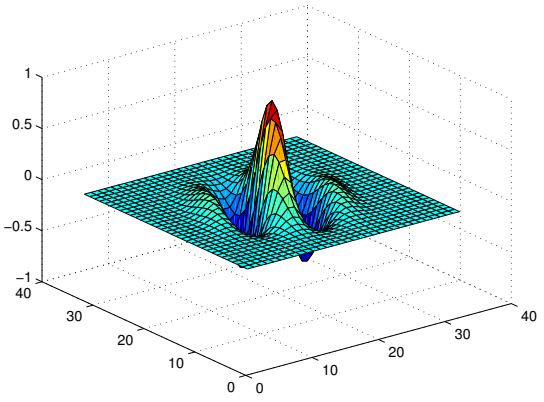
- Použitím **orientačního pole**, kdy je pro každý blok vypočítán histogram orientací papilárních linií. Pokud je v tomto histogramu patrná špička, znamená to jednotný směr linií v obraze a blok je klasifikován jako otisk.
- Použitím **variance jasové intenzity** ve **směru kolmém na směr papilárních linií**. Tento přístup vychází z toho, že mimo oblast otisku je variance jasu nezávislá na detekovaném směru papilárních linií.
- Použitím **gradientu jasové intenzity** počítaném průměrně v každém bloku, následně klasifikovaném podle velikosti gradientu. Tento způsob vychází z toho, že velikost gradientu je výrazně vyšší pro oblasti s papilárními liniemi než oblasti mimo otisk.
- Použitím **Gaborových filtrů**. Každý blok je filtrován sadou 8 Gaborových filtrů s různým natočením a výstup této filtrace je použit jak pro klasifikaci na otisk a pozadí a zároveň může sloužit k posouzení kvality daného bloku. Tento krok může být spojen s následným vyhlazováním otisku pomocí G. filtrace.

Obrázek 3: Ukázka možné segmentace. Zde počítáno prahováním velikosti gradientu jasové intenzity.

2.4 Vylepšení použitím Gaborových filtrů

Filtrování použitím Gaborových filtrů neboli tzv. kontextuální filtrací je směrovou filtrací obrazu. Běžně se používá jako hranový detektor, nám se tyto vlastnoti hodí na zahlazení

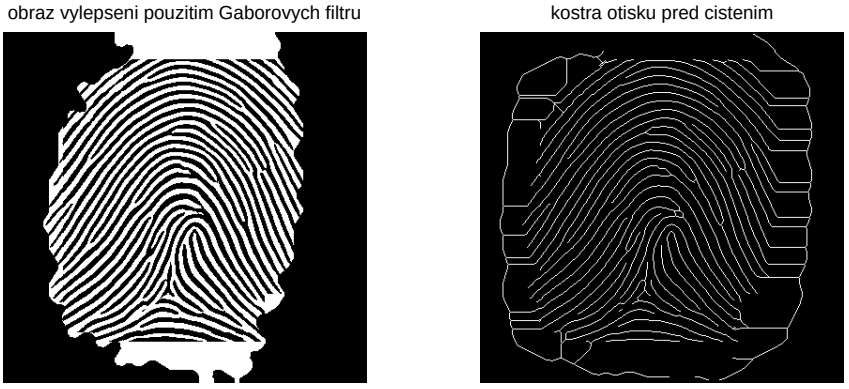
nespojistostí papilárních linií způsobených nekvalitou obrazu. Na Obr. 2.4 je vidět filtrační maska G. filtru v prostorové oblasti. Její vlnový tvar jí dává směrové vlastnosti. Vlna, která je vidět na obrázku je postupně natáčena do různých směrů. Natočením do směru pap. linií způsobí jejich zvýraznění a zahlazení chyb.



Obrázek 4: Maska G. filtrace, která je konvolvována s filtrovaným obrazem. Maska je z izometrického pohledu.

Obraz je opět zpracováván po blocích implementace Gaborovy masky bude v úloze za úkol. Je to jednoduché, její definice je následujícíc:

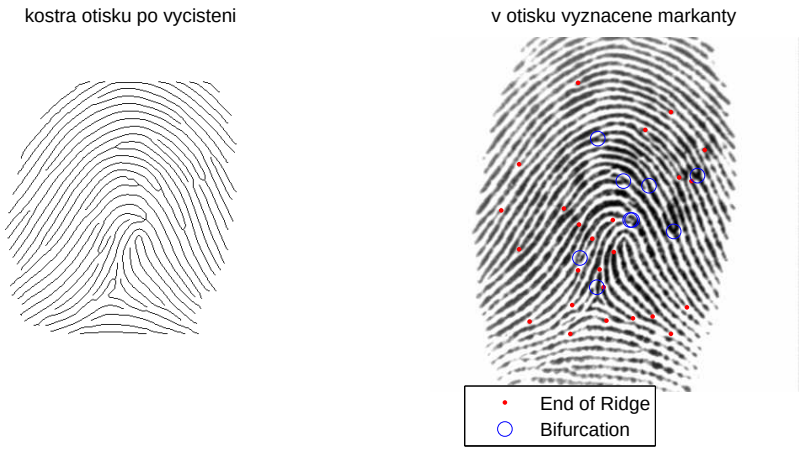
$x =< -16, 16 >, y =< -16, 16 >$
$x_p = sin(angle) \cdot x + cos(angle) \cdot y$
$y_p = sin(angle) \cdot y - cos(angle) \cdot x$
$gab(x, y) = exp\{-\frac{1}{2} \cdot [(\frac{x_p^2}{t_x^2}) + (\frac{y_p^2}{t_y^2})]\} \cdot cos(2\pi f \cdot x_p)$



Obrázek 5: Gaborova filtrace, `enhance2ridgevalley(im, imSegmented, orientationArray, frequencyArray, 0)`

2.5 Kostra otisku, markanty

5. Po filtraci Gaborovými filtry jsou papilární linie ztenčeny na nejmenší šířku - vzniké kostra otisku. V kostře otisku jsou poté hledány markanty např. pomocí jedoduchých pravidel. Zhodnoťte kvalitu nalezených markantů ve dvou zvolených otiscích.



Obrázek 6: Finalní detekce markantů, defaultní nastavení.

Při správné implementaci masky Gaborova filtru by měla funkce ukazka vrátit obrázky podobné, jako jsou zde.

3 Identifikace identity člověka pomocí otisku prstu

Identifikace - neboli určení totožnosti člověka podle otisku prstu je známá metoda používaná v kriminalistice. Potřebujeme k ní dostatečně velikou databázi otisků. Poté dostaneme otisk a naším úkolem je nalézt v databázi odpovídající otisk prstu, který nejpravděpodobněji odpovídá předloze. Případně určit, že nejsme schopni podobný otisk nalézt pro nekvalitu vstupního obrazu nebo nemáme hledaný otisk v databázi.

K identifikaci otisku potřebujeme hlavně metody na srovnání vstupního otisku s těmi v databázi. V předchozích kapitolách jsme si ukázali, jak otisk předzpracovat a připravit na extrakci možných příznaků. Následně si ukážeme dvě metody pro porovnání otisků. První vychází z klasického porovnání pomocí významných bodů, tzv. markantů. Jejichž topologie jednoznačně určuje lidský otisk. A druhá je založena na textuře otisku a extrakci příznakového vektoru z ní použitím Gaborovy filtrace.

3.1 Markantové rozpoznání

Pro markantové porovnánvání otisků je připravena funkce `match.m`, kterou je třeba implementovat.

Algoritmus: Vstupem jsou dvě sady markantů (bodů) v rovině: `mAi1` a `mAi2` a práh pro vzdálenost `d`. Vstupní počty markantů jsou: `|mAi1|`, resp. `|mAi2|`, počet párů markantů iniciovaný `nbmatch = 0` a celková vzdálenost `dist = 0`.

• Pro každý bod z `mAi2` nalezněte nejbližší bod v `mAi1`. Pokud jsou markanty vzdáleny méně, než `d`, potom je označte za pár a vyjměte z `mAi1` i `mAi2`. Za každý nalezený pár `nbmatch + 1`

• Po vyčerpání všech markantů z `mAi2` možných spárovat vypočtěte $matchingScore = \frac{2 \cdot nbmatch}{|mAi1|+|mAi2|}$, což je výsledné ohodnocení přiřazení otisků. Platí, že čim vyšší matchingScore, tím spíš patří patří otisky stejnému původci (inteval je $< 0, 1 >$).

Syntaxe funkce `match` je následující:

```
[matchingScore, nbmatch, inputmatch, dbmatch] = match(mAi1, mAi2);
```

Vstupem jsou pole s vyznačenými markanty pro vzor (*mAi1*) a porovnávaný otisk (*mAi2*). Tato pole jsou přímo výstupem funkce `findminutia`, přičemž *mAi2* musí být zarovnáno s *mAi1*.
Výstupy jsou:
matchingScore - ohodnocení podobnost dvou otisků. Výpočet je níže.
nbmatch - počet dvojic markantů.
inputmatch - pole stejné velikost jako mAi2, kde jsou vyznačeny markanty stejným způsobem jako v mAi2, ale pouze ty, které byly zpárovány.
dbmatch - stejně jako inputmatch, ale pro otisk 1.

3.2 Rozpoznávání pomocí Finger Codu

Pomocí sady Gaborových fitrů s různým natočením filtrujte otisk prstu. Vznikne k filtrovaných obrazů. Každý filtrovaný obrázek zpracujte po blocích N velikosti n x n bez překryvu. Hodnotu pro každý blok spočítejte jako $f(N) = |mean(N) - std(N)|$, kde std značí směrodatnou odchylku, mean střední hodnotu. Z každého bloku vznikne následně jeden pixel. Blokový obraz poté vyřízněte maskou pouze okolo jádra otisku, jak ukazuje Figure 1.

Obrázek 7: Postup při vytváření FingerCodu

Následný příznakový vektor f_1 (odpovídající vzoru, resp. f_2 odpovídající vstupnímu obrazu) vznikne seřazením hodnot ležících v mezikruží (nevynulované hodnoty Figure 1 vpravo) z každého z k obrazů a jejich seřazením za sebe. Porovnání 2 otisků poté proveďte jako
$matchingScoreGabor = mean(abs(f_1 - f_2))$[1]. Čím nižší je výsledná hodnota, tím lépe si otisky odpovídají, tedy jako *score* použijeme zápornou hodnotu vzdálenosti.

Pro Finger Code je připravena funkce `fingercode_creation.m`.
`[fingercode Fmasked] = fingercode_creation(imOriginal, Gfilt, core, maskSize, dia)`,

imOriginal je vstupni obrazek pro vypocet, *Gfilt* je sada k Gaborových filtrů vytvořená pomocí `GaborFilter_creation`, *core* jsou souřadnice jádra otisku, *maskSize* je velikost bloku pro filtraci obrazu a *dia* jsou velikosti vnějšího a vnitřního poloměru ořezu okolo jádra.

Výstupy:
fingercode - vektor velikosti [1 x N], kde N záleží na velikosti mezikruží (viz dále).
Fmasked - blokový obraz s vyříznutým mezikružím (Figure 1, vpravo) pro každý Gaborův filtr. Velikost [M x N x O]. M x N je velikost blokového obrazu a O je počet Gaborových filtrů. **Výstup Fmasked není nutné implementovat.**

4 Vypracování úlohy

Každý student dostane k dispozici 10 otisků různé kvality. Úkolem bude pomocí připraveného toolboxu vytvořit systém, který nalezne v poskytnuté databázi nejpravděpodobnějšího odpovídajícího adepta, případně rozhodne o nemožnosti nalézt vhodného adepta v databázi.

Úkoly:

- Na poskytnutých otiscích nejprve vyzkoušejte výpočet frekvenčního a orientačního pole. Vizuálně zhodnoťte, jakou má kvalita otisku vliv na výsledek výpočtu. [2 b]
- Implementujte vlatní algoritmus segmentace otisku. Můžete si zvolit z algoritmů jmenovaných v Kapitole 2.3 nebo si vybrat jiný z literatury ([1]). Můžete vymyslet vlastní algoritmus, který automaticky segmentuje otisky, které jste dostali v zadání. [4 b]
- Pro umožnění Gaborovy filtrace je potřeba imlementovat výpočet G. masky do funkce `enhance2ridgevalley.m` (v teto funkci je podfunkce `filtergabor`) podle kapitoly 2.4. [2 b]
- Nahrajte vlastní sadu otisků na našich snímačích a srovnejte jak si poradí funkce na předzpracování otisku s výstupy jednotlivých snímačů. Budou vždy nalezené markanty stejné nebo různé. Segmentuje správně Váš algoritmus pro segmentaci otisu nahrané otisky? [2 b]

- Implementujte funkci `match` pro výpočet shody mezi zarovnaným vstupním obrázkem a vzorem (v databázi) pomocí markantů. [4 b]
- Implementujte funkci `fingercode_creation` po výpočet shody mezi obrazy pomocí Finger Codu. [4 b]
- Pro každý z otisků vyzkoušejte přiřadit identitu pomocí poskytnuté databáze. [2 b]

A Příloha: Struktura Sigature Toolboxu.

Toolbox je rozdělen na 2 části: *predzpracovani* a *porovnani*, každá obsahuje funkce pro jednotlivé části úlohy. Funkce potřebné pro každý popisovaný algoritmus jsou popsány v textu.

Reference

[1] Anil K. Jain and David Maltoni. *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.