Cvičení z předmětu Biometrie
Porovnávání otisků prstů

P. Vostatek

November 5, 2012

1 Úloha 2, Otisk prstu - srovnávání otisků (15 bodů)

Termín odevzdání: 18. 11. 2012, 23:59

Porovnávání otisků prstů bude mít 2 hlavní části:

1. Porovnání pomocí markantů
2. Porovnání pomocí FingerCodu (Gaborova filtrace)

Mezi zdrojovými kódy cvicení je opět připraven skript *ukazka.m*, který demonstruje použití dodaných funkcí. Ve skriptu se nejprve načtou dva obrázky otisků, ty jou předzpracovány (funkce *preprocess*), zarovnány na sebe (funkce *align2* nebo *align_manually*) a nakonec je porovnána podobnost otisků pomocí obou výše uvedených metod. Aby systém fungoval předpokládá rozdělení funkcí a dat v adresářích `<cesta>/data`, `<cesta>/predzpracovani`, `<cesta>/porovnani`.

fingercode_score.m slouží k výpočtu skóre z otiskových fingerCodů. K porovnání metodou markantů je funkce *match.m*. Hlavním úkolem cvičení bude implementace těchto 2 funkcí pro porovnání otisků (u fingerCodu se implementuje fce *fingercode_creation*). Otisky vstupují do funkcí už vzájemně zarovnané podle jejich jádra. Po implementaci kódů bude za úkol zhodnotit použitelnost uvedených algoritmů pro porovnání otisků.

Také jsou přibaleny implementované funkce *match* a *fingercode_creation* ve formě p-souborů (*matchP*, resp. *fingercode_creationP*), což umožňuje jejich spuštění, ale ne editaci. Můžete na nich vyzkoušet funkci a správné výstupy.

2 Požadavky

1. Implementujte funkci "match" pro výpočet shody mezi zarovnaným vstupním obrázkem a vzorem (v databázi) pomocí markantů. Popis používaných funkcí je v jejich helpu.

Syntaxe 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 podobnosti 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.

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 $< 0, 1 >$).

2. Implementujte funkci "fingercode_creation":

`[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.**

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.

Figure 1: 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.

3 Poznámky

Pokud nepůjdou otisky zarovnat automaticky, použijte funkci *align_manually*, použití je v její nápovědě. Funkce na porovnání pomocí fingerCodu nefunguje v případě, že některé mezikruží vybíhá z obrázku, poté nejsou příznakové vektory stejně dlouhé. Řešení těchto okrajových případů nemusíte v implementaci zohledňovat.

4 Výsledky

Po implementaci funkcí vyzkoušejte jejich funkčnost na otiscích vybraných v minulém úkolu. Poté otestujte, zda je znatelný rozdíl mezi hodnotami výsledného hodnocení u odpovídajících dvojic otisků a dvojic, které neodpovídají stejnému prstu. Není potřeba počítat statistiku, jenom na vybraných otiscích vyzkoušet a slovně zhodnotit. Správnost zarovnání otisků u odpovídajících si dvojic kontrolujte vizuálně pomocí zarovnaných koster. Pokud nejsou odpovídající si otisky správně zarovnány, zarovnejte manuálně nebo vyřaďte z hodnocení.

Vytvořte protokol, kam dejte obrázky správně zarovnaných otisků a číslené hodnoty dosažených podobnostních score. Všechny implementované funkce zabalte do zip balíku a pošlete i s protokolem do courseware. Do protokolu i mezi kódy sepište a přiložte i výsledky z první části úlohy. **Alternativně** je možné odevzdat ústně, bez psaní protokolu, předvedením implenetovaných kódů, jejich funkce a výsledků na některém ze cvičení bloku.