Základy zpracování obrazu

Tomáš Mikolov, FIT VUT Brno

V tomto cvičení si ukážeme základní techniky používané pro digitální zpracování obrazu. Pro jednoduchost budeme pracovat s obrázky ve stupních šedi - zpracování barevných RGB obrázků probíhá analogicky, musíme však všechny operace provést pro každou barevnou složku zvlášť.

1 Načtení a vykreslení obrázku

```
I=imread('lena.gif');
imshow(I);
```

Nezapomínejte středníky za příkazy, které vrací bitmapy (matice)!

1.1 Základní operace s obrazem

Velikost obrázku můžeme zjistit jednoduše příkazem `size(I)`, který nám vrátí velikost matice. Pro výpis všech nastavených proměnných lze použít příkaz `whos`, kde uvidíme i datový typ přiřazený proměnným - náš obrázek bude mít typ `uint8`.

Otočení obrazu:

```
Irot=imrotate(I, 30, 'bilinear');
imshow(Irot);
```

Uložení obrazu:

```
imwrite (I, 'lenacopy.gif');
```

Úkol

- zjistěte velikost obrázku `lena.gif` před a po otočení o 45 stupňů

2 Základní techniky zpracování obrazu

Mnoho jednoduchých transformací obrazu můžeme realizovat pomocí průchodu přes celou matici obrazu, kde postupně měníme hodnoty jednotlivých prvků podle nějaké předem dané funkce. Tyto operace lze ve velké většině realizovat jedním příkazem v Matlabu. Protože však smyslem tohoto cvičení je ukázat, jak věci fungují, ukážeme si "ruční" přístup, který je sice výrazně méně efektivní, ale více názorný.

Pokud se podíváme na matici I, kterou jsme vytvořili v předcházejícím příkladu načtením obrázku `lena.gif`, zjistíme, že se nám v ní objevují hodnoty v rozsahu 0-255. Samotná hodnota určuje intenzitu bílé barvy na dané pozici v obraze (0 odpovídá černé, 255 bílé barvě). Názorně to ukazuje funkce `colorbar`. Pozice [0, 0] odpovídá levému hornímu rohu - můžeme si vyzkoušet jednoduchý experiment:

```
Icopy=I;
Icopy(1:200, 1:400)=zeros(200, 400);
imshow(Icopy); colorbar;
```

2.1 Inverze barev

Poměrně známou transformací obrazu je inverze barev:

```
for (x=1:512)
  for (y=1:512)
    I2(x,y)=255-I(x,y);
  end;
end;
imshow(I2);
```

Takto byste museli inverzi naprogramovat v C. V Matlabu ale stačí napsat:

```
I2 = 255 - I;
imshow(I2);
```

V tomto cvičení Vám ale schválně budeme většinu operací presentovat pomocí cyklů (i když je to "Matlabově neoptimální"), abyste si dokázali představit, jak tyto operace implementovat v C, Javě, atd.

2.2 Jednoduchá detekce hran

Pro další zpracování obrazu je často vhodné detekovat hrany. Zatím si ukážeme velmi jednoduchý přístup, kdy zobrazujeme rozdíl hodnot sousedních pixelů násobený nějakou konstantou (v našem případě 10) pro vetší kontrast:

```
for (x=1:511)
  for (y=1:512)
    I2(x,y)=(I(x,y)/2 - I(x+1,y)/2)*10;
  end;
end;
imshow(I2);
```

Protože na hranách je rozdíl hodnot sousedních pixelů mnohem větší než na jednobarevných plochách, zobrazí se nám hrany jako bílé čáry.

2.3 Thresholding

Mezi další klasické funkce patří thresholding (česky prahování). Nejprve si určíme hodnotu prahu a poté pixely obrazu rozdělíme podle této hodnoty do dvou tříd.

```
prah=120;
for (x=1:512)
  for (y=1:512)
    if (I(x,y)<prah) I2(x,y)=0;
      else I2(x,y)=255;
    end;
  end;
end;
imshow(I2);
```

Úkol

- vyzkoušejte různé hodnoty pro prahovou hodnotu; jaká z nich je nejlepší a lze ji spočítat?

3 Histogram

Histogram je funkce, která nám určuje počet pixelů dané barvy. V našem případě máme 256 různých barev (0-255) - pokud tedy spočítáme, kolikrát se která barva objevuje v našem obrázku a graf vykreslíme, dostaneme histogram:

```
imhist(I);
```

"Ruční" výpočet by vypadal takto:

```
Pocet=zeros(256);
for (x=1:512)
  for (y=1:512)
    Pocet(I(x,y))=Pocet(I(x,y))+1;
  end;
end;
plot(Pocet);
```

Z histogramu je patrné, že některé hodnoty barev nebyly v našem obrázku použity vůbec, zatímco jiné byly použity mnohokrát. Můžeme tedy všechny barvy roztáhnout tak, aby histogram opravdu využíval všechny hodnoty 0-255:

```
I2=histeq(I);
subplot(221); imshow(I); title ('original'); subplot(222); imhist(I);
subplot(223); imshow(I2); title ('equalized'); subplot(224); imhist(I2);
```

Po tomto příkladu subploty opět vypněte pomocí `subplot(111);` nebo `close all`

4 Lineární filtry

Jak už jsme si ukázali v předchozím příkladu, můžeme detekovat hrany tak, že od sebe odečteme hodnoty sousedních pixelů. Opačná funkce, tedy sečení hodnot sousedních pixelů, nám způsobí rozmazání obrazu. Funkce, které pracují s lineární kombinací okolí pixelu pro vyprodukování nové hodnoty, lze realizovat pomocí lineárních filtrů.

4.1 Blur (rozmazání obrazu)

Nejprve si ukážeme filtrování pomocí příkazu `imfilter`:

```
h = ones(5,5) / 25
I2 = imfilter(I,h);
imshow(I2);
```

Matice `h` nám zde představuje filtr. V tomto případě tedy posčítá všechny sousední pixely do vzdálenosti 2 a vyprodukuje průměrnou hodnotu. Abychom lépe pochopili, jak vše funguje, ukážeme si ruční výpočet:

```
h=ones(5,5)/25
I3=zeros(512,512);
for (x=3:509)
  for (y=3:509)
    for (x2=1:5)
      for (y2=1:5)
        I3(x, y)=I3(x, y)+h(x2, y2)*double(I(x+x2-3, y+y2-3));
      end;
    end;
  end;
end;
imshow(uint8(I3));
```

Procházíme tedy celý obraz a na každé pozici vypočítáme novou hodnotu v obrazu tak, že vynásobíme jádro filtru a odpovídající část obrazu.

4.2 Zaostření obrazu

Opačnou funkcí k rozmazání je zaostření obrazu. Tohoto efektu dosáhneme tak, že od hodnoty pixelu odečteme hodnoty sousedních pixelů.

```
h2=[-1 -1 -1; -1 9 -1; -1 -1 -1]
I2 = imfilter(I,h2);
imshow(I2);
```

4.3 Detekce hran

Jednoduchou detekci hran jsme si již ukázali - odpovídal by jí filtr [5 -5]. Problémem takového přístupu je značné zašumění, protože není uvažováno vůbec okolí pixelu. Dále takový detektor hran není schopen rozpoznat horizontální hrany. Abychom problémy odstranili, použijeme větší filtr a to dvakrát: jednou pro detekci vertikálních hran a jednou pro detekci horizontálních hran.

```
h=[1 0 -1; 2 0 -2; 1 0 -1]
I2 = imfilter(I,h);
h2=h'
I3 = imfilter(I,h2);
imshow(I2/2+I3/2);
```

5 Komprese à la JPEG pomocí DCT

Zřejmě každý uživatel počítače se už v dnešní době setkal s JPEG kompresí obrazu. V tomto příkladu si ukážeme její podstatu. Nejprve rozdělíme obraz na bloky 8x8 pixelů a na ty aplikujeme 2D diskrétní kosinovou transformaci (DCT). Tato transformace nám umožní oddělit složky obrazu s různou frekvencí. Poté "zahodíme" vysokofrekvenční složky obrazu, díky čemuž dosáhneme komprese. Následně zpětnou transformací dostaneme opět bloky 8x8 pixelů, ze kterých zrekonstruujeme obraz. Ačkoliv je patrná ztráta informace ve vysokých frekvencích (hrany jsou rozmazané), kvalita obrazu je pořád poměrně dobrá vzhledem k tomu, že jsme zahodili 58 z 64 koeficientů = 90.625% informace.

```
Id = im2double(I);
T = dctmtx(8)                              % definuje matici DCT bazi
dct = @(x)T * x * T';                      % definuje funkci, kteou budeme
       % aplikovat na kazdy 8x8 blok
                                           % prima DCT
B = blkproc(Id,[8 8],dct);                 % tady to udelame
mask = [1   1   1   0   0   0   0   0  % maska, ktera vybere jen
        1   1   0   0   0   0   0   0  % prvnich par koeficientu ...
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],@(x)mask.* x);        % vsechny bloky 8x8 se vymaskuji
invdct = @(x)T' * x * T;                   % definice zpetne DCT
I2 = blkproc(B2,[8 8],invdct);             % kterou zde aplikujeme.
imshow(I), figure, imshow(I2)
```

Ve skutečnosti je JPEG komprese samozřejmě komplikovanější - méně podstatná informace není zahozena, ale uložena s menší přesností (na menší počet bitů). To samé platí o barevných složkách obrazu, protože lidské oko je na různé barvy jinak citlivé.

Příklad z http://www.mathworks.com/access/helpdesk/help/toolbox/images/f21-16366.html

Úkoly

1. podívejte se na řádky a sloupce matice `T`, která obsahuje DCT báze, např. takto:

   ```
   plot (1:8, T(1:3,:)')   % zobrazi prvni 3 radky
   plot (1:8, T')          % zobrazi vsechny radky
   plot (1:8, T(:,1:3))   % zobrazi prvni 3 sloupce
   plot (1:8, T)          % zobrazi vsechny sloupce
   ```

   Komentujte, které frekvence která báze "hledá".

2. zkuste ručně zkonstruovat následující obrázky:

   - celý černý.
   - celý bílý.
   - s vodorovnými proužky.
   - se svilými proužky.
   - "šachovnici" po jednoitlivých pixelech.

   Obrázky prožeňte uvedeným algoritmem a

   - Zobrazte a komentujte obsah matice `B`, zobrazte např.

     ```
     B (1:8, 1:8)
     B2 (1:8, 1:8)
     ```

   - Komentujte, jak kvalitní je výstup.

3. Řešení pomocí funkce `blockproc` je velmi efektivní, ale nevidíme přesně, co se děje. Zkuste příklad přepsat pomocí cyklů a "ručně" zpracovat každý blok 8x8.

6 Chyba v obraze

Provádíme-li pokusy s obrazem jako například v předchozím příkladu, je užitečné umět spočítat, jaké chyby jsme se dopustili. To můžeme provést například tak, že spočítáme průměrnou chybu na jeden pixel:

```
noise=0;
I2=im2uint8(I2);
for (x=1:512)
  for (y=1:512)
    noise=noise+double(abs(I(x,y)-I2(x,y)));
  end;
end;
noise=noise/512/512
```

7 Reference

http://www.mathworks.com/access/helpdesk/help/toolbox/images/f0-3373.html
http://www.fit.vutbr.cz/study/courses/ISS/public/pred/2d/aux.m

8 Řešení

Ruční zpracování DCT:

```
Id = im2double(I);
% prima dct
T = dctmtx(8)
B = zeros (512,512);
for x=1:8:511,
  for y=1:8:511,
    vysek = Id (x:(x+7),y:(y+7));
    dctvyseku = T * vysek * T';
    B(x:(x+7),y:(y+7)) = dctvyseku;
  end
end
% maskovani
mask = [1   1   1   0   0   0   0   0  % maska, ktera vybere jen
        1   1   0   0   0   0   0   0  % prvnich par koeficientu ...
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = zeros (512,512);
for x=1:8:511,
  for y=1:8:511,
    vysek = B (x:(x+7),y:(y+7));
    vymaskovano = vysek .* mask;
    B2(x:(x+7),y:(y+7)) = vymaskovano;
  end
end
% zpetna DCT
I2 = zeros (512,512);
for x=1:8:511,
  for y=1:8:511,
    vysek = B2 (x:(x+7),y:(y+7));
    zpet = T' * vysek * T;;
    I2(x:(x+7),y:(y+7)) = zpet;
  end
end
imshow(I), figure, imshow(I2)
```