Vaja 2: Razpoznavanje s histogrami in konvolucija

Laboratorijske Vaje

29. oktober 2011

Vaja je sestavljena iz dveh sklopov. V prvem sklopu se boste v praksi seznanili s pomembnejšimi metodami primerjave histogramov. V drugem sklopu se boste spoznali z operacijo konvolucije v sliki, ki je osnova za filtriranje slik. V vaši delovni mapi kreirajte mapo '`vaja2/`', ter si s spletne strani predmeta vanjo razpakirajte datoteko '`vaja2.zip`'. Rešitve nalog boste pisali v Matlab/Octave skripte v mapi '`vaja2/`' in jih zagovarjali ob zagovoru nalog. Pri nekaterih nalogah so vprašanja, ki zahtevajo razmislek. Odgovore na ta vprašanja si zabeležite v pisni obliki in jih prinesite na zagovor.

1 Naloga: Globalni pristop k opisu slik

V prejšnji vaji ste se seznanili s postopki gradnje histogramov. V tej nalogi si bomo ogledali kako lahko uporabimo histograme kot globalne opisnike za primerjavo slik.

(a) Najprej bomo implementirali funkcijo za primerjanje histogramov, ki je sposobna izvajati primerjavo z uporabo različnih mer razdalj. Funkcija za vhod funkcija prejme dva 1D histograma in ime mere podobnosti, vrne pa vrednost te mere med histogrami. Za izhodišče uporabite spodnjo kodo, nato pa implementirajte mero razdalje tipa L_2, ki je razložena spodaj.

```
function d = getDistanceHist( h1, h2, dist_name )
 switch dist_name
    case 'l2'
        d = getL2Distance( h1, h2 ) ; % vaša koda
    case 'chi2'
        d = getChi2Distance( h1, h2 ) ; % vaša koda
    case 'hell'
        d = getHellingerDistance( h1, h2 ) ; % vaša koda
    case 'intersect'
        d = getIntersectionDistance( h1, h2 ) ; % vaša koda
    otherwise
        error('Nepoznan parameter!') ;
 end
```

Mera razdalje tipa L_2 je evklidska razdalja, kjer vsak histogram s številom celic N_{bins} obravnavamo kot točko v N_{bins}-dimenzionalnem prostoru. Za histograma h_1 in h_2 je torej L_2 definirana kot:

$$L_2 = [\sum_{i=1:N_{bins}} (h_1(i) - h_2(i))^2]^{\frac{1}{2}}. \quad (1)$$

V Octave/Matlab lahko zgornjo enačbo implementirate v vektorskem zapisu:
`d = sqrt( sum( (h1-h2).^2 ) ) ;`

(b) Preizkusite vašo mero razdalje. Naložite slike:

- 'data/model/obj58__0.png',
- 'data/model/obj68__0.png',
- 'data/model/obj37__0.png'.

Slike pretvorite v sivinske in izračunajte za vsako sivinski histogram z osmimi celicami (uporabite `myhist()` z Vaje 1). S funkcijo '`subplot(,,)`' prikažite tri slike, pod njimi pa histograme[1]. **Pozor:** *Vsak histogram morate normirati tako, da bo njegova vsota 1 (torej, cel histogram delite z vsoto njegovih celic)! Normalizacijo dodajte kar na konec funkcije* `myhist()`*! Kasneje, ko bomo uporabljali RGB histograme jih prav tako ne pozabite normirati.* Izračunajte L_2 razdaljo med histogrami h_{obj58} in h_{obj68} ter L_2 razdaljo med histogrami h_{obj58} in h_{obj37}. Rešitev te naloge napišite v skripto '`vaja2_naloga1bc.m`'.

***Vprašanje:** Glede na mero razdalje L_2, katera slika je bolj podobna sliki '`obj58__0.png`'? Vidimo, da imajo vsi trije histogrami močno izraženo določeno komponento (ena celica je precej bolj polna od ostalih). Kaj v sliki je vzrok temu?

(c) Ponovite zgornji eksperiment še za naslednje mere razdalj, ki ste jih spoznali na predavanjih (rešitve dodajte v skripto '`vaja2_naloga1bc.m`'):

- Chi-kvadrat razdalja $\chi^2 = \frac{1}{2} \sum\limits_{i=1:N_{bins}} \frac{(h_1(i)-h_2(i))^2}{h1(i)+h2(i)+\varepsilon_0}$, kjer je ε_0 zelo majhna konstantna (npr., $1e-10$) vrednost s katero se ognemo patološkim primerom deljenja z ničlo.
- Presek $I = 1 - \sum_{i=1:N_{bins}} \min(h_1(i), h_2(i))$
- Hellingerjeva razdalja $H = (\frac{1}{2} \sum\limits_{i=1:N_{bins}} (h_1(i)^{\frac{1}{2}} - h_2(i)^{\frac{1}{2}})^2)^{\frac{1}{2}}$

(d) Sedaj imate vse pripravljeno za preizkus iskanja podobnosti med slikami s histogrami. V delovni mapi imate datoteko `data/slike.txt`, ki vsebuje spisek slik v mapi `data/slike/`. Najprej moramo slike prebrati in za vsako izračunati histogram. Implementirajte funkcijo `getAllHists.m`, ki vzame za vhod datoteko `files='data/slike.txt'`, za vsako sliko izračuna RGB histogram[2], ga preuredi v 1D histogram in vrne 2D matriko histogramov vseh slik. Matrika je sestavljena tako, da njena i-ta vrstica vsebuje histogram i-te slike. Za izhodišče si pomagajte s sledečo kodo:

```
function histograms = getAllHists(files,bins)
 files=textread(files,'%s');
 histograms=0;
 %nb=1;%matlab
 nb=2;%octave
 % inicializirajmo matriko histogramov
 histograms=zeros(size(files,nb),bins^3);
 % izračunaj histogram za vsako sliko
 for i = 1 : size(files,nb)
    %filename=char(files(i));%matlab
    filename=['data/',nth(files,i)];%octave
    I = imread(filename) ;
```

[1] Če uporabljate Octave, potem zaradi preglednosti raje ne rišite vse na en subplot ampak narišite za vsako sliko posebej.

[2] vzemite funkcijo `function h = myhist2RGB(img1,bins)`, ki ste jo uporabili v Vaji 1.

```
    h = myhist2RGB(I,bins);
    histograms(i,:) = h(:) ;
end
```

(e) Rešitev te naloge napišite v skripto `vaja2_naloga1e.m`. Uporabite funkcijo s prejšnje točke in izračunajte histograme za vse slike v spisku `slike.txt`. Vzemite tretjo sliko (histogram) v spisku za referenčno in izračunajte razdalje do vseh ostalih slik (njihovih histogramov). Izrišite referenčno sliko in prvih pet slik (uporabite `sort()` za sortiranje mere podobnosti), ki so po histogramu najbližje prvi sliki. Če delate v Matlabu, si pod njimi izrišite še pripadajoče histograme (kar v 1D s funkcijo `bar()`). Za izris na skupno sliko uporabite funkcijo `subplot(,,)`. Preglejte razvstitve za vse vaše mere razdalje, ki ste jih implementirali.

Vprašanje: Izberite število celic 8 na barvni kanal – katera mera razdalje po vašem mnenju deluje najbolje (utemeljite odgovor)? Kako se spreminja razvrstitev najbolj podobnih slik, če zelo povečamo število celic v histogramih ali če jo zelo znižamo? *Za dodatek: dober pripomoček za vizualizacijo razdalj je, če si za izrišete graf, pri katerem imate na x osi zaporedno številko slike, na y osi pa razdaljo do referenčne slike. Tako lažje vidite ali so referenčni sliki najbolj podobne slike bistveno bolj podobne od ostalih.*

2 Naloga: Konvolucija in filtriranje

To vajo pišite v skripto '`vaja2_naloga2.m`'. Na predavanjih ste se spoznali z linearnim filtriranjem, katerega osrednji del je operacija konvolucija. Najprej si oglejmo implementacijo konvolucije na 1D signalu. Konvolucija jedra $g(x)$ preko slike $I(x)$ je definirana z naslednjim izrazom

$$I_g(x) = g(x) * I(x) = \int_{-\infty}^{\infty} g(u)I(x-u)du, \tag{2}$$

oziroma v primeru diskretiziranih signalov

$$I_g(i) = g(i) * I(i) = \sum_{-\infty}^{\infty} g(j)I(i-j)dj. \tag{3}$$

Zelo lepo vizualizacijo konvolucije si lahko ogledate na spletni strani Wikipedia[3]. Ker je ponavadi naše jedro končne velikosti, zgornja vsota teče samo od *levega roba jedra* do *njegovega desnega roba*. Na primer, recimo, da je naše jedro velikosti $N + 1 + N$ elementov. V i-ti točki signala $I(i)$ se vrednost konvolucije v Octave izračuna kot `I_g(i) = sum(I(i-N:i+N).*g)`. To pomeni, da center jedra 'položimo' na signal v i-ti točki in seštejemo produkt istoležnih elementov.

(f) Implementirajte funkcijo `preprostaKonvolucija.m`, ki za vhod vzame 1D signal I in simetrično jedro g velikosti (2N+1), ter izračuna konvolucijo I_g. Zaradi enostavnosti lahko začnete konvolucijo računati na mestu $i = N + 1$ in končate na $i = length(I) - N$. To pomeni, da za prvih N elementov in zadnjih N elementov signala I kovolucije ne boste izračunali – *ali se lahko spomnete rešitve, kako bi tudi za tiste elemente konvolucijo izračunali?* Implementirajte spodnjo funkcijo. Z diska preberite signal

[3] `http://en.wikipedia.org/wiki/Convolution`

`I=load('data/konvsignal.txt')`, jedro `g=load('gauss_sigma2.txt')` in izračunajte konvolucijo.

***Vprašanje:** Izrišite si signal I, jedro g in rezultat Ig. Ali prepoznate obliko jedra g? Kakšna je vsota vseh elementov jedra? Rešitev naloge pišite v skripto '`vaja1_naloga2fg.m`'.

```
function Ig = preprostaKonvolucija(I, g)
  N = (length(g)-1)/2 ;
  Ig = zeros(1, length(I)) ;
  for i = N+1 : length(I)-N
   i_levo = max([1, i-N]) ;
   i_desno = min([length(I), i+N]) ;
   Ig(i) = sum(g.*I(i_levo:i_desno)) ;
  end
```

(g) Ponovno izračunajte konvolucijo, vendar tokrat uporabite že vgrajeno funkcijo `Ig=conv2(I,g,'same')`. Poglejte v `help` kaj pomeni parameter `'same'`. V čem se rezultat razlikuje od `preprostaKonvolucija(I, g)`? Kaj je vzrok? Rezultat dodajte v skripto '`vaja1_naloga2fg.m`'.

(h) Sedaj si oglejmo zelo pogosto uporabljano Gaussovo jedro, ki je definirano kot

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2}). \tag{4}$$

Pomembna lastnost Gaussovega jedra je, da njegova vrednost postane zelo majhna za $|x| > 3\sigma$. Zato je tudi jedro ponavadi velikosti $2 * 3\sigma + 1$. Naredite novo datoteko `gauss.m` in napišite funkcijo, ki ji podate parameter `sigma`, vrne pa Gaussovo jedro.

```
function g=gauss(sigma)
 x = [-round(3.0*sigma):1:round(3.0*sigma)] ;
 g = ... % <-- !!! tukaj izračunajte Gaussovo jedro na x
 g = g / sum(g) ; % normalizirajte, da bo vsota jedra 1
```

Generirajte jedro s $\sigma = 2$, si ga izrišite, preverite, da je vsota njegovih elementov 1 in da je po obliki podoben jedru v datoteki `gauss_sigma2.txt`.

(i) Naslednja pomembna lastnost Gaussovega jedra, ki ste jo spoznali na predavanjih je separabilnost v večih dimenzijah. Gaussovo jedro v 2D prostoru namreč zapišemo kot

$$\begin{aligned} G(x,y) &= \frac{1}{2\pi\sigma} \exp(-0.5\frac{x^2+y^2}{\sigma^2}) && (5)\\ &= [\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2})][\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{y^2}{2\sigma^2})] && (6)\\ &= g(x)g(y), && (7) \end{aligned}$$

torej kot produkt dveh 1D Gaussovih jeder, vsakega v svoji dimenziji. Če sedaj ponovno zapišemo zvezno konvolucijo

$$\begin{aligned} G(x,y) * I(x,y) &= \int_u \int_v g(u)g(v)I(x-u, y-v)dudv && (8)\\ &= \int_u g(u)(\int_v g(v)I(x-u, y-v)du)dv && (9)\\ &= g(x) * [g(y) * I(x,y)], && (10) \end{aligned}$$

vidimo, da dobimo enak rezultat, če enkrat filtriramo z 2D jedrom ali če filtriramo najprej po eni in nato po drugi dimenziji. Torej lahko (počasno) nD filtriranje prevedemo na sekvenco hitrih 1D filtriranj.

(j) Napišite funkcijo `gaussianfilter.m`, ki generira Gaussov filter in ga nato aplicira na 2D sliko. Uporabite dejstvo, da je jedro separabilno, zato generirajte 1D jedro, z njim najprej filtrirajte sliko po eni dimenziji (`Ib = conv2(I,g,'same')`), nato pa po drugi dimenziji (`Ig = conv2(Ib,g','same')`) preprosto tako, da jedro transponirate pred konvolucijo (t.j., `g'`). Preizkusite filter v spodnji kodi, ki naloži sliko '`data/xuq13lm7.bmp`', jo spremeni v sivinsko in pokvari z Gaussovim šumom, in šumom sol-in-poper. Pokvarjene slike filtrira z vašim Gaussovim filtrom s $\sigma = 1$.

***Vprašanje:** Kateri šum Gaussov filter bolje odstrani?

```
function vaja2_naloga2j()
 A = imread('data/xuq13lm7.bmp') ;
 A = rgb2gray(A) ;
 % Gaussov šum
 Icg = double(A) + 15*randn(size(A)) ;
 figure(1); clf ;
 subplot(2,2,1); imagesc(Icg); colormap gray;
 axis equal; axis tight; title('Gauss šum') ;
 % Šum tipa Sol in poper
 Ics = imnoise(A,'salt & pepper',0.1) ;
 subplot(2,2,2) ; imagesc( uint8(Ics) ) ; colormap gray ;
 axis equal; axis tight; title('Sol in poper') ;
 % naredi Gaussov filter
 sigma = 1 ;
 g=gauss(sigma) ;
 Icg_b = gaussianfilter( Icg, g ) ;
 Ics_b = gaussianfilter( Ics, g ) ;
 subplot(2,2,3) ; imagesc( uint8(Icg_b) ) ; colormap gray ;
 axis equal; axis tight; title('filtrirana') ;
 subplot(2,2,4) ; imagesc( uint8(Ics_b) ) ; colormap gray ;
 axis equal; axis tight; title('filtrirana') ;
```

(k) Sedaj implementirajte nelinaren filter, ki ste ga obravnavali na predavanjih – *medianin filter*. Medtem, ko Gaussov filter izračuna lokalno uteženo povprečno vrednost v signalu, medianin filter lokalne vrednosti v signalu (t.j., vrednosti znotraj okna filtra) uredi po velikosti in vzame vrednost, ki je na sredini urejene množice (t.j., mediano). Implementirajte spodnji preprosti 1D medianin filter, ki za vhod vzame signal I in širino filtra W.

```
function Ig = preprostaMediana(I, W)
 N = ceil((W-1)/2) ;
 Ig = zeros(1, length(I)) ;
 for i = N+1 : length(I)-N
    i_levo = max([1, i-N]) ;
    i_desno = min([length(I), i+N]) ;
    vals = sort(I(i_levo:i_desno)) ;
    Ig(i) = vals(round(length(vals)/2)) ;
 end
```

Uporabite spodnjo funkcijo, ki generira 1D stopnico, jo pokvari s šumom sol-in-poper, in nato zažene vaš Gaussov in medianin filter. Nastavite parametre filtrov tako, da bo rezultat filtriranja najboljši.

***Vprašanje:** Kateri filter se obnese bolje?

```
function testStopnica()
 x = [zeros(1,10),5,zeros(1,3),ones(1,5),5,ones(1,5),zeros(1,15)] ;
 figure(1) ; clf ;
 subplot(1,3,1); plot(x); axis([1,40,0,7]); title('vhod')
 sigma = 1 ;
 g=gauss(sigma) ;
 x_g = conv2(x,g,'same') ;
 W = 5 ;
 x_m = preprostaMediana(x, W) ;
 subplot(1,3,2); plot(x_g); axis([1,40,0,7]); title('Gauss')
 subplot(1,3,3); plot(x_m); axis([1,40,0,7]); title('Mediana')
```

***Dodatno vprašanje:** Primerjajte (ocenite analitično) kolikšna je računska kompleksnost $O(?)$ Gaussovega filtra in kakšna medianinega, če v mediani uporabimo quicksort za sortiranje.