Základy počítačovej grafiky a spracovania obrazu

Histogram, zašumenie a zmena veľkosti obrazu

Doc. RNDr. Milan Ftáčnik, CSc.

.

Výpočet histogramu

- Funkcia má tvar

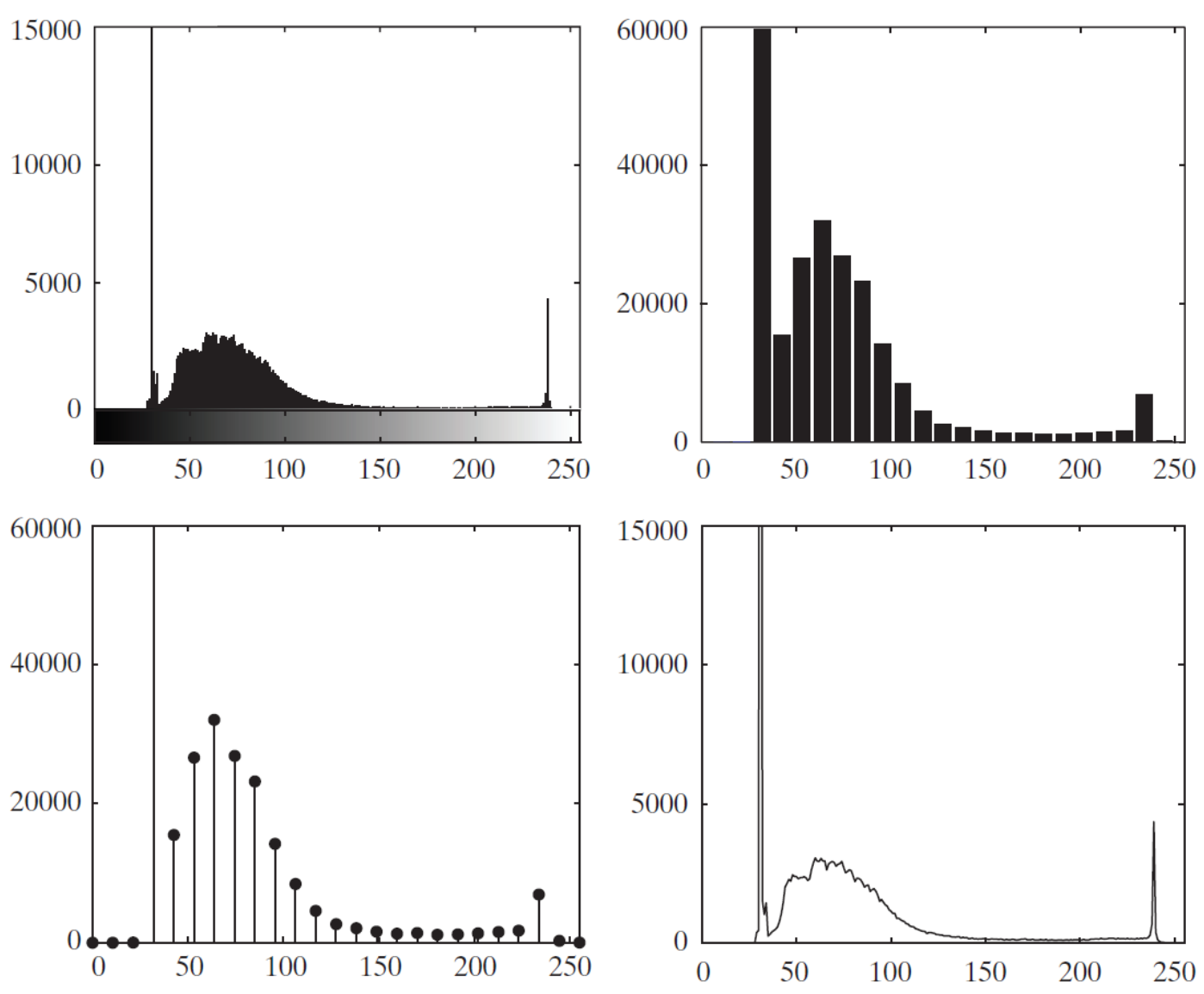
```
h = imhist(f, b)
```

- kde *b* je počet jasových úrovní, ak nie je uvedené, tak sa berie 256
- Normalizovaný histogram určíme ako

```
p = imhist(f, b)/numel(f)
```

- kde *numel(f)* je počet prvkov (pixlov) v *f*

Zobrazenie histogramu



Vypočítajte histogram Vášho obrazu

- Príkaz na výpočet histogramu obrazu *f*

```
h = imhist(f, b)
```

- Zobrazenie histogramu príkazom

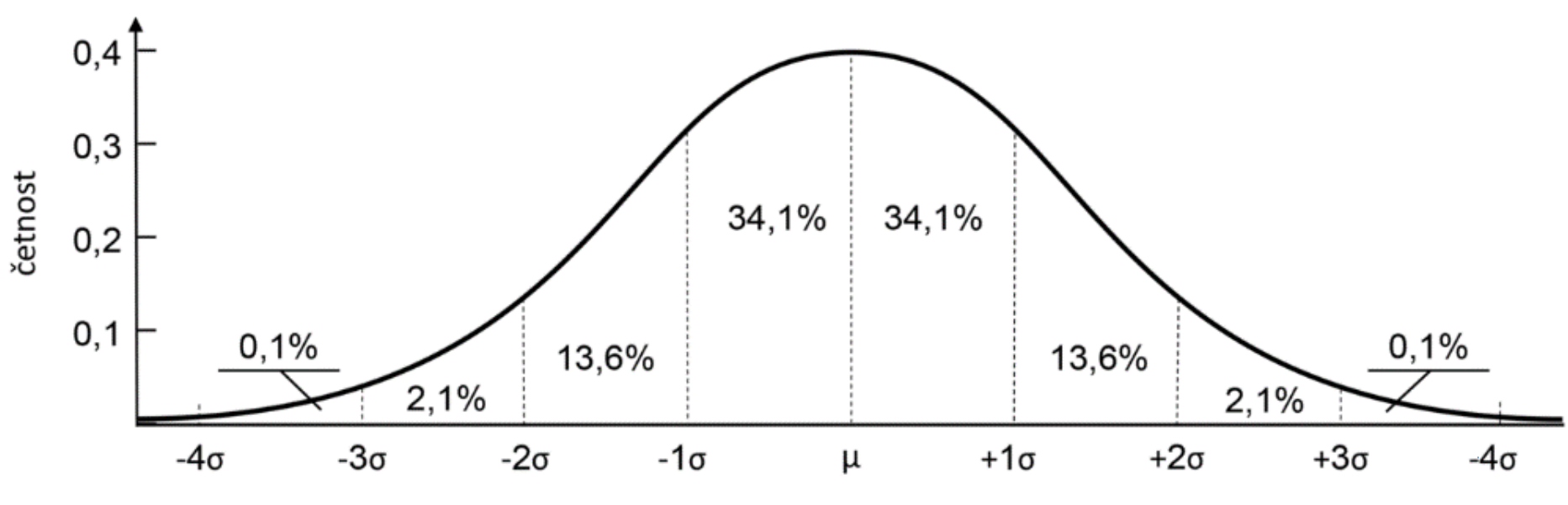
`plot(h)` **alebo** `bar(h)`

- Z histogramu je možné prečítať, či je obraz tmavý, svetlý, nízko kontrastný alebo vysoko kontrastný

Zašumenie obrazu

- Zvolíme aditívny Gaussovský šum – veľkosť smerodajnej odchýlky bude vstupným parametrom programu
- Ako druhý zvolíme čiernobiely šum (salt & pepper), pričom vstupom budú dve hodnoty (percento zašumenia čiernym šumom a percento zašumenia bielym šumom)

Gaussovský šum



Gaussovský šum 2

- Funkcia randn generuje náhodné čísla za predpokladu, že stredná hodnota $\mu = 0$ a odchýlka $\sigma = 1$
- Randn(size(A)) vygeneruje maticu náhodných čísiel o veľkosti rovnakej ako má obraz *A* s normálnym rozdelením *N(0,1)*
- b*randn(size(A)) vygeneruje maticu s rozdelením *N(0,b)*

Gaussovský šum 3

- Keď už máme maticu s náhodnými číslami a rozdelením *N(0,b)*, kde *b* vstup funkcie, tak spočítam túto maticu so vstupným obrazom (predtým z neho treba urobiť typ double), aby sa správne spočítali
- Súčet treba vrátiť do intervalu *(0,255)* alebo *(0,1)*, čo dosiahneme pomocou funkcie im2uint8 alebo orezaním hodnôt >1 na 1
- Potom výstup zobrazíme

Čierno-biely impulzný šum

- Chceme **20**% bieleho šumu, **10**% čierneho šumu, ostatná časť obrazu sa nezmení – tentoraz použijeme rand(size(A)) (náhodné čísla z uniformnej distribúcie)
- Každé vygenerované číslo nám reprezentuje pravdepodobnosť *p* (0 až 1), že práve tento pixel sa zafarbí na bielu/čiernu
- Tie pixely, pre ktoré bude *p* nižšia ako **10**%, zafarbíme na čierno, a tie, pre kt. bude *p* nižšia ako **20%** zafarbíme na bielo.

Čierno-biely impulzný šum

- Tie pixely, pre ktoré bude *p* nižšia ako **20**%, zafarbíme na bielo, a tie, pre kt. bude *p* nižšia ako **10%** zafarbíme na čierno
- → takto si ale časť bielo zafarbených pixelov vzápätí prefarbíme na čierno, čo nechceme
- → miesto podmienky *p<20%* použijeme pre biely šum podmienku *p>(100-20)%*, teda *p>80%*

Čierno-biely impulzný šum III

- Pixel prefarbíme na čiernu tak, že doň zapíšeme hodnotu *0,*
- Pixel prefarbíme na bielu tak, že doň zapíšeme hodnotu *255* (pre *uint8*).
- (ak máme obraz v *double*, biela bude reprezentovaná hodnotou *1*)

Zväčšovanie a zmenšovanie obrazu

- Zväčšovanie a zmenšovanie obrazu budeme robiť dvoma technikami – metódou najbližšieho suseda a bilineárnou interpoláciou a porovnáme výstup
- Parametre funkcie sú počet riadkov *new_rows* a počet stĺpcov *new_cols* výstupného obrazu *J*, pričom vstupný obraz *I* má rozmery old_rows a old_cols
- Pomôcka nasleduje:

Zmena veľkosti - najbližší sused

```
[old_rows old_cols] = size(I);
J = zeros(new_rows, new_cols, 'uint8');
Sr = old_rows/new_rows; Sc = old_cols/new_cols;
for r = 1: new_rows
    for c = 1: new_cols
        rf = r*Sr; cf = c*Sc;
        rf = floor(rf); cf =floor(cf);
        if rf < 1, rf = 1; end
        if cf < 1, cf = 1; end
        J(r,c) = I(rf,cf);
    end;
end;
```

Zmena veľkosti - najbližší sused II

- **function** J = *myimresize*(I, s)
- [old_rows old_cols] = size(I);
- r = round(linspace(1, old_rows, round(s*old_rows)));
- c = round(linspace(1, old_cols, round(s*old_cols)));
- J = I(r,c);
- **end**

- Porovnajte to s metódou imresize, ktorú poskytuje priamo MATLAB

Bilineárna interpolácia

- Nová hodnota zo 4 susedov $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$: za predpokladu, že tí majú súradnice: (0,0), (0,1), (1,0), (1,1)

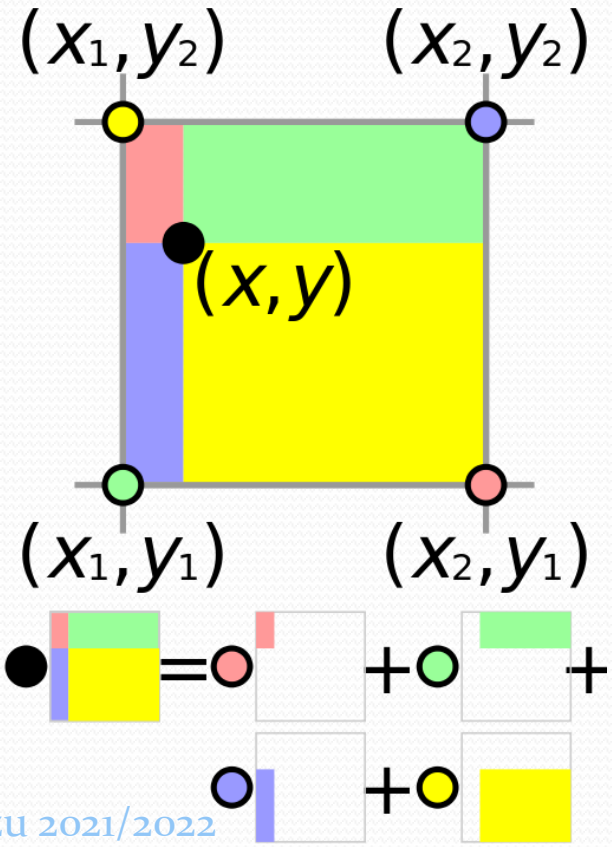
- $f(x, y) \approx$

$\approx f(0,0)(1-x)(1-y) +$

$+ f(1,0)x(1-y) +$

$+ f(0,1)(1-x)y +$

$+ f(1,1)xy$



Bilineárna interpolácia 2

Resampling Through Bilinear Interpolation

Let **I** be an $R \times C$ image.
We want to resize **I** to $R' \times C'$.
Call the new image **J**.

Let $s_R = R / R'$ and $s_C = C / C'$.

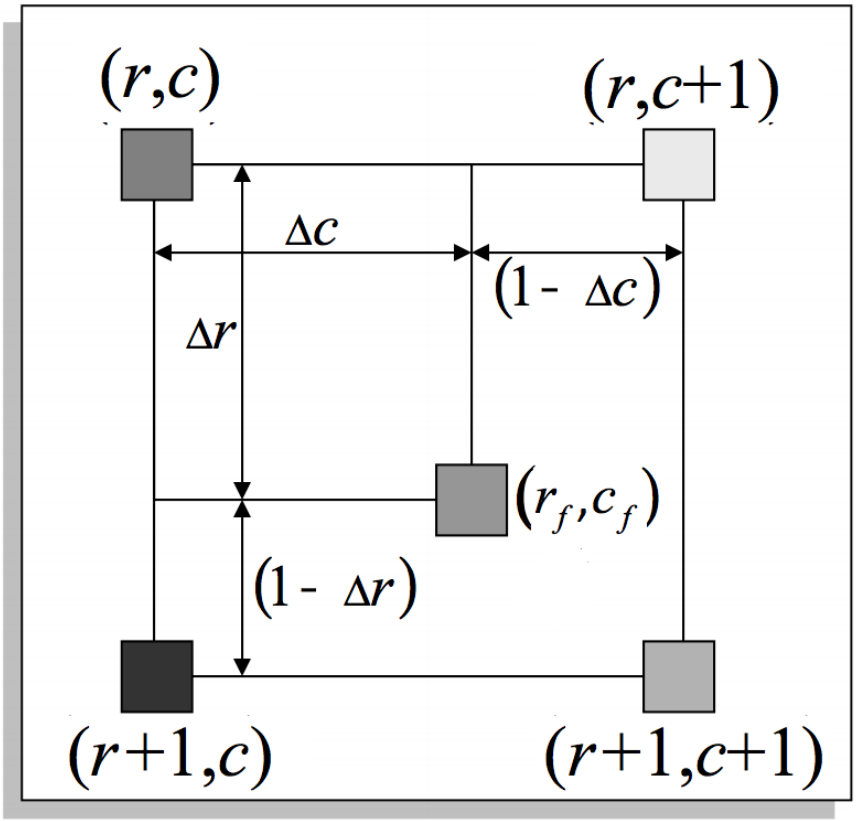
Let $r_f = r' \cdot s_R$ for $r' = 1, \ldots, R'$

and $c_f = c' \cdot s_C$ for $c' = 1, \ldots, C'$.

Let $r = \lfloor r_f \rfloor$ and $c = \lfloor c_f \rfloor$.

Let $\Delta r = r_f - r$ and $\Delta c = c_f - c$.

$$\begin{aligned}\text{Then } \mathbf{J}(r',c') = {} & \mathbf{I}(r,c)\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & +\mathbf{I}(r+1,c)\cdot\Delta r\cdot(1-\Delta c) \\ & +\mathbf{I}(r,c+1)\cdot(1-\Delta r)\cdot\Delta c \\ & +\mathbf{I}(r+1,c+1)\cdot\Delta r\cdot\Delta c.\end{aligned}$$



Zväčšenie a zmenšenie obrazu II

- Použite priložený obraz:
- Zmenšite ho faktorom 10 a potom ho zväčšite na pôvodnú veľkosť cez najbližšieho suseda a cez bilineárnu interpoláciu
- Výsledky porovnajte
- (pozor, obraz treba najprv previesť na šedoúrovňový - funkcia *rgb2gray(I)*)